Characterizing optimization bias in terms of optimization geometry

Suriya Gunasekar
TTI Chicago


Jason Lee
Princeton


Daniel Soudry
Technion


Nati Srebro
TTIC

Characterizing optimization bias in terms of optimization geometry

Suriya Gunasekar
TTI Chicago

Call it squared loss vs logistic loss talk

Jason Lee
Princeton

Daniel Soudry
Technion

Nati Srebro
TTIC

Deep learning

very large neural networks + large scale datasets

loss minimization over examples using variations of gradient descent (GD)

$$\min_{\mathbf{w}} \sum_{(\mathbf{x},y) \text{ in } D} \text{loss}\left(f(\mathbf{w}, \mathbf{x}), y \right)$$

Deep learning

very large neural networks + large scale datasets

loss minimization over examples using variations of gradient descent (GD)

$$\min_{\mathbf{w}} \sum_{(\mathbf{x},y) \text{ in } D} \text{loss}\big(f(\mathbf{w}, \mathbf{x}), y \big)$$

Overparamterized models: typically #parameters in $\mathbf{w}$ >> #examples in D

- multiple functions $f(\mathbf{w}, .)$ *globally* minimize the loss over D
- different minimizers have different performance on the learning problem of prediction over new samples

Deep learning

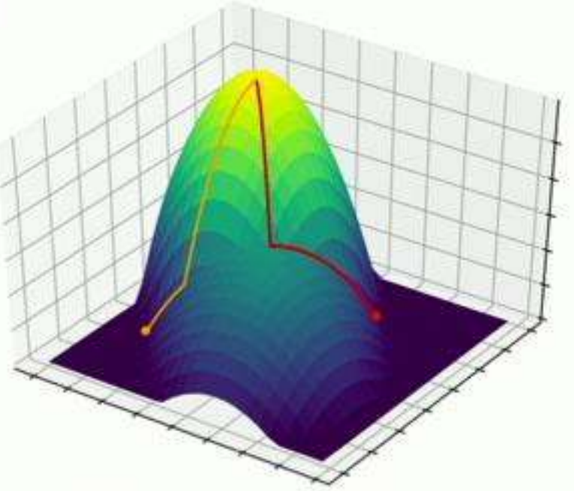
very large neural networks + large scale datasets

loss minimization over examples using variations of gradient descent (GD)

$$\min_{\mathbf{w}} \sum_{(\mathbf{x},y) \text{ in } D} \text{loss}\big(f(\mathbf{w}, \mathbf{x}), y \big)$$

Overparamterized models: typically #parameters in $\mathbf{w}$ >> #examples in D

- multiple functions $f(\mathbf{w}, .)$ *globally* minimize the loss over D
- different minimizers have different performance on the learning problem of prediction over new samples



different algorithms
→ different models

Deep learning

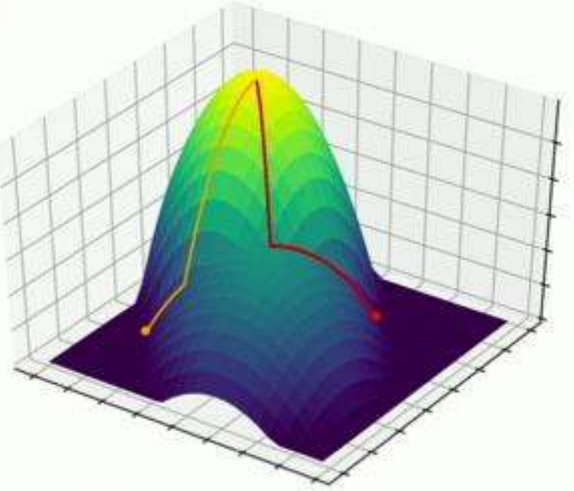
very large neural networks + large scale datasets

loss minimization over examples using variations of gradient descent (GD)

$$\min_{\mathbf{w}} \sum_{(\mathbf{x},y) \text{ in } D} \text{loss}\big(f(\mathbf{w}, \mathbf{x}), y \big)$$

Overparamterized models: typically #parameters in $\mathbf{w}$ >> #examples in D

- multiple functions $f(\mathbf{w}, .)$ *globally* minimize the loss over D
- different minimizers have different performance on the learning problem of prediction over new samples



different algorithms
→ different models

What does (stochastic) gradient descent converge to in overparamterized learning?

Back to basics: linear models

$$f_{\mathbf{w}}(\mathbf{x}) = \text{linear-function}(\mathbf{x})$$

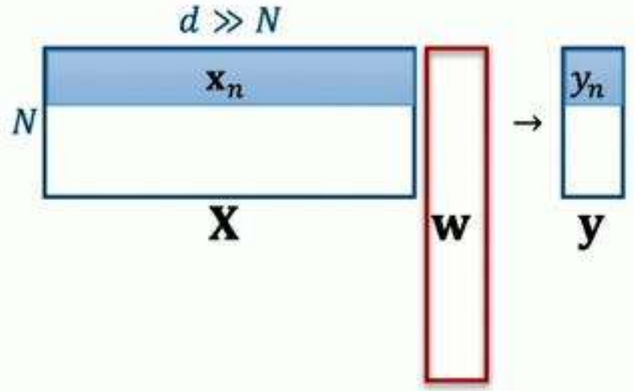
- Simple and natural starting point to build analytical tools
- Interesting and surprising results even in learning linear models
- Intuitions and empirical guidelines for neural networks

Linear regression

$$f(\mathbf{w}, \mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle$$

$$\min_{\mathbf{w}} L(\mathbf{w}) = \sum_{n=1}^{N} (\langle \mathbf{x}_n, \mathbf{w} \rangle - y_n)^2 = \|X\mathbf{w} - y\|_2^2$$

Many minimizers with zero loss

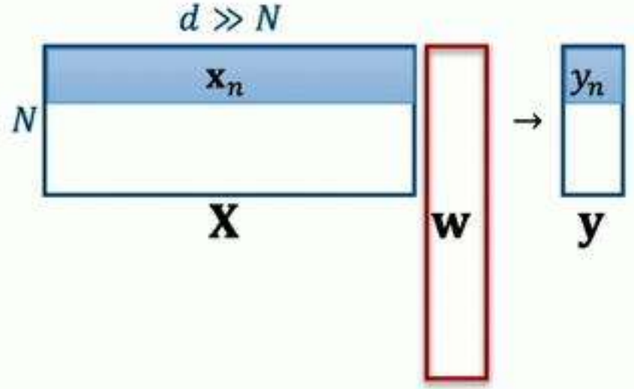


Linear regression

$$f(\mathbf{w}, \mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle$$

$$\min_{\mathbf{w}} L(\mathbf{w}) = \sum_{n=1}^{N} (\langle \mathbf{x}_n, \mathbf{w} \rangle - y_n)^2 = \|X\mathbf{w} - y\|_2^2$$

Many minimizers with zero loss



Gradient descent implicitly
minimizes the Euclidean norm of parameters

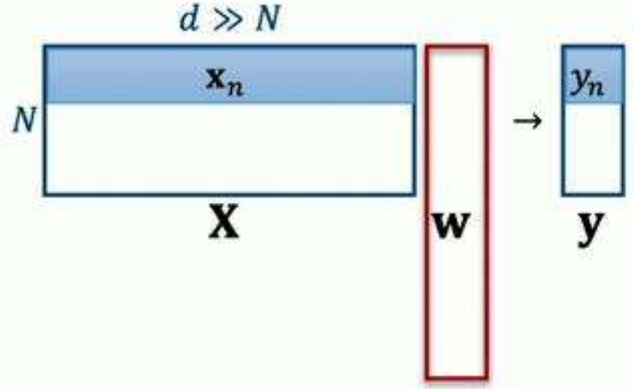
$$\mathbf{w}(t) \rightarrow \underset{X\mathbf{w}=y}{\operatorname{argmin}} \|\mathbf{w}\|_2 \quad \text{when } \mathbf{w}(0) = 0$$

Linear regression

$$f(\mathbf{w}, \mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle$$

$$\min_{\mathbf{w}} L(\mathbf{w}) = \sum_{n=1}^{N} (\langle \mathbf{x}_n, \mathbf{w} \rangle - y_n)^2 = \|X\mathbf{w} - y\|_2^2$$

Many minimizers with zero loss



Gradient descent implicitly
minimizes the Euclidean norm of parameters

$$\mathbf{w}(t) \rightarrow \operatorname*{argmin}_{X\mathbf{w}=y} \|\mathbf{w}\|_2 \quad \text{when } \mathbf{w}(0) = 0$$

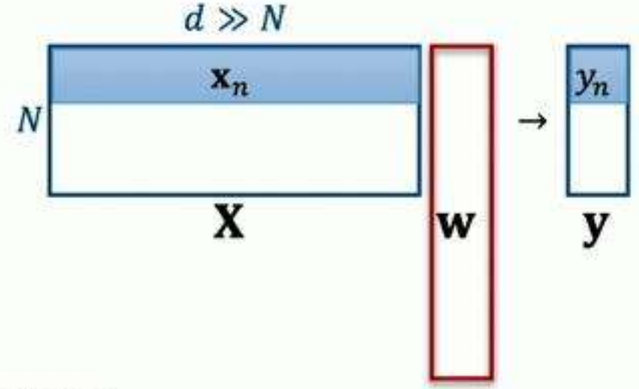
$$\mathbf{w}(t) \rightarrow \operatorname*{argmin}_{X\mathbf{w}=y} \|\mathbf{w} - \mathbf{w}(0)\|_2$$

Linear regression

$$f(\mathbf{w}, \mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle$$

$$\min_{\mathbf{w}} L(\mathbf{w}) = \sum_{n=1}^{N} (\langle \mathbf{x}_n, \mathbf{w} \rangle - y_n)^2 = \|X\mathbf{w} - y\|_2^2$$

Many minimizers with zero loss



Gradient descent implicitly
minimizes the Euclidean norm of parameters

$$\mathbf{w}(t) \rightarrow \underset{X\mathbf{w}=y}{\operatorname{argmin}} \|\mathbf{w}\|_2 \quad \text{when } \mathbf{w}(0) = 0$$

$$\mathbf{w}(t) \rightarrow \underset{X\mathbf{w}=y}{\operatorname{argmin}} \|\mathbf{w} - \mathbf{w}(0)\|_2$$

Proof sketch:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta_t \nabla L(\mathbf{w}(t))$$

$$\nabla L(\mathbf{w}(t)) = \sum_{n=1}^{N} (\langle \mathbf{x}_n, \mathbf{w}(t) \rangle - y_n)\mathbf{x}_n$$

updates lie on a low-dim. manifold
$\Delta \mathbf{w}(t) \in span(\mathbf{x}_n)$

Geometry induced by algorithms

GRADIENT DESCENT

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta_t \nabla L(\mathbf{w}(t))$$

$$\mathbf{w}(t+1) = \underset{\mathbf{w}}{\operatorname{argmin}} \, \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|_2^2$$

Geometry induced by algorithms

GRADIENT DESCENT

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta_t \nabla L(\mathbf{w}(t))$$

$$\mathbf{w}(t+1) = \underset{\mathbf{w}}{\text{argmin}} \, \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|_2^2$$

STEEPEST DESCENT w.r.t GENERAL NORM $\|.\|$

$$\mathbf{w}(t+1) = \underset{\mathbf{w}}{\text{argmin}} \, \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|^2$$

e.g., coordinate descent $\|.\| = \|.\|_1$

Geometry induced by algorithms

GRADIENT DESCENT

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta_t \nabla L(\mathbf{w}(t))$$

$$\mathbf{w}(t+1) = \underset{\mathbf{w}}{\operatorname{argmin}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|_2^2$$

STEEPEST DESCENT w.r.t GENERAL NORM $\|.\|$

$$\mathbf{w}(t+1) = \underset{\mathbf{w}}{\operatorname{argmin}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|^2$$

e.g., coordinate descent $\|.\| = \|.\|_1$

MIRROR DESCENT w.r.t strongly convex potential ψ

$$\mathbf{w}(t+1) = \underset{\mathbf{w}}{\operatorname{argmin}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{\eta_t} D_\psi(\mathbf{w}, \mathbf{w}(t))$$

e.g., exponentiated gradient descent

Geometry induced by algorithms

GRADIENT DESCENT

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta_t \nabla L(\mathbf{w}(t))$$

$$\mathbf{w}(t+1) = \underset{\mathbf{w}}{\operatorname{argmin}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|_2^2$$

STEEPEST DESCENT w.r.t GENERAL NORM $\|.\|$

$$\mathbf{w}(t+1) = \underset{\mathbf{w}}{\operatorname{argmin}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|^2$$

e.g., coordinate descent $\|.\| = \|.\|_1$

MIRROR DESCENT w.r.t strongly convex potential ψ

$$\mathbf{w}(t+1) = \underset{\mathbf{w}}{\operatorname{argmin}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{\eta_t} D_\psi(\mathbf{w}, \mathbf{w}(t))$$

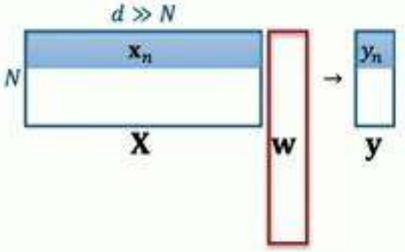
e.g., exponentiated gradient descent

Can we relate optimization bias to optimization geometry?

Algorithms in non-Euclidean geometry

- Mirror descent

$$\mathbf{w}(t+1) = \underset{\mathbf{w}}{\text{argmin}} \, \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{\eta_t} D_\psi(\mathbf{w}, \mathbf{w}(t))$$

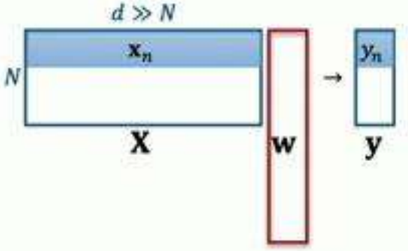


Gunasekar, Lee, Soudry, Srebro, (ICML 2018)

Algorithms in non-Euclidean geometry

- Mirror descent

$$\mathbf{w}(t+1) = \operatorname*{argmin}_{\mathbf{w}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{\eta_t} D_\psi(\mathbf{w}, \mathbf{w}(t))$$



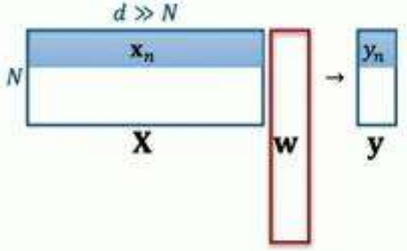
Theorem: for any step-size, instancewise SGD

$$\mathbf{w}(t) \to \operatorname*{argmin}_{X\mathbf{w}=y} D_\psi(\mathbf{w}, \mathbf{w}(0))$$

Gunasekar, Lee, Soudry, Srebro, (ICML 2018)

Algorithms in non-Euclidean geometry

- Mirror descent $\mathbf{w}(t+1) = \underset{\mathbf{w}}{\operatorname{argmin}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{\eta_t} D_\psi(\mathbf{w}, \mathbf{w}(t))$



Theorem: for any step-size, instancewise SGD

$$\mathbf{w}(t) \to \underset{X\mathbf{w}=y}{\operatorname{argmin}} D_\psi(\mathbf{w}, \mathbf{w}(0))$$

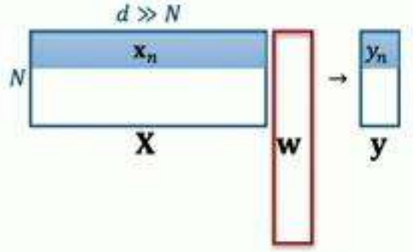
- Steepest descent $\mathbf{w}(t+1) = \underset{\mathbf{w}}{\operatorname{argmin}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|^2$

$$\mathbf{w}(t) \overset{?}{\to} \underset{\mathbf{w} \text{ minimizes } L(\mathbf{w})}{\operatorname{argmin}} \|\mathbf{w}\|$$

Gunasekar, Lee, Soudry, Srebro, (ICML 2018)

Algorithms in non-Euclidean geometry

- Mirror descent $\mathbf{w}(t+1) = \underset{\mathbf{w}}{\operatorname{argmin}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{\eta_t} D_\psi(\mathbf{w}, \mathbf{w}(t))$



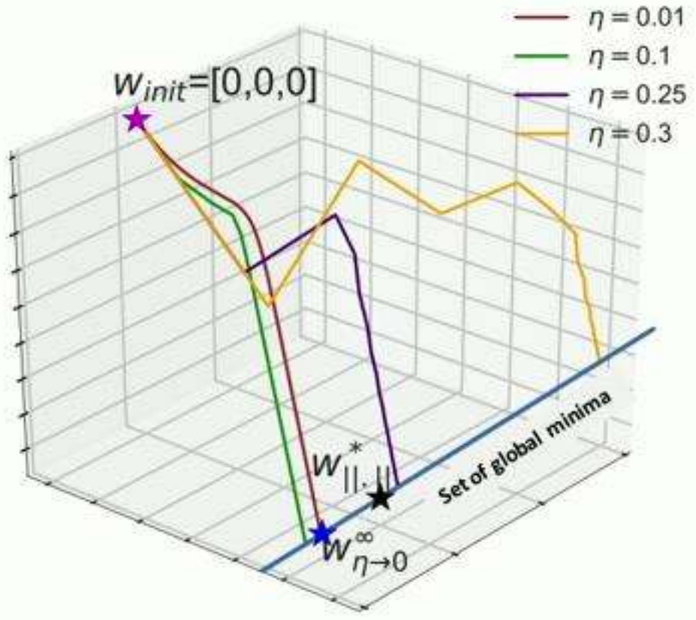
Theorem: for any step-size, instancewise SGD

$$\mathbf{w}(t) \to \underset{X\mathbf{w}=y}{\operatorname{argmin}} D_\psi(\mathbf{w}, \mathbf{w}(0))$$

- Steepest descent $\mathbf{w}(t+1) = \underset{\mathbf{w}}{\operatorname{argmin}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|^2$

$$\mathbf{w}(t) \overset{?}{\to} \underset{\mathbf{w} \text{ minimizes } L(\mathbf{w})}{\operatorname{argmin}} \|\mathbf{w}\|$$

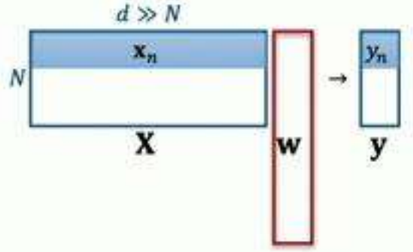
even for $\eta \to 0$: expected characterization does not hold



Gunasekar, Lee, Soudry, Srebro, (ICML 2018)

Algorithms in non-Euclidean geometry

- Mirror descent $\mathbf{w}(t+1) = \operatorname{argmin}_{\mathbf{w}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{\eta_t} D_\psi(\mathbf{w}, \mathbf{w}(t))$



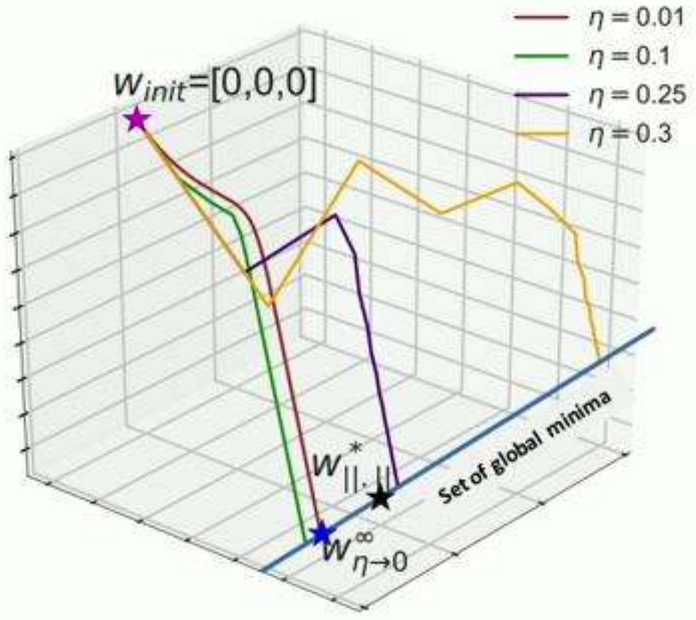
Theorem: for any step-size, instancewise SGD

$$\mathbf{w}(t) \to \operatorname*{argmin}_{X\mathbf{w}=y} D_\psi(\mathbf{w}, \mathbf{w}(0))$$

- Steepest descent $\mathbf{w}(t+1) = \operatorname{argmin}_{\mathbf{w}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|^2$

$$\mathbf{w}(t) \overset{?}{\to} \operatorname*{argmin}_{\mathbf{w} \text{ minimizes } L(\mathbf{w})} \|\mathbf{w}\|$$

even for $\eta \to 0$: expected characterization does not hold



Previously: ϵ-boosting traces ℓ_1 reg. path when the reg. path is monotone in each coordinate Efron, Hastie, Johnstone, Tibshirani, 2004

Gunasekar, Lee, Soudry, Srebro, (ICML 2018)

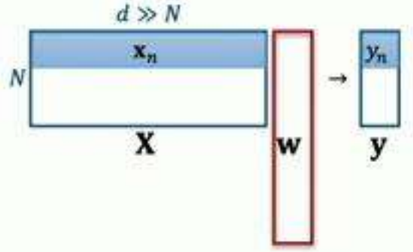
Geometry induced by parameters

- Many ways to parameterize a function class $\mathcal{F}$
  - e.g., standard parameterization of linear functions is $\mathbf{w} \in \mathbb{R}^d$
    - equivalent parameterizations: $w = u \cdot v$, $w = u * v$ for $u, v \in \mathbb{R}^d$

- Same algorithm (e.g., GD) on different parameterizations induce different biases in the function space

Algorithms in non-Euclidean geometry

- Mirror descent $\mathbf{w}(t+1) = \underset{\mathbf{w}}{\operatorname{argmin}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{\eta_t} D_\psi(\mathbf{w}, \mathbf{w}(t))$



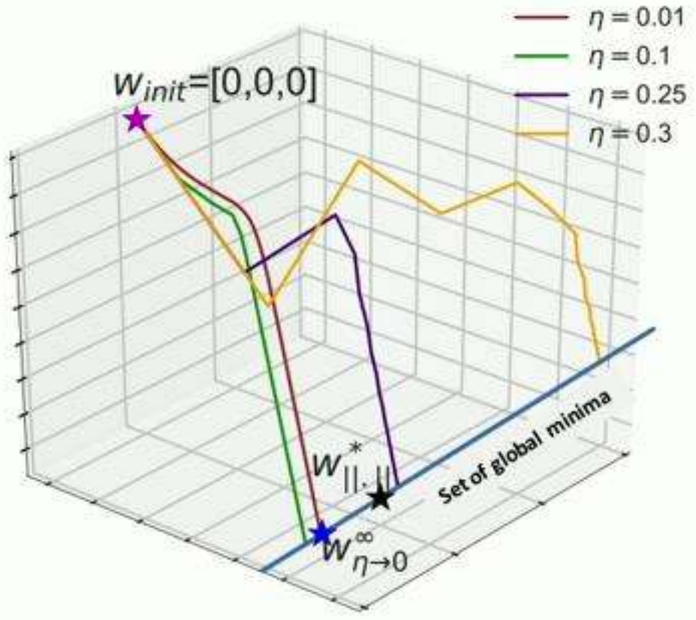
Theorem: for any step-size, instancewise SGD

$$\mathbf{w}(t) \to \underset{X\mathbf{w}=y}{\operatorname{argmin}} D_\psi(\mathbf{w}, \mathbf{w}(0))$$

- Steepest descent $\mathbf{w}(t+1) = \underset{\mathbf{w}}{\operatorname{argmin}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|^2$

$$\mathbf{w}(t) \overset{?}{\to} \underset{\mathbf{w} \text{ minimizes } L(\mathbf{w})}{\operatorname{argmin}} \|\mathbf{w}\|$$

even for $\eta \to 0$: expected characterization does not hold



Previously: ϵ-boosting traces ℓ_1 reg. path when the reg. path is monotone in each coordinate Efron, Hastie, Johnstone, Tibshirani, 2004

Gunasekar, Lee, Soudry, Srebro, (ICML 2018)

Geometry induced by parameters

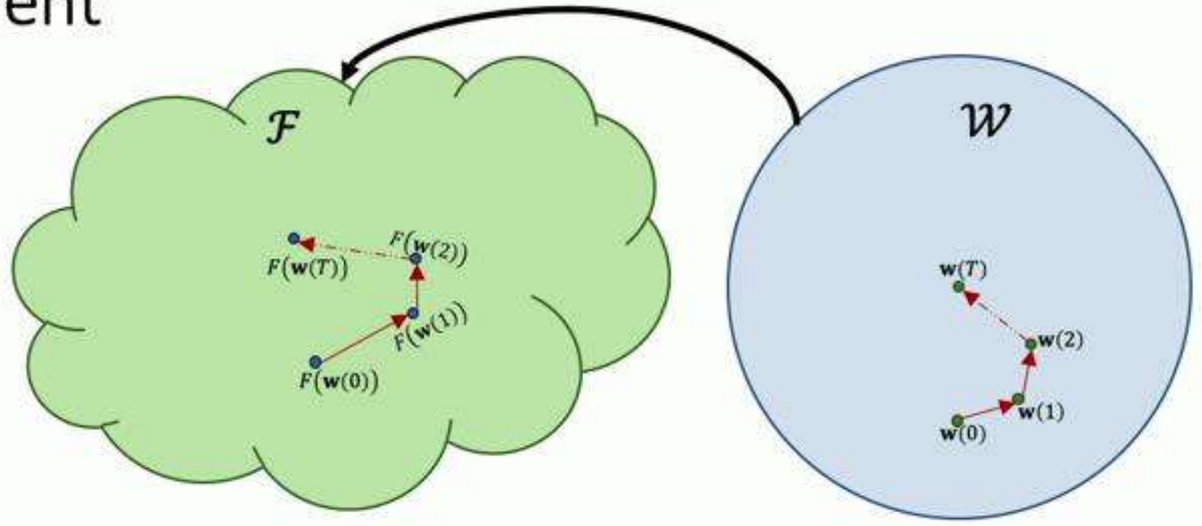
- Many ways to parameterize a function class $\mathcal{F}$
  - e.g., standard parameterization of linear functions is $\mathbf{w} \in \mathbb{R}^d$
    - equivalent parameterizations: $w = u \cdot v$, $w = u * v$ for $u, v \in \mathbb{R}^d$

- Same algorithm (e.g., GD) on different parameterizations induce different biases in the function space

Geometry induced by parameters

- Many ways to parameterize a function class $\mathcal{F}$
  - e.g., standard parameterization of linear functions is $\mathbf{w} \in \mathbb{R}^d$
    - equivalent parameterizations: $w = u \cdot v$, $w = u * v$ for $u, v \in \mathbb{R}^d$

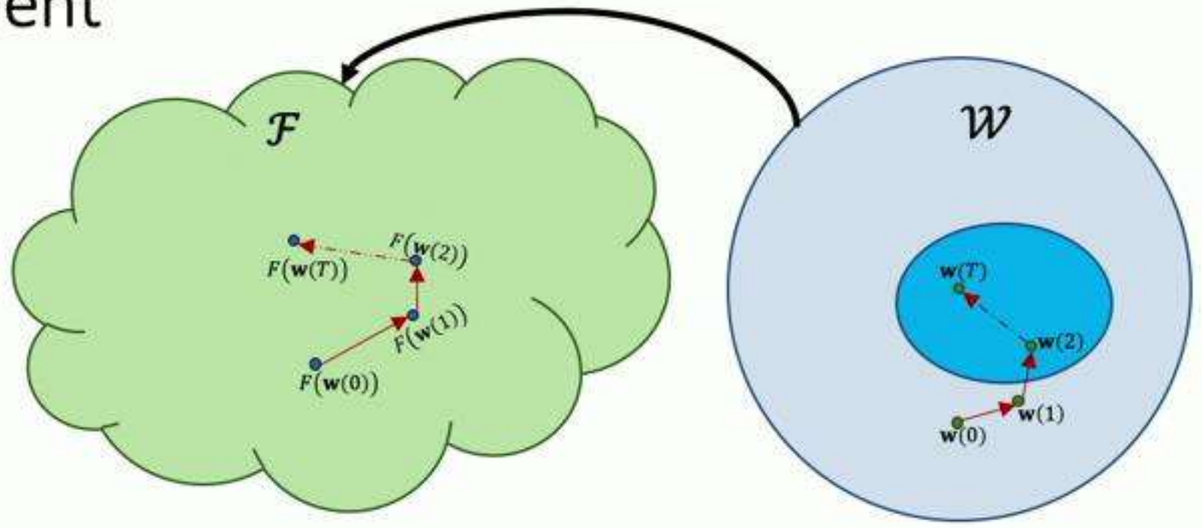
- Same algorithm (e.g., GD) on different parameterizations induce different biases in the function space



Geometry induced by parameters

- Many ways to parameterize a function class $\mathcal{F}$
  - e.g., standard parameterization of linear functions is $\mathbf{w} \in \mathbb{R}^d$
    - equivalent parameterizations: $w = u \cdot v, w = u * v$ for $u, v \in \mathbb{R}^d$

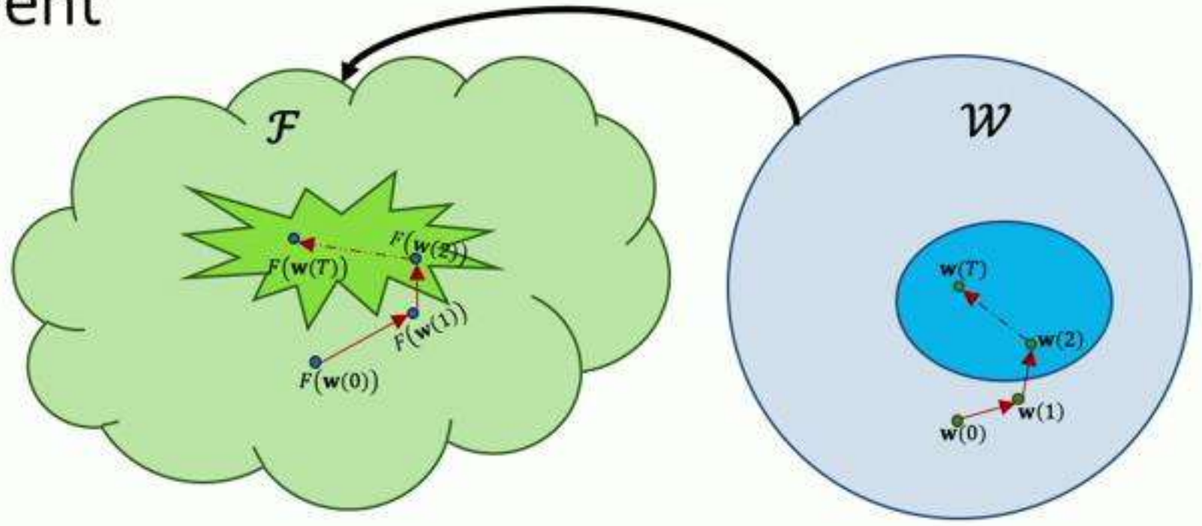
- Same algorithm (e.g., GD) on different parameterizations induce different biases in the function space



Geometry induced by parameters

- Many ways to parameterize a function class $\mathcal{F}$
  - e.g., standard parameterization of linear functions is $\mathbf{w} \in \mathbb{R}^d$
    - equivalent parameterizations: $w = u \cdot v$, $w = u * v$ for $u, v \in \mathbb{R}^d$

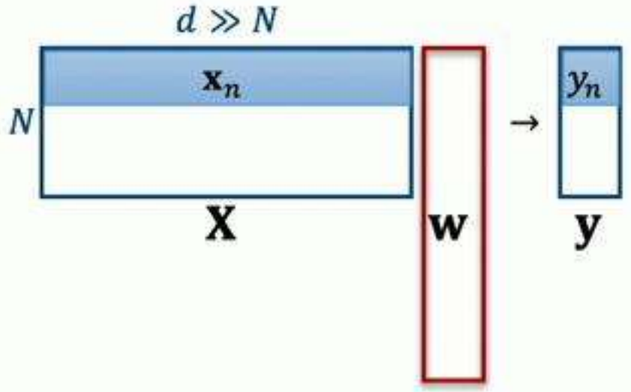
- Same algorithm (e.g., GD) on different parameterizations induce different biases in the function space



Geometry induced by parameters

$$\min_{\mathbf{w}} L(\mathbf{w}) := \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

GD converges to the **minimum Euclidean norm** solution

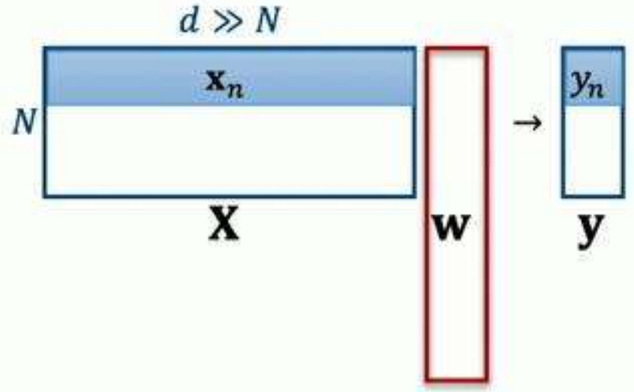


Gunasekar, Woodworth, Bhojanapalli, Neyshabur, Srebro, (NIPS 2017)

Geometry induced by parameters

$$\min_{\mathbf{w}} L(\mathbf{w}) := \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

GD converges to the **minimum Euclidean norm** solution



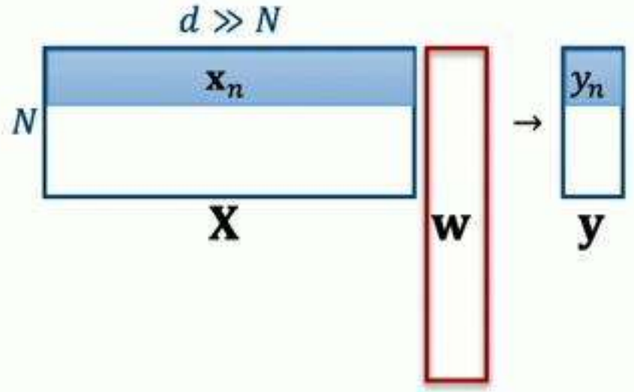
$$\min_{\mathbf{u},\mathbf{v}} \tilde{L}(\mathbf{u}, \mathbf{v}) := \|\mathbf{X}\mathbf{u} \cdot \mathbf{v} - \mathbf{y}\|_2^2$$

Gunasekar, Woodworth, Bhojanapalli, Neyshabur, Srebro, (NIPS 2017)

Geometry induced by parameters

$$\min_{\mathbf{w}} L(\mathbf{w}) := \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

GD converges to the **minimum Euclidean norm** solution



$$\min_{\mathbf{u},\mathbf{v}} \tilde{L}(\mathbf{u},\mathbf{v}) := \|\mathbf{X}\mathbf{u}\cdot\mathbf{v} - \mathbf{y}\|_2^2$$

Theorem: Gradient descent with infinitesimal step size and initialization returns **minimum ℓ_1 norm solution**

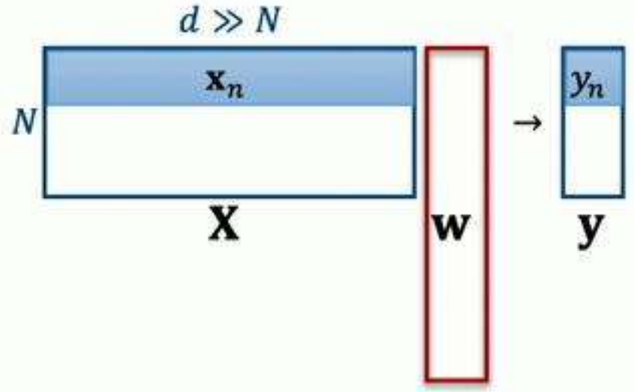
$$\text{GD on } (\mathbf{u},\mathbf{v}) \rightarrow \operatorname*{argmin}_{\mathbf{X}\mathbf{w}=\mathbf{y}} \|\mathbf{w}\|_1$$

Gunasekar, Woodworth, Bhojanapalli, Neyshabur, Srebro, (NIPS 2017)

Geometry induced by parameters

$$\min_{\mathbf{w}} L(\mathbf{w}) := \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

GD converges to the **minimum Euclidean norm** solution



$$\min_{\mathbf{u},\mathbf{v}} \tilde{L}(\mathbf{u},\mathbf{v}) := \|\mathbf{X}\mathbf{u}\cdot\mathbf{v} - \mathbf{y}\|_2^2$$

$$\|\mathbf{w}\|_1 \propto \min_{\mathbf{w}=\mathbf{u}\cdot\mathbf{v}} \|\mathbf{u}\|_2^2 + \|\mathbf{v}\|_2^2$$

Theorem: Gradient descent with infinitesimal step size and initialization returns **minimum ℓ_1 norm solution**

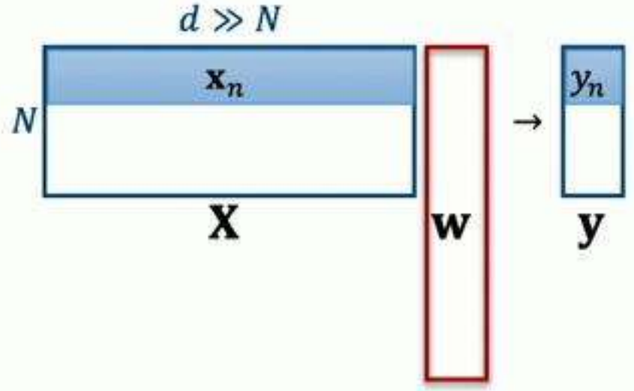
$$\text{GD on } (\mathbf{u},\mathbf{v}) \to \operatorname*{argmin}_{\mathbf{X}\mathbf{w}=\mathbf{y}} \|\mathbf{w}\|_1$$

Gunasekar, Woodworth, Bhojanapalli, Neyshabur, Srebro, (NIPS 2017)

Geometry induced by parameters

$$\min_{\mathbf{w}} L(\mathbf{w}) := \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

GD converges to the **minimum Euclidean norm** solution



$$\min_{\mathbf{u},\mathbf{v}} \tilde{L}(\mathbf{u},\mathbf{v}) := \|\mathbf{X}\mathbf{u}\cdot\mathbf{v} - \mathbf{y}\|_2^2$$

Theorem: Gradient descent with infinitesimal step size and initialization returns **minimum ℓ_1 norm solution**

$$\text{GD on } (\mathbf{u},\mathbf{v}) \to \operatorname*{argmin}_{\mathbf{X}\mathbf{w}=\mathbf{y}} \|\mathbf{w}\|_1$$

$$\|\mathbf{w}\|_1 \propto \min_{\mathbf{w}=\mathbf{u}\cdot\mathbf{v}} \|\mathbf{u}\|_2^2 + \|\mathbf{v}\|_2^2$$

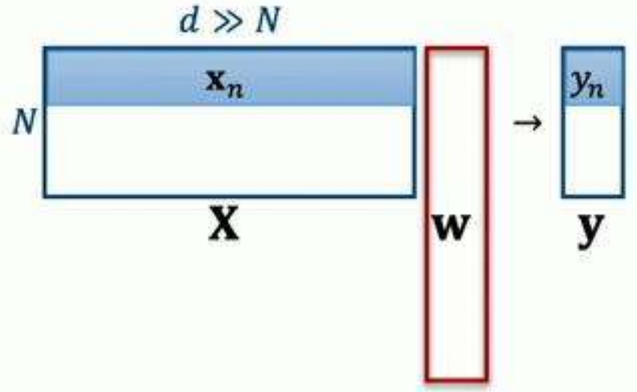
- implicit bias related to Euclidean norm of parameters
- but, like steepest descent, implicit bias depends on step-size, and characterization is holds only for infinitesimal initialization

Gunasekar, Woodworth, Bhojanapalli, Neyshabur, Srebro, (NIPS 2017)

Geometry induced by parameters

$$\min_{\mathbf{w}} L(\mathbf{w}) := \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

GD converges to the **minimum Euclidean norm** solution



$$\min_{\mathbf{u},\mathbf{v}} \tilde{L}(\mathbf{u},\mathbf{v}) := \|\mathbf{X}\mathbf{u}\cdot\mathbf{v} - \mathbf{y}\|_2^2$$

Theorem: Gradient descent with infinitesimal step size and initialization returns **minimum ℓ_1 norm solution**

$$\text{GD on } (\mathbf{u},\mathbf{v}) \rightarrow \operatorname*{argmin}_{\mathbf{X}\mathbf{w}=\mathbf{y}} \|\mathbf{w}\|_1$$

$$\|\mathbf{w}\|_1 \propto \min_{\mathbf{w}=\mathbf{u}\cdot\mathbf{v}} \|\mathbf{u}\|_2^2 + \|\mathbf{v}\|_2^2$$

- implicit bias related to Euclidean norm of parameters
- but, like steepest descent, implicit bias depends on step-size, and characterization is holds only for infinitesimal initialization
- updates still in low dimensional manifold but a non-affine manifold

Gunasekar, Woodworth, Bhojanapalli, Neyshabur, Srebro, (NIPS 2017)

Implicit bias for squared loss

- General mirror descent: characterization for all initializations $\mathbf{w}(t) \to \underset{X\mathbf{w}=y}{\operatorname{argmin}} D_\psi(\mathbf{w}, \mathbf{w}(0))$
  - ⇢ for any step-size, dual momentum, instancewise SGD

Implicit bias for squared loss

- General mirror descent: characterization for all initializations $\mathbf{w}(t) \to \underset{X\mathbf{w}=y}{\operatorname{argmin}} D_\psi(\mathbf{w}, \mathbf{w}(0))$
  - ⇢ for any step-size, dual momentum, instancewise SGD
- General steepest descent: expected characterization does not hold $\mathbf{w}(t) \not\to \underset{X\mathbf{w}=y}{\operatorname{argmin}} \|\mathbf{w} - \mathbf{w}(0)\|$
  - ⇢ even for $\|.\|^2$ is strongly convex and infinitesimal step size

Implicit bias for squared loss

- General mirror descent: characterization for all initializations $\mathbf{w}(t) \to \underset{X\mathbf{w}=y}{\operatorname{argmin}} D_\psi(\mathbf{w}, \mathbf{w}(0))$
  - ⇢ for any step-size, dual momentum, instancewise SGD
- General steepest descent: expected characterization does not hold $\mathbf{w}(t) \not\to \underset{X\mathbf{w}=y}{\operatorname{argmin}} \|\mathbf{w} - \mathbf{w}(0)\|$
  - ⇢ even for $\|.\|^2$ is strongly convex and infinitesimal step size
- Geometry induced by parameters:

  Matrix product: $\mathbf{W} = \mathbf{UV}$ (G, Woodworth, Bhojanapalli, Neyshabur, Srebro 2017; Li, Zhang, Ma 2018)
  - ⇢ experiments: minimum nuclear norm solution with small initialization and step size
  - ⇢ proven for special RIP measurements

Implicit bias for squared loss

- General mirror descent: characterization for all initializations $\mathbf{w}(t) \to \underset{X\mathbf{w}=y}{\operatorname{argmin}} D_\psi(\mathbf{w}, \mathbf{w}(0))$
  - ⇢ for any step-size, dual momentum, instancewise SGD
- General steepest descent: expected characterization does not hold $\mathbf{w}(t) \not\to \underset{X\mathbf{w}=y}{\operatorname{argmin}} \|\mathbf{w} - \mathbf{w}(0)\|$
  - ⇢ even for $\|.\|^2$ is strongly convex and infinitesimal step size
- Geometry induced by parameters:

  Matrix product: $\mathbf{W} = \mathbf{UV}$ (G, Woodworth, Bhojanapalli, Neyshabur, Srebro 2017; Li, Zhang, Ma 2018)
  - ⇢ experiments: minimum nuclear norm solution with small initialization and step size
  - ⇢ proven for special RIP measurements

  Elementwise product: $\mathbf{w} = \mathbf{u}.\mathbf{v}$
  - ⇢ with infinitesimal initialization and step size → minimum ℓ_1 norm solution

Implicit bias for squared loss

- General mirror descent: characterization for all initializations $\mathbf{w}(t) \to \underset{X\mathbf{w}=y}{\operatorname{argmin}} D_\psi(\mathbf{w}, \mathbf{w}(0))$
  - ⇢ for any step-size, dual momentum, instancewise SGD
- General steepest descent: expected characterization does not hold $\mathbf{w}(t) \not\to \underset{X\mathbf{w}=y}{\operatorname{argmin}} \|\mathbf{w} - \mathbf{w}(0)\|$
  - ⇢ even for $\|.\|^2$ is strongly convex and infinitesimal step size
- Geometry induced by parameters:

  Matrix product: $\mathbf{W} = \mathbf{UV}$ (G, Woodworth, Bhojanapalli, Neyshabur, Srebro 2017; Li, Zhang, Ma 2018)
  - ⇢ experiments: minimum nuclear norm solution with small initialization and step size
  - ⇢ proven for special RIP measurements

  Elementwise product: $\mathbf{w} = \mathbf{u}.\mathbf{v}$
  - ⇢ with infinitesimal initialization and step size → minimum ℓ_1 norm solution
  - ⇢ Follow up: scale of initialization plays a crucial role: interpolates between minimizing ℓ_1 norm and minimizing ℓ_2 norm (Woodworth, G, Lee, Soudry, Srebro 2019)

Implicit bias for squared loss

- General mirror descent: characterization for all initializations $\mathbf{w}(t) \to \underset{X\mathbf{w}=y}{\operatorname{argmin}} D_\psi(\mathbf{w}, \mathbf{w}(0))$
  - ⇢ for any step-size, dual momentum, instancewise SGD
- General steepest descent: expected characterization does not hold $\mathbf{w}(t) \not\to \underset{X\mathbf{w}=y}{\operatorname{argmin}} \|\mathbf{w} - \mathbf{w}(0)\|$
  - ⇢ even for $\|.\|^2$ is strongly convex and infinitesimal step size
- Geometry induced by parameters:

  Matrix product: $\mathbf{W} = \mathbf{UV}$ (G, Woodworth, Bhojanapalli, Neyshabur, Srebro 2017; Li, Zhang, Ma 2018)
  - ⇢ experiments: minimum nuclear norm solution with small initialization and step size
  - ⇢ proven for special RIP measurements

  Elementwise product: $\mathbf{w} = \mathbf{u}.\mathbf{v}$
  - ⇢ with infinitesimal initialization and step size → minimum ℓ_1 norm solution
  - ⇢ Follow up: scale of initialization plays a crucial role: interpolates between minimizing ℓ_1 norm and minimizing ℓ_2 norm (Woodworth, G, Lee, Soudry, Srebro 2019)
  - ⇢ Follow up: get minimum ℓ_1 norm solution even for $\mathbf{w} = \mathbf{u_1}.\mathbf{u_2}.\ldots\mathbf{u_k}$ - implicit bias is not directly related to Euclidean norm of parameters (Arora, Cohen, Hu, Luo 2019)

Implicit bias for squared loss

Optimization geometry directs optimization bias, but expected characterization does not always hold

- ⇢ initialization and step size crucially influence the optimization bias
- ⇢ precise characterization of the nature of implicit bias is elusive even in simple linear models

New tools required to analyze the nature of optimization bias

Implicit bias for squared loss

Optimization geometry directs optimization bias, but expected characterization does not always hold

- ⇢ initialization and step size crucially influence the optimization bias
- ⇢ precise characterization of the nature of implicit bias is elusive even in simple linear models

New tools required to analyze the nature of optimization bias



But why are we not looking at logistic loss?

Implicit bias for squared loss

Optimization geometry directs optimization bias, but expected characterization does not always hold

- ⇢ initialization and step size crucially influence the optimization bias
- ⇢ precise characterization of the nature of implicit bias is elusive even in simple linear models

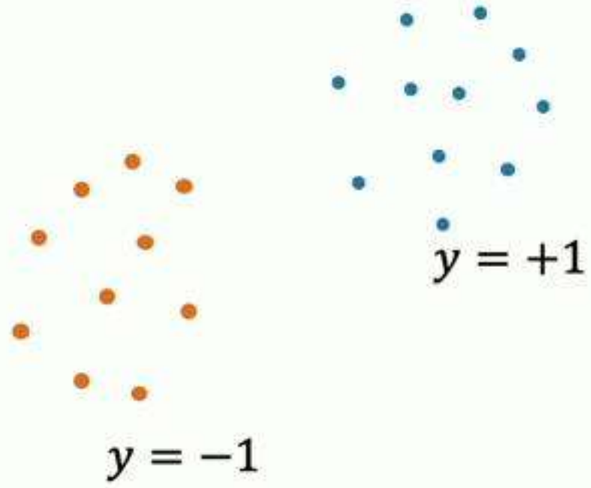
New tools required to analyze the nature of optimization bias



But why are we not looking at logistic loss?

Part II. Nature of optimization bias is very different for some classification losses

Linear classification using logistic loss



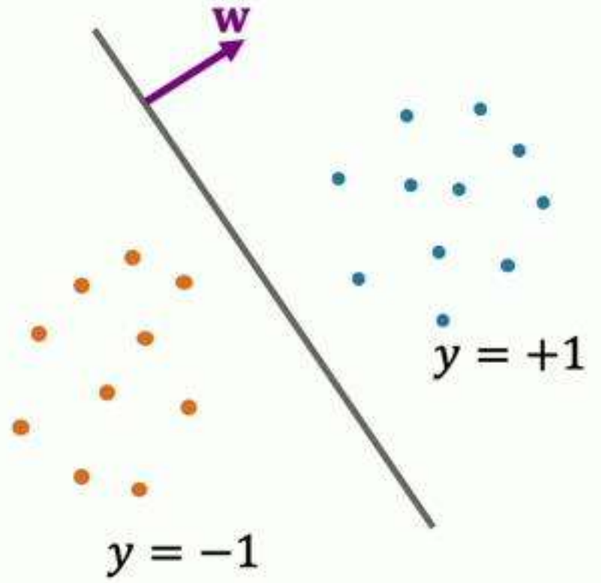
Linear classification using logistic loss

$$\min_{\mathbf{w}} \sum_{(\mathbf{x},y) \text{ in } D} \text{loss}\big(\langle \mathbf{w}, \mathbf{x} \rangle, y\big)$$

$\log(1 + \exp(-y\langle \mathbf{w}, \mathbf{x} \rangle))$

$\exp(-y\langle \mathbf{w}, \mathbf{x} \rangle)$

$$\text{prediction} = \text{sign}\left(\langle \mathbf{w}, \mathbf{x} \rangle\right)$$

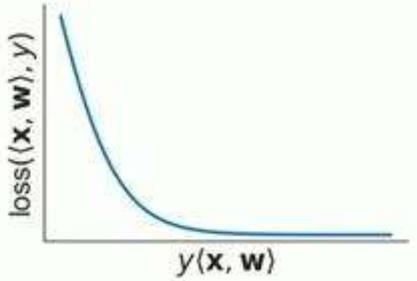


Linear classification using logistic loss

$$\min_{\mathbf{w}} \sum_{(\mathbf{x},y) \text{ in } D} \text{loss}\big(\langle \mathbf{w}, \mathbf{x} \rangle, y\big)$$

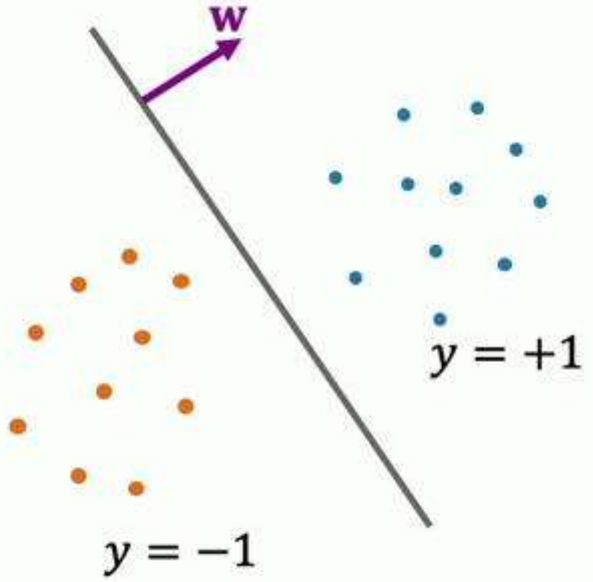
$\log(1 + \exp(-y\langle \mathbf{w}, \mathbf{x} \rangle))$

$\exp(-y\langle \mathbf{w}, \mathbf{x} \rangle)$



- no finite minimizers
- gradient descent iterates $\mathbf{w}(t)$ will diverge in norm

$$\text{prediction} = \text{sign}\left(\langle \mathbf{w}, \mathbf{x} \rangle\right)$$

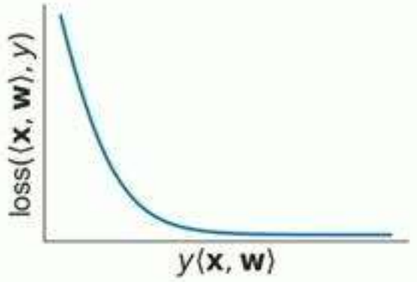


Linear classification using logistic loss

$$\min_{\mathbf{w}} \sum_{(\mathbf{x},y) \text{ in } D} \text{loss}\big(\langle \mathbf{w}, \mathbf{x} \rangle, y\big)$$

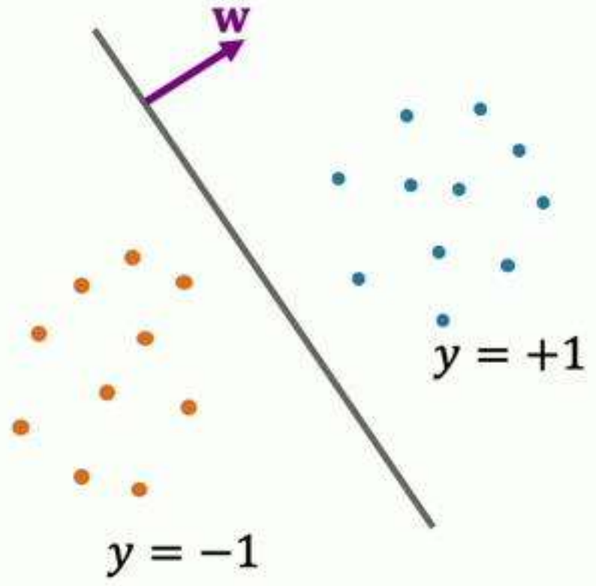
$\log(1 + \exp(-y\langle \mathbf{w}, \mathbf{x} \rangle))$

$\exp(-y\langle \mathbf{w}, \mathbf{x} \rangle)$



- no finite minimizers
- gradient descent iterates $\mathbf{w}(t)$ will diverge in norm
- but what about direction$(\mathbf{w}(t))$, *i.e.*, $\frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|}$?
  - ⇢ different separating directions = different classifiers

prediction $= \text{sign}\,(\langle \mathbf{w}, \mathbf{x} \rangle)$

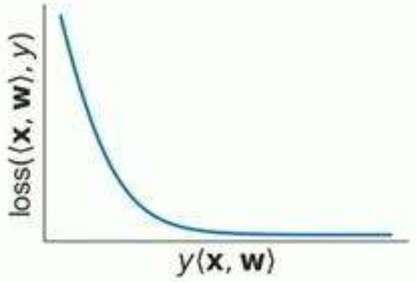


Linear classification using logistic loss

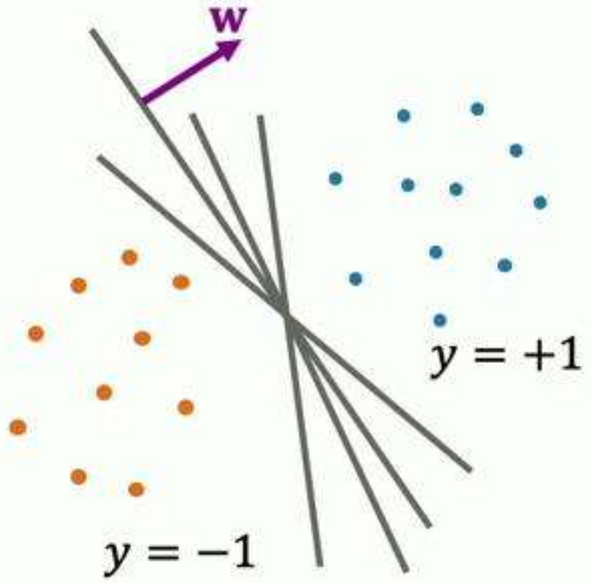
$$\min_{\mathbf{w}} \sum_{(\mathbf{x},y) \text{ in } D} \text{loss}\big(\langle \mathbf{w}, \mathbf{x} \rangle, y\big)$$

$\log(1 + \exp(-y\langle \mathbf{w}, \mathbf{x} \rangle))$

$\exp(-y\langle \mathbf{w}, \mathbf{x} \rangle)$



$\text{prediction} = \text{sign}\left(\langle \mathbf{w}, \mathbf{x} \rangle\right)$



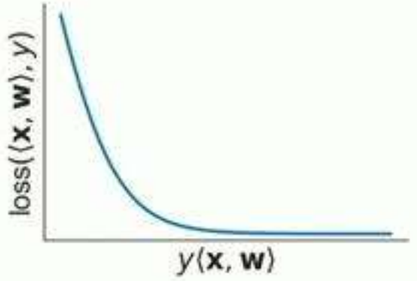
- no finite minimizers
- gradient descent iterates $\mathbf{w}(t)$ will diverge in norm
- but what about direction$(\mathbf{w}(t))$, *i.e.*, $\frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|}$?
  - ⇢ different separating directions = different classifiers

Linear classification using logistic loss

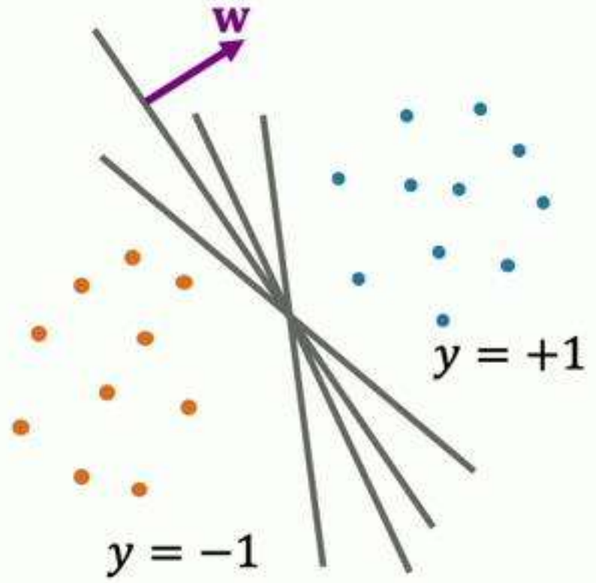
$$\min_{\mathbf{w}} \sum_{(\mathbf{x},y) \text{ in } D} \text{loss}\big(\langle \mathbf{w}, \mathbf{x} \rangle, y\big)$$

$\log(1 + \exp(-y\langle \mathbf{w}, \mathbf{x} \rangle))$

$\exp(-y\langle \mathbf{w}, \mathbf{x} \rangle)$



$$\text{prediction} = \text{sign}\left(\langle \mathbf{w}, \mathbf{x} \rangle\right)$$

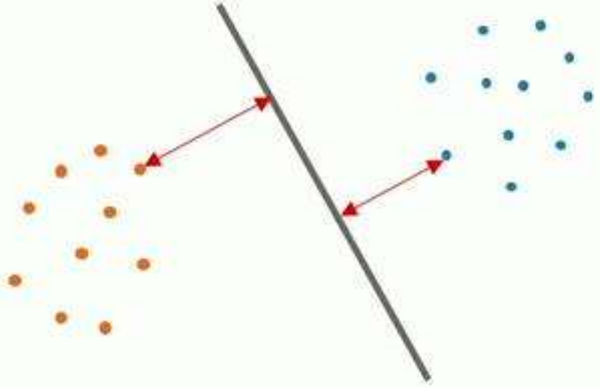


- no finite minimizers
- gradient descent iterates $\mathbf{w}(t)$ will diverge in norm
- but what about direction$(\mathbf{w}(t))$, *i.e.*, $\frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|}$?
  - ⇢ different separating directions = different classifiers

Which classifier/direction does gradient descent converge to?

Gradient descent for logistic regression

Theorem: Gradient descent returns the **Euclidean maximum margin classifier** aka hard margin support vector machine

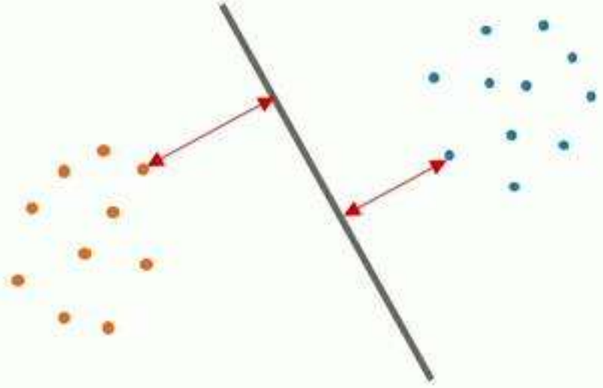


Gradient descent for logistic regression

Theorem: Gradient descent returns the
Euclidean maximum margin classifier aka
hard margin support vector machine

$$\underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{w}\|_2^2 \quad \text{s.t.} \quad y\langle \mathbf{w}, \mathbf{x} \rangle \geq 1 \quad \forall\, (\mathbf{x}, y) \text{ in } D$$

Independent of step size η and initialization $\mathbf{w}(0)$

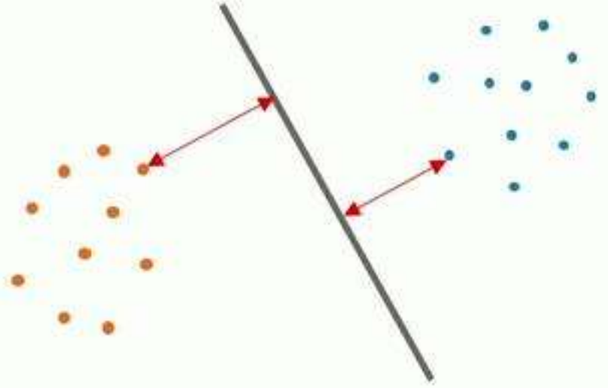


Gradient descent for logistic regression

Theorem: Gradient descent returns the **Euclidean maximum margin classifier** aka hard margin support vector machine

$$\underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{w}\|_2^2 \quad \text{s.t.} \quad y\langle \mathbf{w}, \mathbf{x} \rangle \geq 1 \quad \forall (\mathbf{x}, y) \text{ in } D$$

Independent of step size η and initialization $\mathbf{w}(0)$



Previously: ℓ_1 maximum margin from boosting (coordinate descent) Telgarsky 2013

Follow up: Ji & Telgarsky 2018 **prove generalization guarantees and convergence rates of (S)GD with non-separable data**

Optimization bias with exp-tail losses

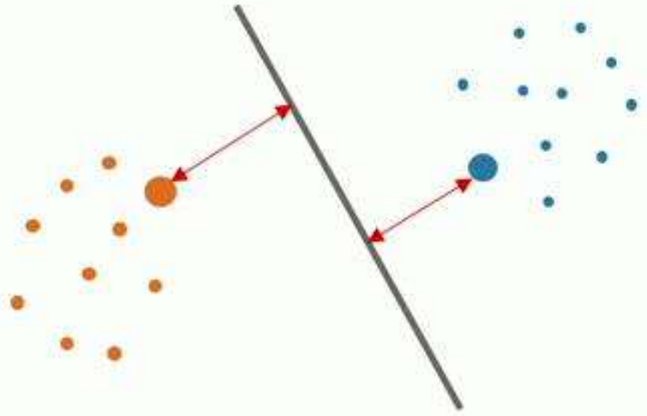
$$-\nabla L(\mathbf{w}(t)) \approx \sum_{n=1}^{N} \exp(-y_n \langle \mathbf{w}(t), \mathbf{x}_n \rangle)\, y_n \mathbf{x}_n$$

Soudry, Hoeffer, Schpigel, Gunasekar, Srebro, (JMLR 2018)

Optimization bias with exp-tail losses

$$-\nabla L(\mathbf{w}(t)) \approx \sum_{n=1}^{N} \boxed{\exp(-y_n \langle \mathbf{w}(t), \mathbf{x}_n \rangle)} \, y_n \mathbf{x}_n$$

large for data points closest to the separator – "support vectors"

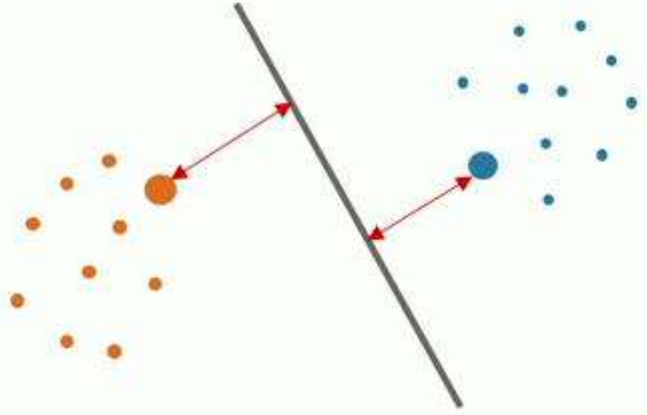


Soudry, Hoeffer, Schpigel, Gunasekar, Srebro, (JMLR 2018)

Optimization bias with exp-tail losses

$$-\nabla L(\mathbf{w}(t)) \approx \sum_{n=1}^{N} \boxed{\exp(-y_n \langle \mathbf{w}(t), \mathbf{x}_n \rangle)}\, y_n \mathbf{x}_n$$

large for data points closest to the separator – "support vectors"



$$\mathbf{w}(t) = \mathbf{w}(0) - \textstyle\sum_{t'<t} \nabla L(\mathbf{w}(t'))$$

dominated by positive span of support vectors
→ sufficient condition for Euclidean max-margin separator

Soudry, Hoeffer, Schpigel, Gunasekar, Srebro, (JMLR 2018)

Optimization bias with exponential loss

Steepest descent w.r.t. general norms

$$\mathbf{w}(t+1) = \underset{\mathbf{w}}{\text{argmin}} \ \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|^2$$

Theorem: steepest descent returns maximum margin solution w.r.t. norm of optimization geometry

$$\underset{\mathbf{w}}{\text{argmin}} \ \|\mathbf{w}\| \quad \text{s.t.} \quad \forall n, y_n \langle \mathbf{w}, \mathbf{x}_n \rangle \geq 1$$

Generalization of analysis in **Telgarsky (2013)**

Gunasekar, Lee, Soudry, Srebro (ICML 2018)

Optimization bias with exponential loss

Steepest descent w.r.t. general norms

$$\mathbf{w}(t+1) = \underset{\mathbf{w}}{\text{argmin}}\ \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|^2$$

Theorem: steepest descent returns maximum margin solution w.r.t. norm of optimization geometry

$$\underset{\mathbf{w}}{\text{argmin}} \|\mathbf{w}\| \quad \text{s.t.} \quad \forall n, y_n \langle \mathbf{w}, \mathbf{x}_n \rangle \geq 1$$

Generalization of analysis in **Telgarsky (2013)**

Mirror descent w.r.t. general potentials

$$\mathbf{w}(t+1) = \underset{\mathbf{w}}{\text{argmin}}\ \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{\eta_t} D_\psi(\mathbf{w}, \mathbf{w}(t))$$

$$\frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|} \overset{?}{\propto} \underset{\mathbf{w}}{\text{argmin}}\ \psi(\mathbf{w}) \quad \text{s.t.} \quad \forall n, y_n \langle \mathbf{w}, \mathbf{x}_n \rangle \geq 1$$

Gunasekar, Lee, Soudry, Srebro (ICML 2018)

Optimization bias with exponential loss

Steepest descent w.r.t. general norms

$$\mathbf{w}(t+1) = \operatorname*{argmin}_{\mathbf{w}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|^2$$

Theorem: steepest descent returns maximum margin solution w.r.t. norm of optimization geometry

$$\operatorname*{argmin}_{\mathbf{w}} \|\mathbf{w}\| \quad \text{s.t.} \quad \forall n, y_n \langle \mathbf{w}, \mathbf{x}_n \rangle \geq 1$$

Generalization of analysis in **Telgarsky (2013)**

Mirror descent w.r.t. general potentials

$$\mathbf{w}(t+1) = \operatorname*{argmin}_{\mathbf{w}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{\eta_t} D_\psi(\mathbf{w}, \mathbf{w}(t))$$

$$\frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|} \overset{?}{\propto} \operatorname*{argmin}_{\mathbf{w}} \psi(\mathbf{w}) \quad \text{s.t.} \quad \forall n, y_n \langle \mathbf{w}, \mathbf{x}_n \rangle \geq 1$$

For non-homogeneous ψ, margin of 1 is not special

Gunasekar, Lee, Soudry, Srebro (ICML 2018)

Optimization bias with exponential loss

Steepest descent w.r.t. general norms

$$\mathbf{w}(t+1) = \operatorname*{argmin}_{\mathbf{w}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}(t)\|^2$$

Theorem: steepest descent returns maximum margin solution w.r.t. norm of optimization geometry

$$\operatorname*{argmin}_{\mathbf{w}} \|\mathbf{w}\| \quad \text{s.t.} \quad \forall n, y_n \langle \mathbf{w}, \mathbf{x}_n \rangle \geq 1$$

Generalization of analysis in **Telgarsky (2013)**

Mirror descent w.r.t. general potentials

$$\mathbf{w}(t+1) = \operatorname*{argmin}_{\mathbf{w}} \langle \mathbf{w}, \nabla L(\mathbf{w}(t)) \rangle + \frac{1}{\eta_t} D_\psi(\mathbf{w}, \mathbf{w}(t))$$

$$\frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|} \overset{?}{\propto} \operatorname*{argmin}_{\mathbf{w}} \psi(\mathbf{w}) \quad \text{s.t.} \quad \boxed{\forall n, y_n \langle \mathbf{w}, \mathbf{x}_n \rangle \geq 1}$$

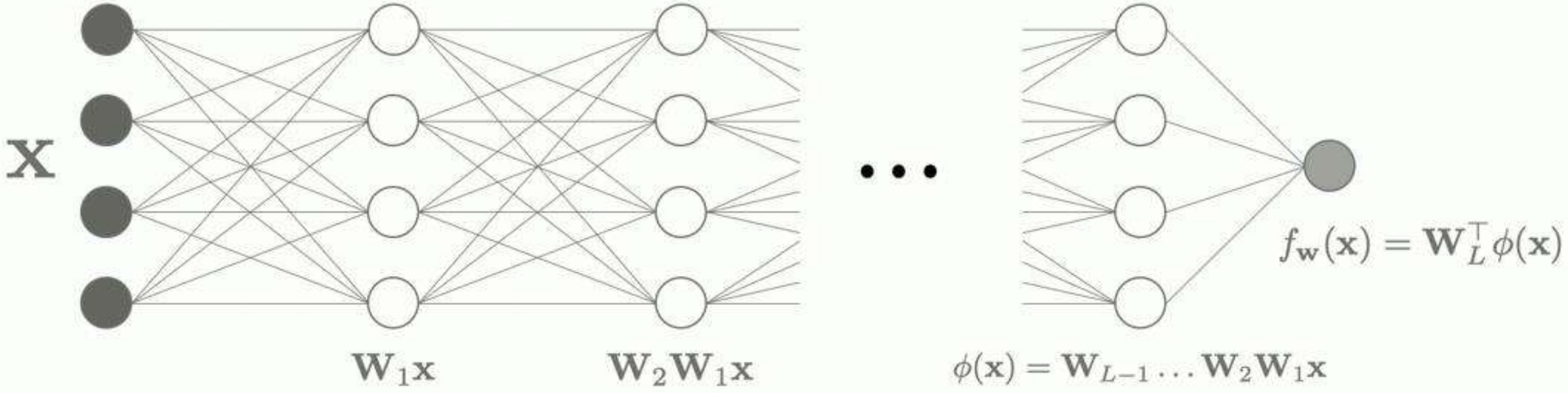
For non-homogeneous ψ, margin of 1 is not special

For some potentials, MD converges to limit of margin path

$$\lim_{\gamma \to \infty} \frac{\operatorname*{argmin}_{\mathbf{w}} \psi(\mathbf{w}) \quad \text{s.t.} \quad \forall n, y_n \langle \mathbf{w}, \mathbf{x}_n \rangle \geq \gamma}{\gamma}$$

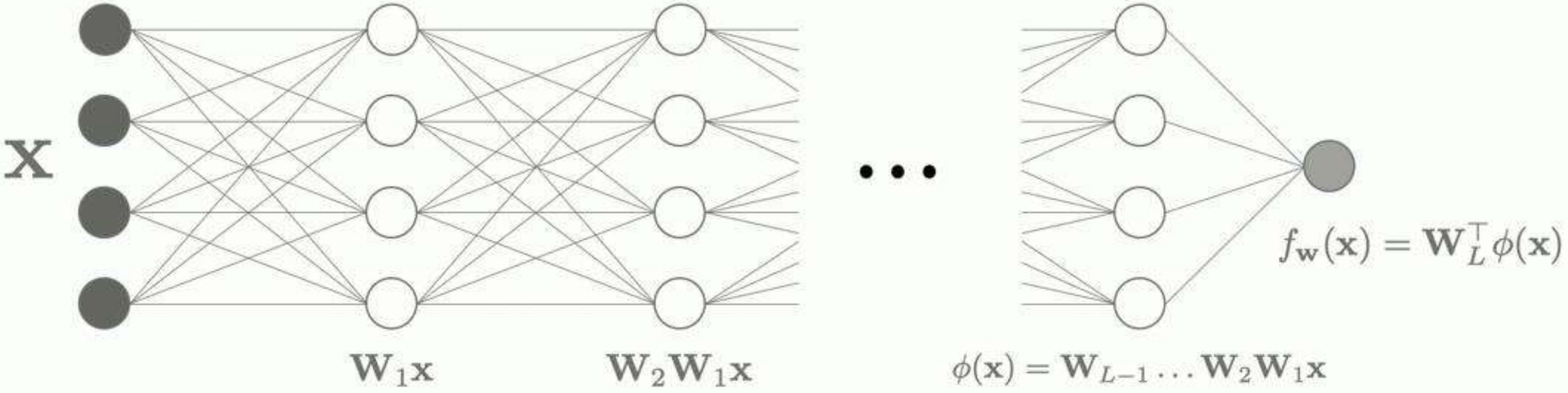
Gunasekar, Lee, Soudry, Srebro (ICML 2018)

Multilayer linear network



Gunasekar, Lee, Soudry, Srebro (NeurIPS 2018)

Multilayer linear network

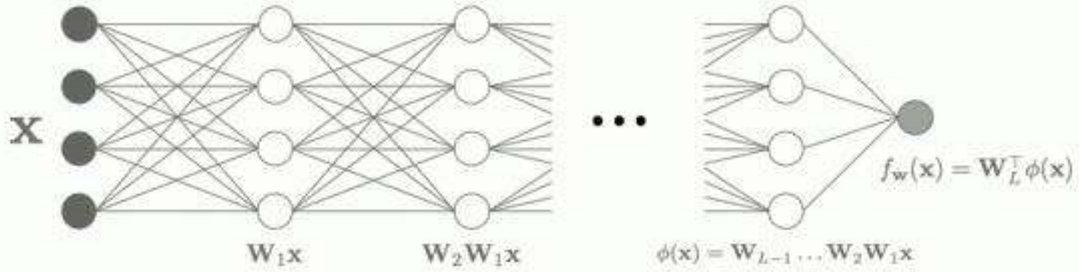


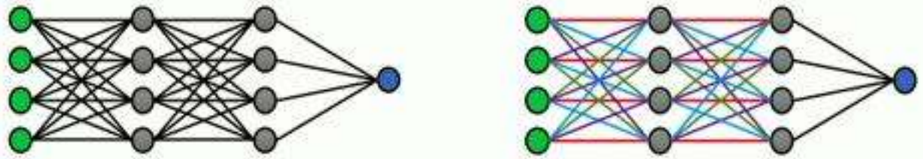
different constraints on $\mathbf{w} = [\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_{L+1}]$ $\longrightarrow$ different architectures

e.g., fully connected networks,

convolutional networks

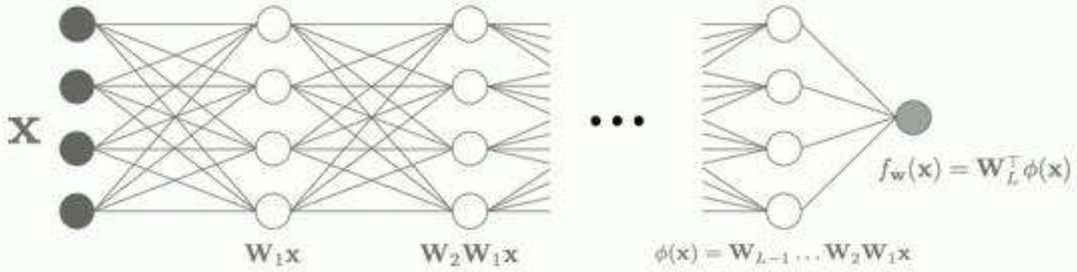
Gunasekar, Lee, Soudry, Srebro (NeurIPS 2018)

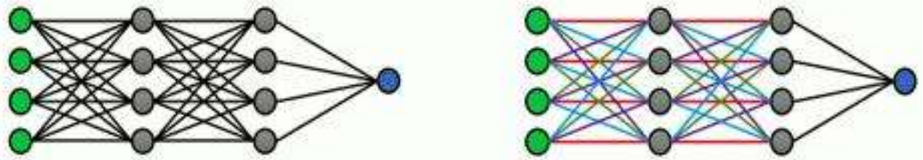




$$\min_{\mathbf{w}} \; L(\mathbf{w}) \; := \sum_{(\mathbf{x},y) \text{ in } D} \exp\big(-y f_{\mathbf{w}}(\mathbf{x})\big)$$

Gunasekar, Lee, Soudry, Srebro (NeurIPS 2018)

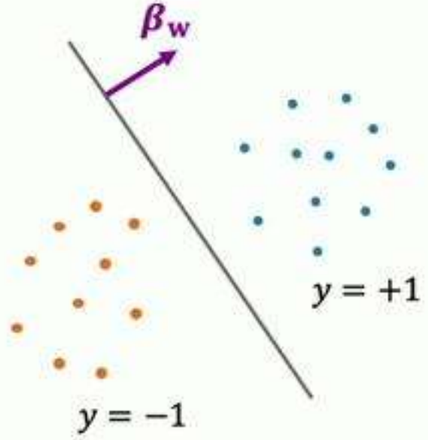




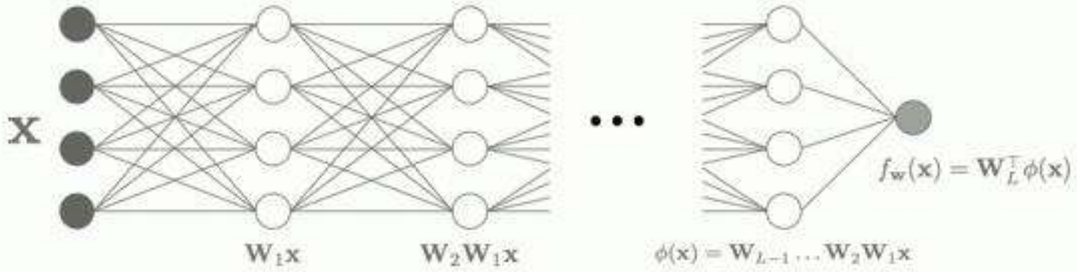
$$\min_{\mathbf{w}} \; L(\mathbf{w}) \; := \sum_{(\mathbf{x},y) \text{ in } D} \exp\big(-y f_{\mathbf{w}}(\mathbf{x})\big)$$

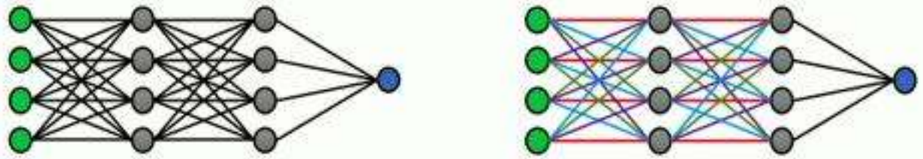
all networks implement only linear classifiers

$\mathbf{w} \to \boldsymbol{\beta}_{\mathbf{w}}$ such that $f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{x}, \boldsymbol{\beta}_{\mathbf{w}} \rangle$



Gunasekar, Lee, Soudry, Srebro (NeurIPS 2018)

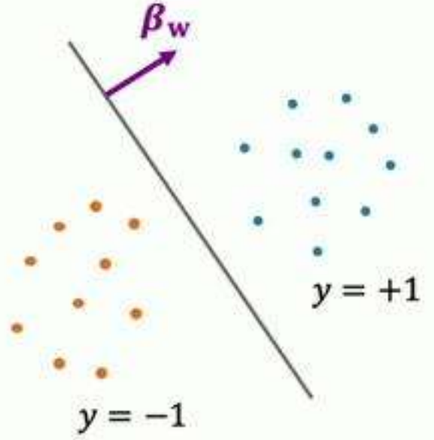




$$\min_{\mathbf{w}} \; L(\mathbf{w}) \; := \sum_{(\mathbf{x},y) \text{ in } D} \exp\big(-y f_{\mathbf{w}}(\mathbf{x})\big)$$

all networks implement only linear classifiers

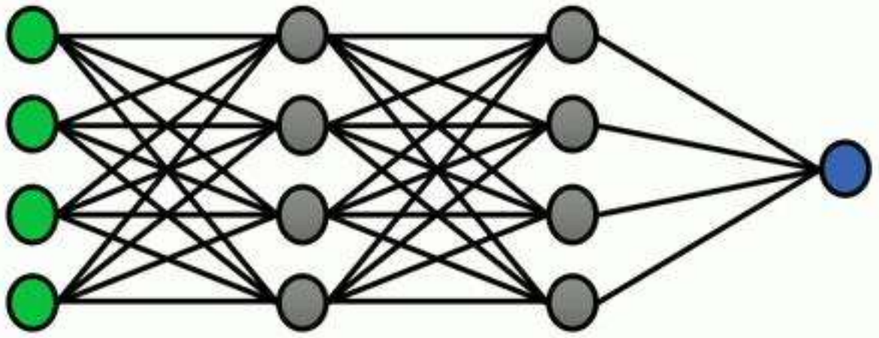
$\mathbf{w} \to \boldsymbol{\beta}_{\mathbf{w}}$ such that $f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{x}, \boldsymbol{\beta}_{\mathbf{w}} \rangle$



What classifier does gradient descent on $L(\mathbf{w})$ return for different architectures?

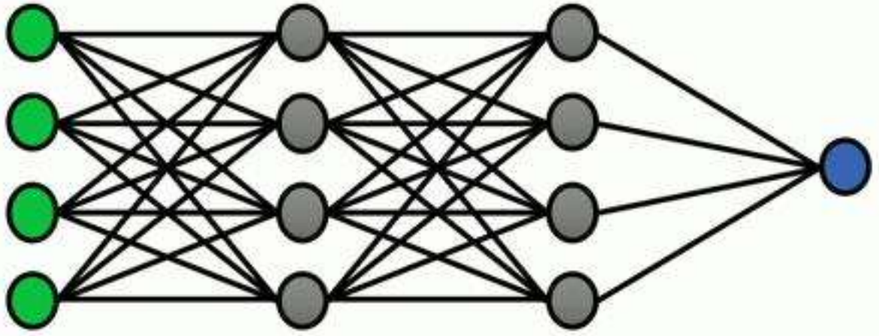
Gunasekar, Lee, Soudry, Srebro (NeurIPS 2018)

Linear fully connected networks



Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

Linear fully connected networks

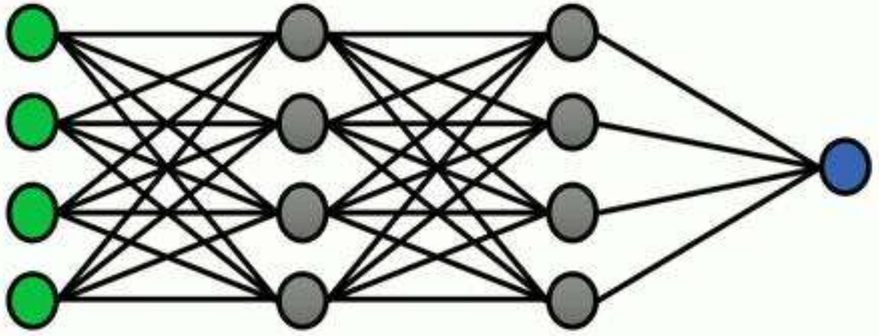


Theorem*: for any depth (#layers), gradient descent again returns the **Euclidean max-margin** classifier

$$\underset{\beta}{\text{argmin}} \|\beta\|_2^2 \quad \text{s.t.} \quad y\langle \beta, \mathbf{x} \rangle \geq 1 \quad \forall\, (\mathbf{x}, y) \text{ in } D$$

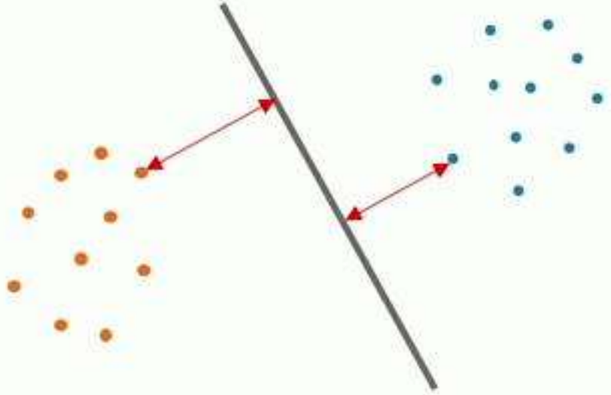
Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

Linear fully connected networks



Theorem*: for any depth (#layers), gradient descent again returns the **Euclidean max-margin** classifier

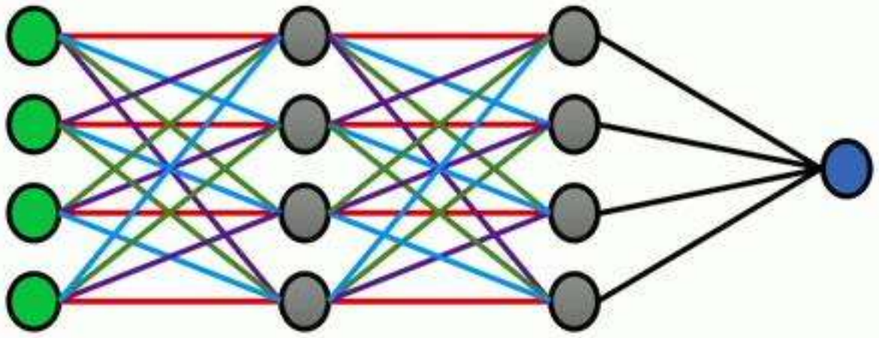
$$\underset{\beta}{\text{argmin}} \|\beta\|_2^2 \quad \text{s.t.} \quad y\langle \beta, \mathbf{x} \rangle \geq 1 \quad \forall\, (\mathbf{x}, y) \text{ in } D$$



same solution as logistic regression!

Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

Linear convolutional network



Theorem*: gradient descent implicitly introduces biases that

1. promote sparsity in the frequency domain (Fourier coefficients) → frequency filtering
2. changes with the depth/#layers of network

2 layers

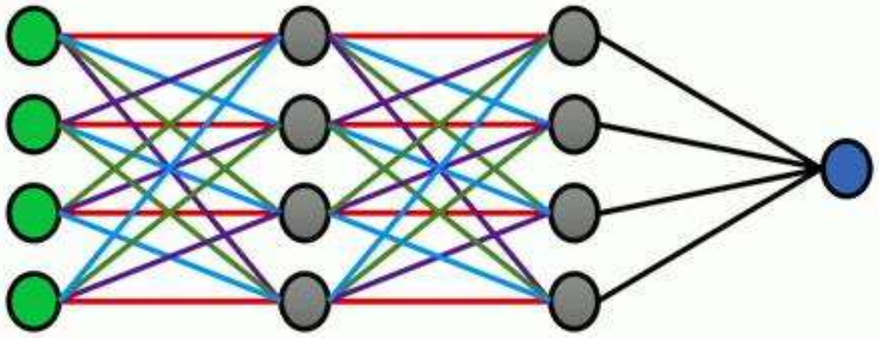
$$\underset{\beta}{\text{argmin}} \; \|\text{DFT}(\beta)\|_1 \quad \text{s.t.} \quad y\langle \beta, \mathbf{x} \rangle \geq 1 \quad \forall\, (\mathbf{x}, y) \text{ in } D$$

Fourier coefficients/ frequency components

minimizing ℓ_1 norm is known to encourage sparse solutions

Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

Linear convolutional network



Theorem*: gradient descent implicitly introduces biases that

1. promote sparsity in the frequency domain (Fourier coefficients) → frequency filtering
2. changes with the depth/#layers of network

2 layers

$$\underset{\beta}{\text{argmin}} \|\text{DFT}(\beta)\|_1 \quad \text{s.t.} \quad y\langle \beta, \mathbf{x}\rangle \geq 1 \quad \forall\, (\mathbf{x}, y) \text{ in } D$$

Fourier coefficients/ frequency components

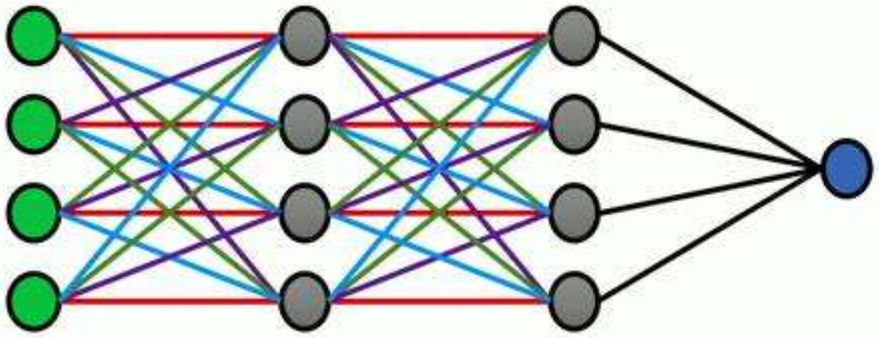
minimizing ℓ_1 norm is known to encourage sparse solutions

L convolutional layers

stationary points of $\underset{\beta}{\text{argmin}} \|\text{DFT}(\beta)\|_{2/L} \quad \text{s.t.} \quad y\langle \beta, \mathbf{x}\rangle \geq 1 \quad \forall\, (\mathbf{x}, y) \text{ in } D$

Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

Linear convolutional network



Theorem*: gradient descent implicitly introduces biases that

1. promote sparsity in the frequency domain (Fourier coefficients) → frequency filtering
2. changes with the depth/#layers of network

2 layers

$$\underset{\beta}{\text{argmin}} \|\text{DFT}(\beta)\|_1 \quad \text{s.t.} \quad y\langle \beta, \mathbf{x} \rangle \geq 1 \quad \forall\, (\mathbf{x}, y) \text{ in } D$$

Fourier coefficients/ frequency components

minimizing ℓ_1 norm is known to encourage sparse solutions

Bias not simply because of convolutional architecture

but from gradient descent on convolutional parameterization

L convolutional layers

stationary points of $\underset{\beta}{\text{argmin}} \|\text{DFT}(\beta)\|_{2/L}$ s.t. $y\langle \beta, \mathbf{x} \rangle \geq 1 \quad \forall\, (\mathbf{x}, y) \text{ in } D$

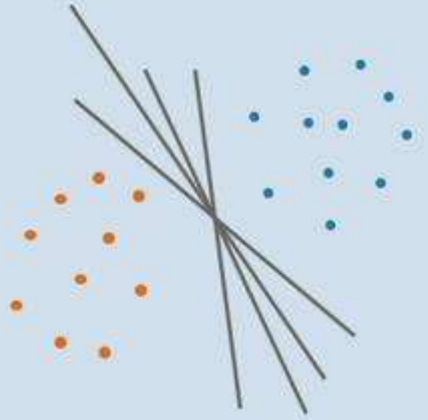
Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

Theorem details and assumptions

Gradient descent on exp-loss for separable datasets

- Many directions/classifiers separate the data

  ⇢ **which classifier does gradient descent return? ← our work!**



$$\min_{\mathbf{w}} L(\mathbf{w}) = \sum_{(\mathbf{x},y)\text{ in } D} \exp\left(-y f_{\mathbf{w}}(\mathbf{x})\right)$$

Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

Theorem details and assumptions

Gradient descent on exp-loss for separable datasets

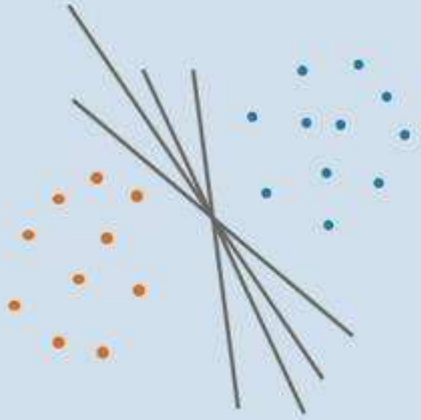
- Many directions/classifiers separate the data

  ⇢ **which classifier does gradient descent return? ← our work!**

Related questions

- Optimization objective is non-convex

  ⇢ does gradient descent globally minimize the objective?



$$\min_{\mathbf{w}} L(\mathbf{w}) = \sum_{(\mathbf{x},y) \text{ in } D} \exp\left(-y f_{\mathbf{w}}(\mathbf{x})\right)$$

Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

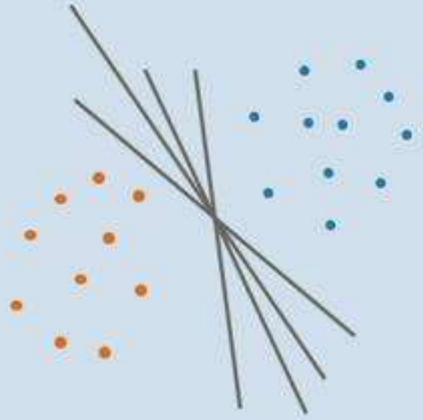
Theorem details and assumptions

Gradient descent on exp-loss for separable datasets

- Many directions/classifiers separate the data
  - ⇢ **which classifier does gradient descent return? ← our work!**

Related questions

- Optimization objective is non-convex
  - ⇢ does gradient descent globally minimize the objective?
- No finite global minimizers for exponential loss
  - ⇢ optimization iterates will diverge in norm
  - ⇢ does the direction of iterates (decision boundary) converge?



$$\min_{\mathbf{w}} L(\mathbf{w}) = \sum_{(\mathbf{x},y) \text{ in } D} \exp\left(-y f_{\mathbf{w}}(\mathbf{x})\right)$$

Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

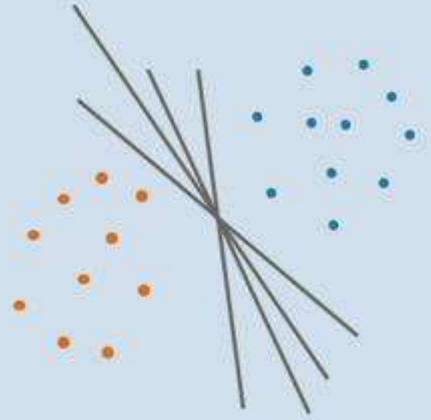
Theorem details and assumptions

Gradient descent on exp-loss for separable datasets

- Many directions/classifiers separate the data
  - ⇢ **which classifier does gradient descent return? ← our work!**

Related questions

- Optimization objective is non-convex
  - ⇢ does gradient descent globally minimize the objective?
- No finite global minimizers for exponential loss
  - ⇢ optimization iterates will diverge in norm
  - ⇢ does the direction of iterates (decision boundary) converge?



$$\min_{\mathbf{w}} L(\mathbf{w}) = \sum_{(\mathbf{x},y) \text{ in } D} \exp\left(-y f_{\mathbf{w}}(\mathbf{x})\right)$$

$L(\mathbf{w}(t)) \longrightarrow 0$

$\frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|}$ converges to a separator

$\frac{\nabla L(\mathbf{w}(t))}{\|\nabla L(\mathbf{w}(t))\|}$ converges

Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

Theorem details and assumptions

Gradient descent on exp-loss for separable datasets

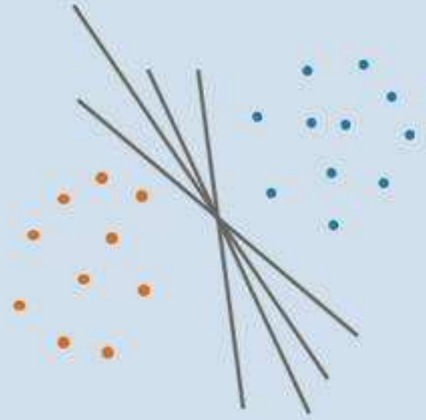
- Many directions/classifiers separate the data
  - ⇢ **which classifier does gradient descent return? ← our work!**

Related questions

- Optimization objective is non-convex
  - ⇢ does gradient descent globally minimize the objective?
- No finite global minimizers for exponential loss
  - ⇢ optimization iterates will diverge in norm
  - ⇢ does the direction of iterates (decision boundary) converge?

Follow up work:

- **Ji & Telgarsky 2018 – prove the result for linear fully connected networks without assumptions on convergence of loss and update direction**



$$\min_{\mathbf{w}} L(\mathbf{w}) = \sum_{(\mathbf{x},y) \text{ in } D} \exp\left(-y f_{\mathbf{w}}(\mathbf{x})\right)$$

$$L(\mathbf{w}(t)) \longrightarrow 0$$

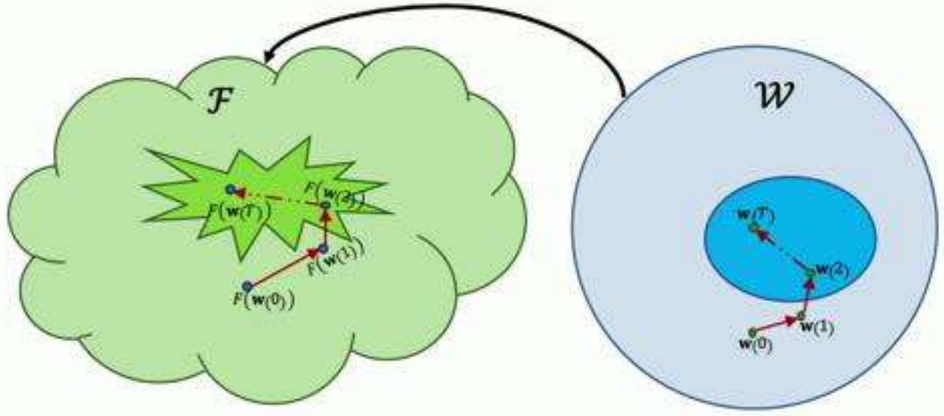
$\frac{\mathbf{w}(t)}{\|\mathbf{w}(t)\|}$ converges to a separator

$\frac{\nabla L(\mathbf{w}(t))}{\|\nabla L(\mathbf{w}(t))\|}$ converges

Homogeneous linear models

$$f_{\mathbf{w}}(\mathbf{x}) = \langle \mathcal{P}(\mathbf{w}), \mathbf{x} \rangle$$

$\mathcal{P}$ is a homogeneous function $\mathcal{P}(\alpha \mathbf{w}) = \alpha^{\nu} \mathcal{P}(\mathbf{w})$



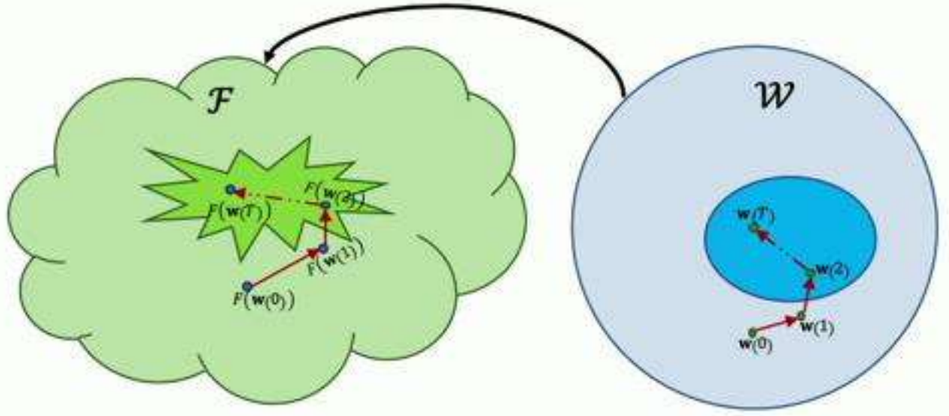
Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

Homogeneous linear models

$f_{\mathbf{w}}(\mathbf{x}) = \langle \mathcal{P}(\mathbf{w}), \mathbf{x} \rangle$ $\mathcal{P}$ is a homogeneous function $\mathcal{P}(\alpha \mathbf{w}) = \alpha^{\nu} \mathcal{P}(\mathbf{w})$

Theorem*: gradient descent implicitly **minimizes Euclidean norm of parameters**

stationary points of $\underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{w}\|_2 \quad \text{s.t.} \quad y\langle \mathcal{P}(\mathbf{w}), \mathbf{x} \rangle \geq 1 \quad \forall (\mathbf{x}, y) \text{ in } D$



Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

Homogeneous linear models

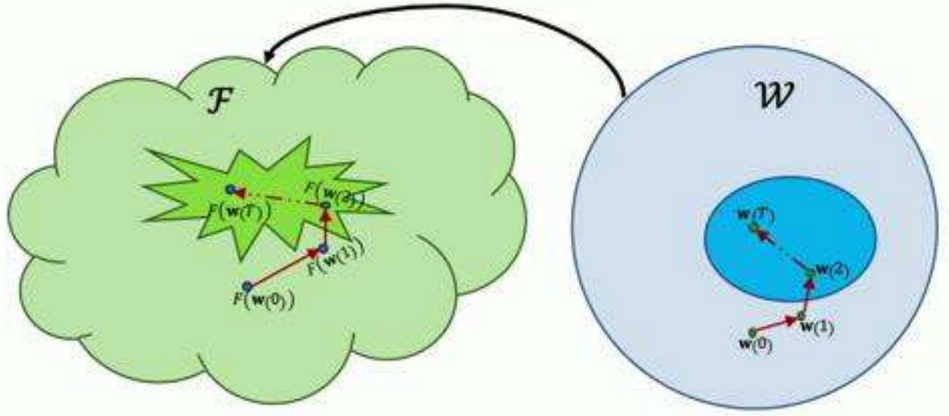
$f_{\mathbf{w}}(\mathbf{x}) = \langle \mathcal{P}(\mathbf{w}), \mathbf{x} \rangle$ $\mathcal{P}$ is a homogeneous function $\mathcal{P}(\alpha \mathbf{w}) = \alpha^{\nu} \mathcal{P}(\mathbf{w})$

Theorem*: gradient descent implicitly **minimizes Euclidean norm of parameters**

stationary points of $\underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{w}\|_2 \quad \text{s.t.} \quad y\langle \mathcal{P}(\mathbf{w}), \mathbf{x} \rangle \geq 1 \quad \forall (\mathbf{x}, y) \text{ in } D$

Induced bias on prediction functions :

$\mathcal{R}_{\mathcal{P}}(\boldsymbol{\beta}) := \inf_{\boldsymbol{\beta} = \mathcal{P}(\mathbf{w})} \|\mathbf{w}\|_2^2$



Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

Homogeneous linear models

$f_{\mathbf{w}}(\mathbf{x}) = \langle \mathcal{P}(\mathbf{w}), \mathbf{x} \rangle$ $\mathcal{P}$ is a homogeneous function $\mathcal{P}(\alpha \mathbf{w}) = \alpha^{\nu} \mathcal{P}(\mathbf{w})$

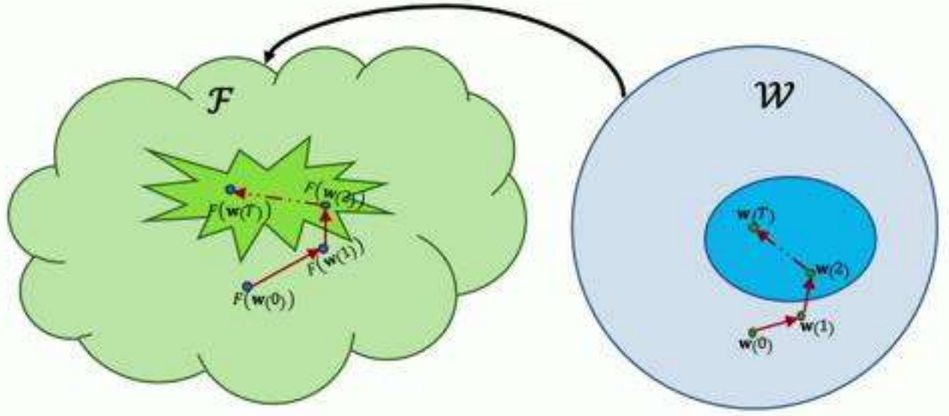
Theorem*: gradient descent implicitly **minimizes Euclidean norm of parameters**

stationary points of $\underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{w}\|_2 \quad \text{s.t.} \quad y\langle \mathcal{P}(\mathbf{w}), \mathbf{x} \rangle \geq 1 \quad \forall (\mathbf{x}, y) \text{ in } D$

Induced bias on prediction functions :

$$\mathcal{R}_{\mathcal{P}}(\boldsymbol{\beta}) := \inf_{\boldsymbol{\beta} = \mathcal{P}(\mathbf{w})} \|\mathbf{w}\|_2^2$$

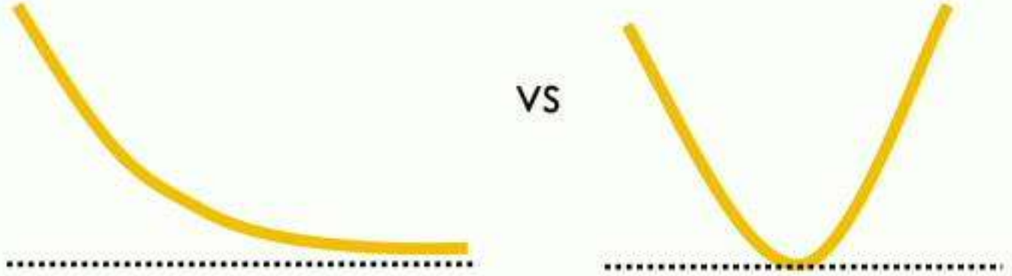
Follow up: [Lyu & LI 2019][Nacson, Gunasekar, Lee, Soudry, Srebro 2019] – generalization to homogeneous models



Gunasekar, Lee, Soudry, Srebro NeurIPS 2018

Optimization bias with exponential loss

Nature of optimization bias is very different from squared loss



Exp-tailed losses on separable data

optimization path has infinite arc length

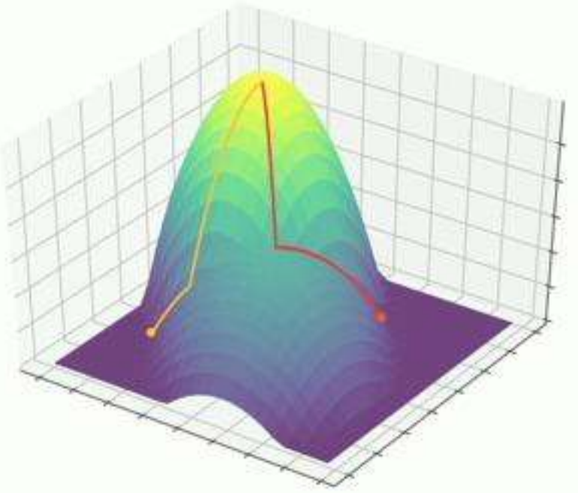
→ max-margin vs min-norm solutions

→ initialization and step size independent characterizations

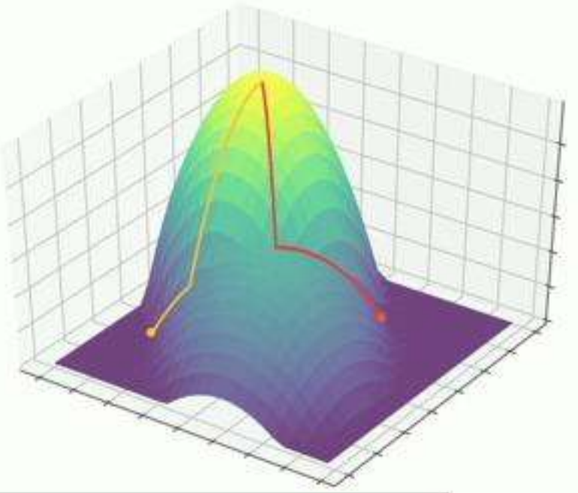
- Interesting non-RKHS biases independent of step-size and initialization (as long as they are finite)
  e.g. max margin w.r.t $\|\mathrm{DFT}(\beta)\|_1$ for convolutional networks
- WIP: reconciling with NTK type analysis

→ simple asymptotic characterizations for many complicated models

Role of optimization in ML extends beyond reaching any global minimum

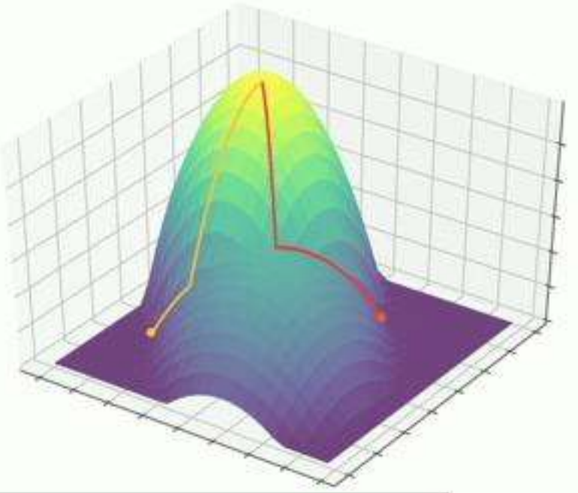


Role of optimization in ML extends
beyond reaching any global minimum



- *geometry of updates* as well as *geometry of parameters* determine the nature of optimization bias
- characterization of bias and its dependence on hyperparameters are very different for regression and classification tasks

Role of optimization in ML extends
beyond reaching any global minimum



- *geometry of updates* as well as *geometry of parameters* determine the nature of optimization bias
- characterization of bias and its dependence on hyperparameters are very different for regression and classification tasks

Thank
you!

A New Perspective on Adversarial Perturbations

Aleksander Mądry

Based on joint works with:

Logan Engstrom

Andrew Ilyas

Shibani Santurkar

Brandon Tran

Dimitris Tsipras

Alexander Turner

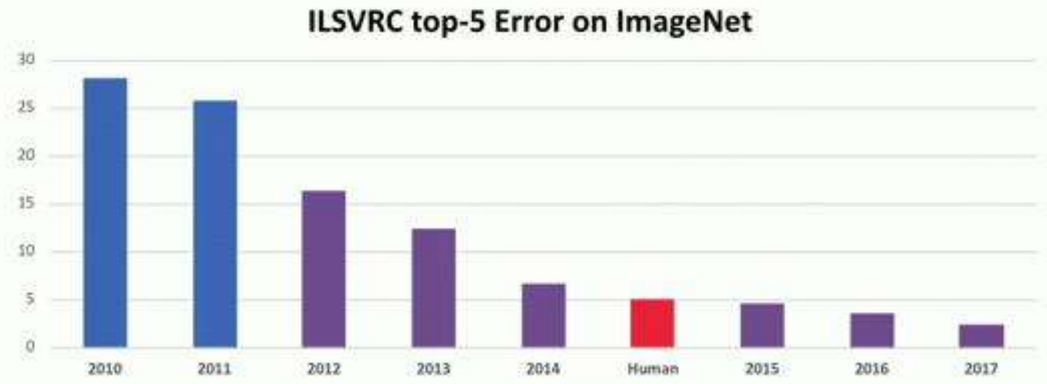
gradientscience.org

Why do we love deep learning?

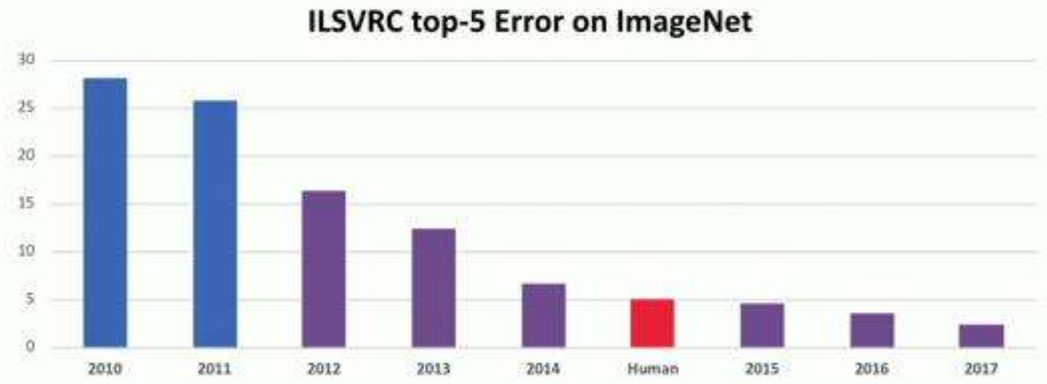
Why do we love deep learning?

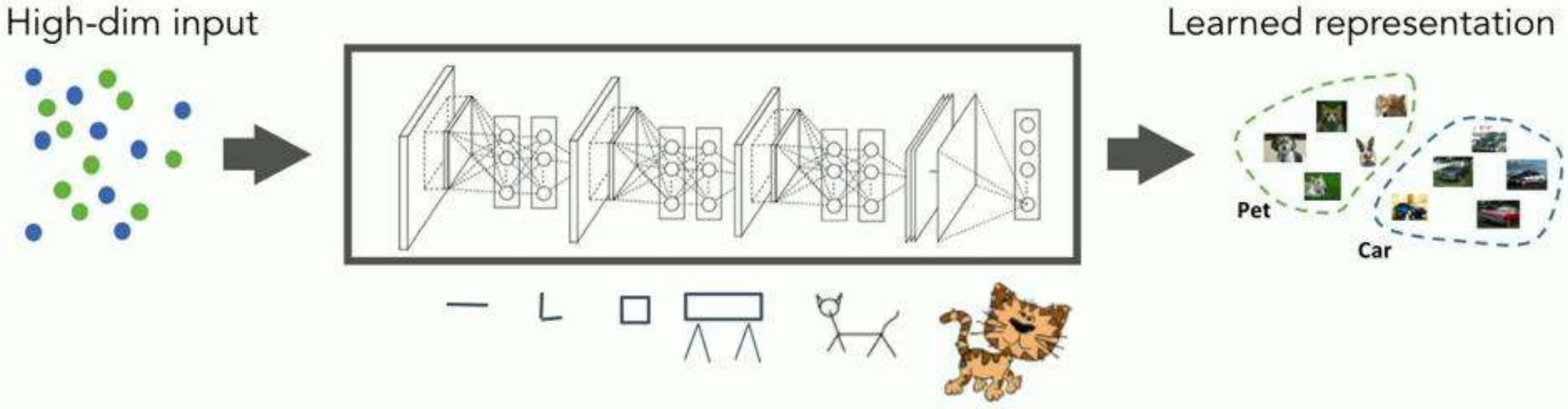
Why do we love deep learning?

Why do we love deep learning?



Why do we love deep learning?





But...

But...

But...

But...

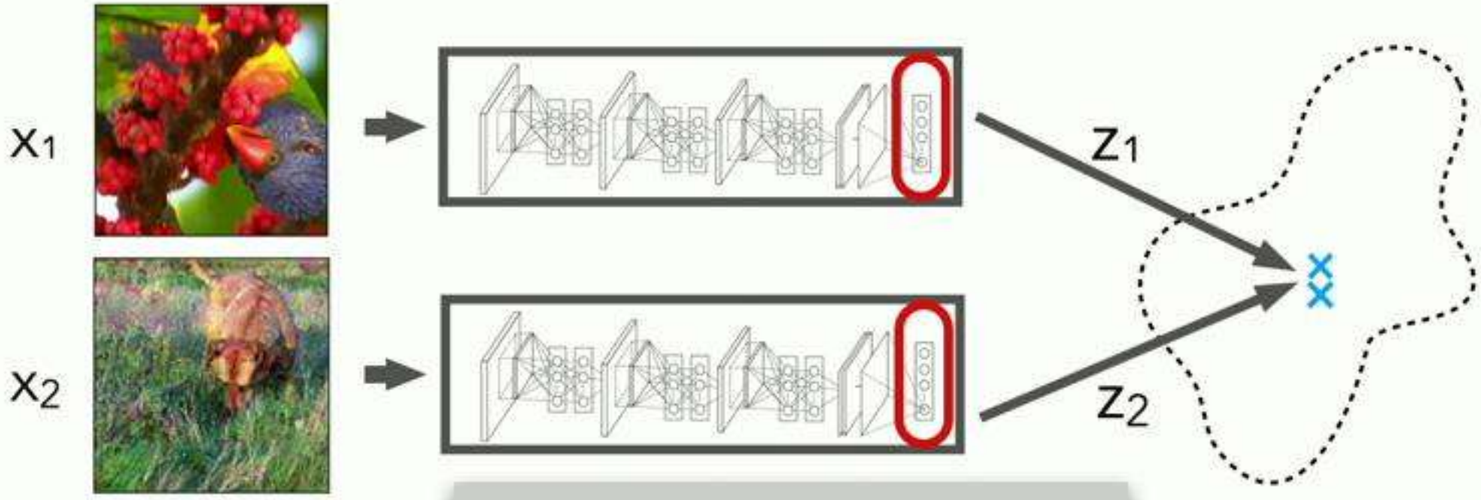
Correct label: insect

But...

Correct label: insect
Predicted label: dog

But...

Correct label: insect
Predicted label: dog



$x_1 \neq x_2$ but $z_1 \approx z_2$

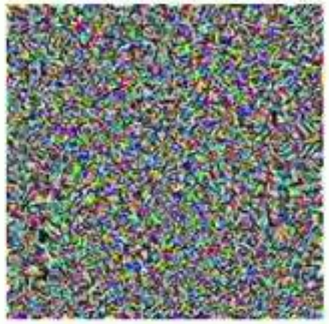
What's going on?

Key Problem: Adversarial Perturbations

[Szegedy et al 2013] [Biggio et al 2013]

“pig” (91%)

+ 0.005 x



noise (NOT random)

=

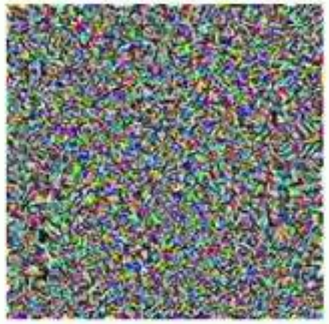
“airliner” (99%)

Key Problem: Adversarial Perturbations

[Szegedy et al 2013] [Biggio et al 2013]

"pig" (91%)

+ 0.005 x



noise (NOT random)

=

"airliner" (99%)

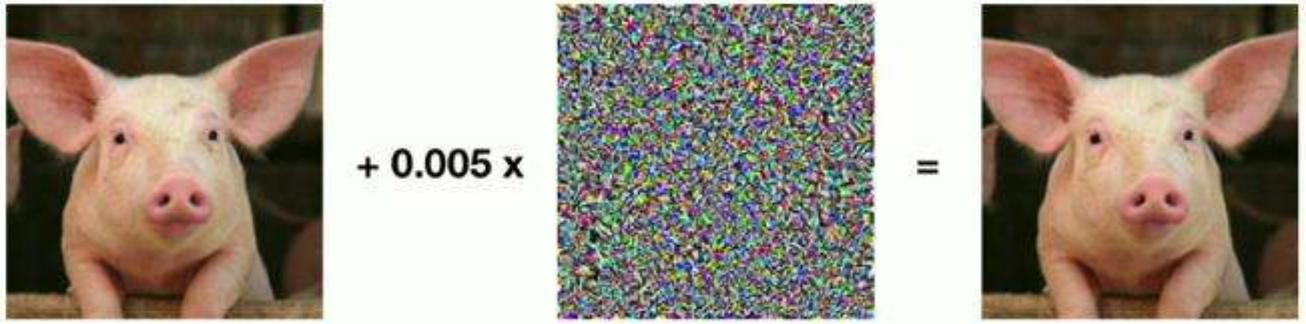
Emerging goal: (Adversarially) robust generalization

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}[\max_{\delta\in\Delta} \ell(\theta; x+\delta, y)]$$

Desired invariance

Key Problem: Adversarial Perturbations

[Szegedy et al 2013] [Biggio et al 2013]



"pig" (91%) **noise (NOT random)** **"airliner" (99%)**

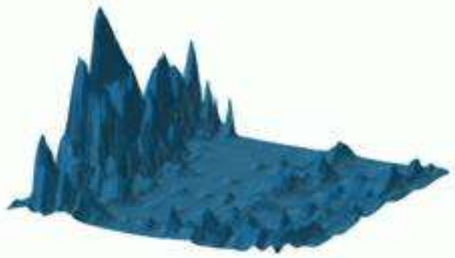
Emerging goal: (Adversarially) robust generalization

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D}[\max_{\delta\in\Delta} \ell(\theta; x+\delta, y)]$$

→ We are (finally) succeeding in achieving this goal

ML via Adversarial Robustness Lens

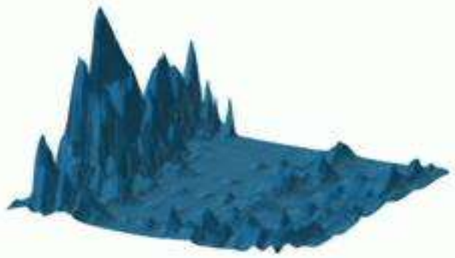
ML via Adversarial Robustness Lens



- Training is harder and models need to be more complex

[M Makelov Schmidt Tsipras Vladu 2018]

ML via Adversarial Robustness Lens



- Training is harder and models need to be more complex

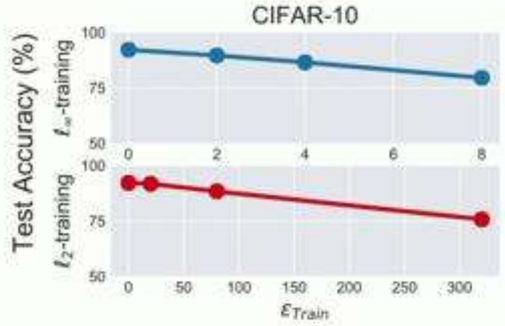
  [M Makelov Schmidt Tsipras Vladu 2018]

- Models may <u>have</u> to be less accurate

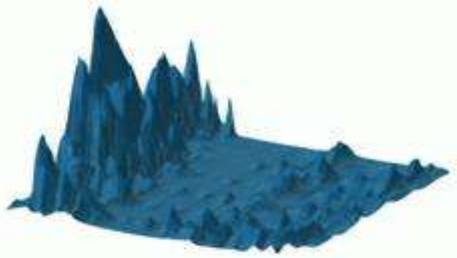
  [Tsipras Santurkar Engstrom Turner M 2018]

  [Bubeck Price Razenshteyn 2018]

  [Degwekar Nakkiran Vaikunatanathan 2018]



ML via Adversarial Robustness Lens



- Training is harder and models need to be more complex

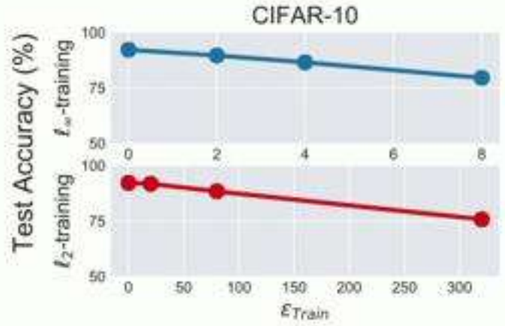
  [M Makelov Schmidt Tsipras Vladu 2018]

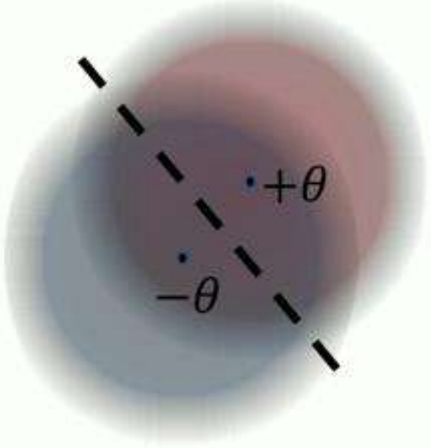
- Models may <u>have</u> to be less accurate

  [Tsipras Santurkar Engstrom Turner M 2018]

  [Bubeck Price Razenshteyn 2018]

  [Degwekar Nakkiran Vaikunatanathan 2018]





- We might need more training data

  [Schmidt Santurkar Tsipras Talwar M 2018]

But: “How”/“what” does not tell us “why”

But: "How"/"what" does not tell us "why"

Why adversarial perturbations **exist**
(and **are so widespread**)?

But: "How"/"what" does not tell us "why"

Why adversarial perturbations **exist**
(and **are so widespread**)?

Why these perturbations tend to **transfer**?

But: "How"/"what" does not tell us "why"

Why adversarial perturbations **exist**
(and **are so widespread**)?

Why these perturbations tend to **transfer**?

Why **robust training** works?

But: "How"/"what" does not tell us "why"

Why adversarial perturbations **exist**
(and **are so widespread**)?

Why these perturbations tend to **transfer**?

Why **robust training** works?

Why **randomized smoothing** works?

But: "How"/"what" does not tell us "why"

Why adversarial perturbations **exist** (and **are so widespread**)?

Why these perturbations tend to **transfer**?

Why **robust training** works?

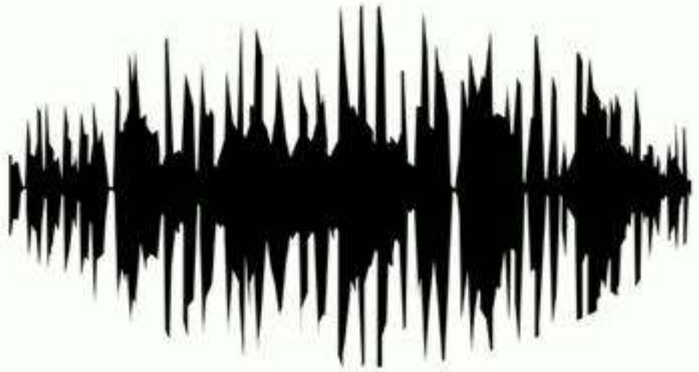
Why **randomized smoothing** works?

Why are our models brittle?

$$d \to \infty$$

Why are our models brittle?

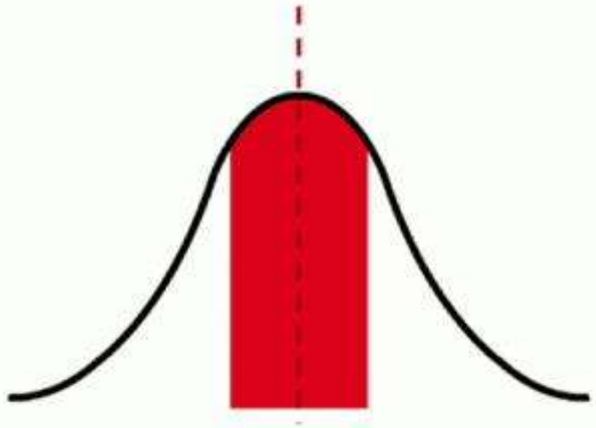
$d \to \infty$



Why are our models brittle?

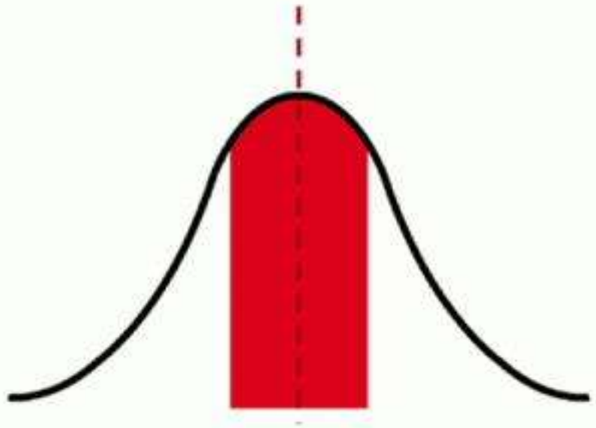
$d \to \infty$

Why are our models brittle?



$$d \to \infty$$

Why are our models brittle?



$d \to \infty$

Why are our models brittle?

Unifying theme: Adversarial examples are aberrations

Why Are Adv. Perturbations Bad?

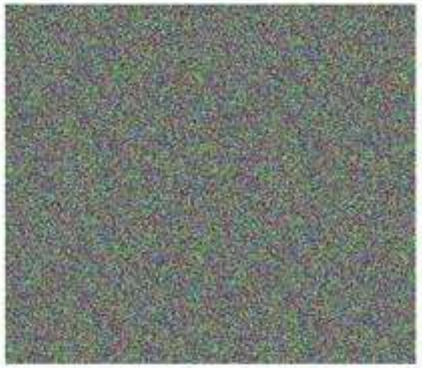
Why Are Adv. Perturbations Bad?

dog

Why Are Adv. Perturbations Bad?

dog

+

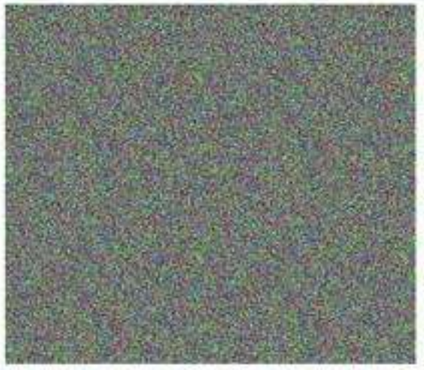


meaningless
perturbation

Why Are Adv. Perturbations Bad?

dog

+



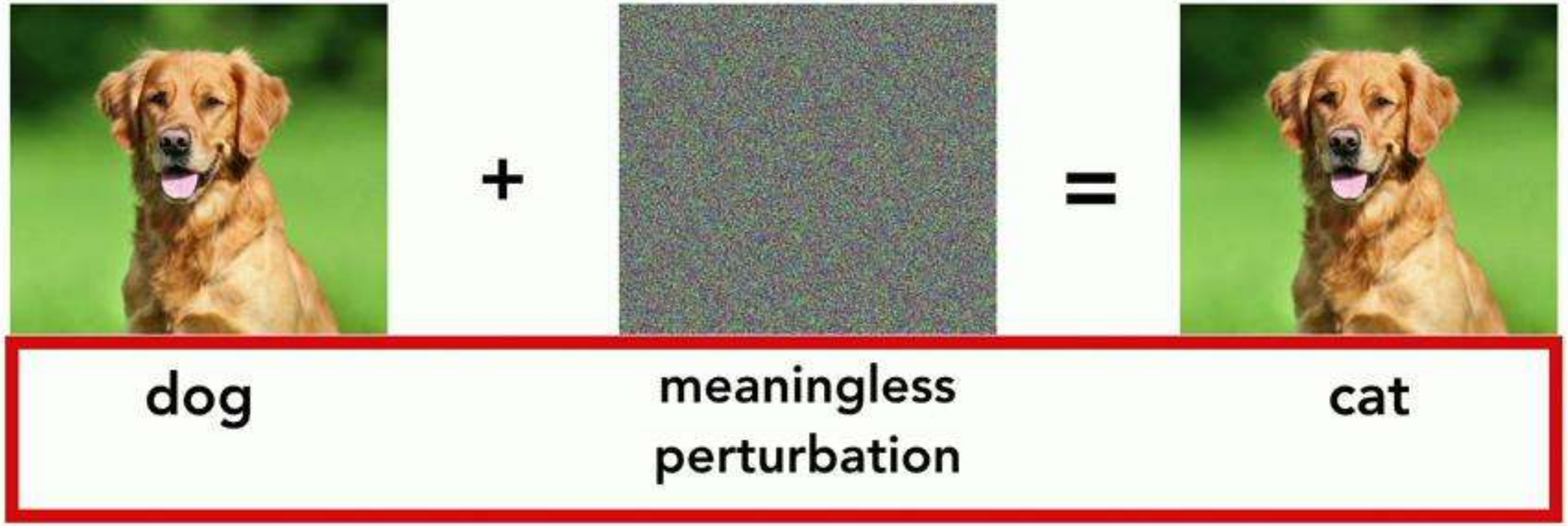
meaningless
perturbation

=

cat

Why Are Adv. Perturbations Bad?

Why Are Adv. Perturbations Bad?



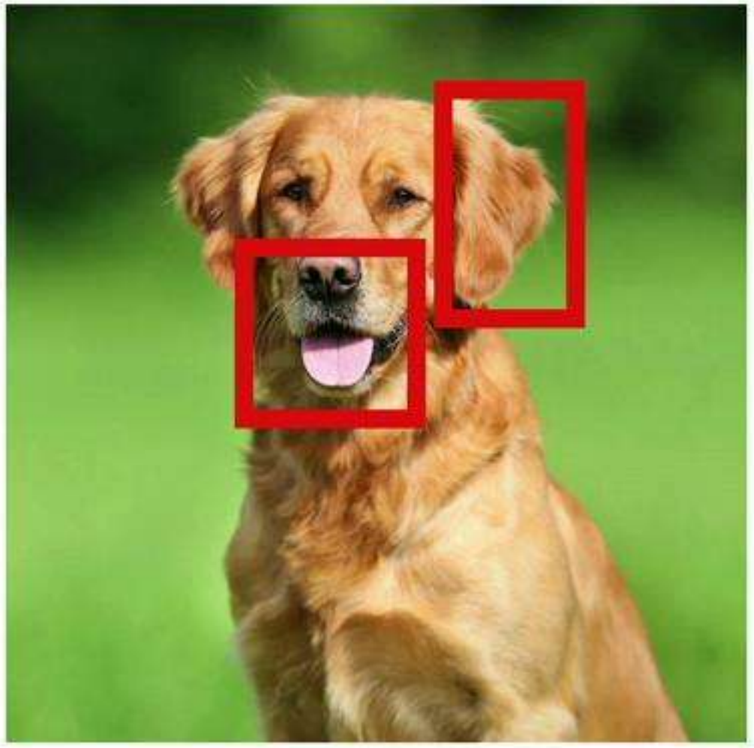
But: This is only a "human" perspective

Human Perspective

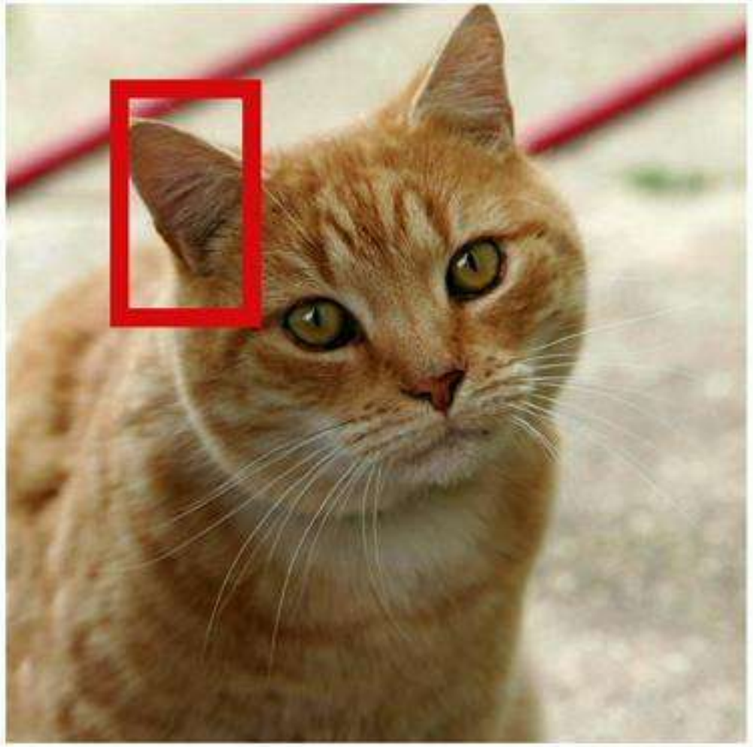
dog

cat

Human Perspective



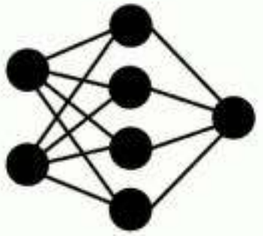
dog



cat

ML Perspective

ML Perspective



dog

ML Perspective

dog

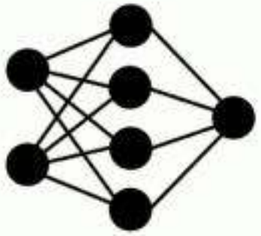


Image is meaningless

ML Perspective

dog

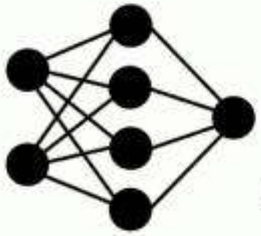


Image is meaningless

Classes are meaningless

ML Perspective

dog

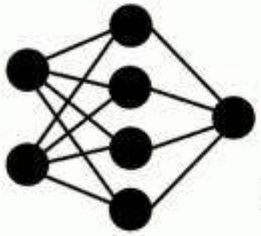


Image is meaningless

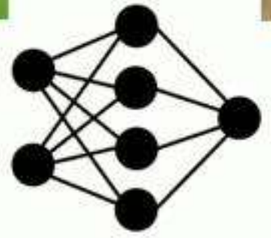
Classes are meaningless

Only goal:

Max (test) accuracy

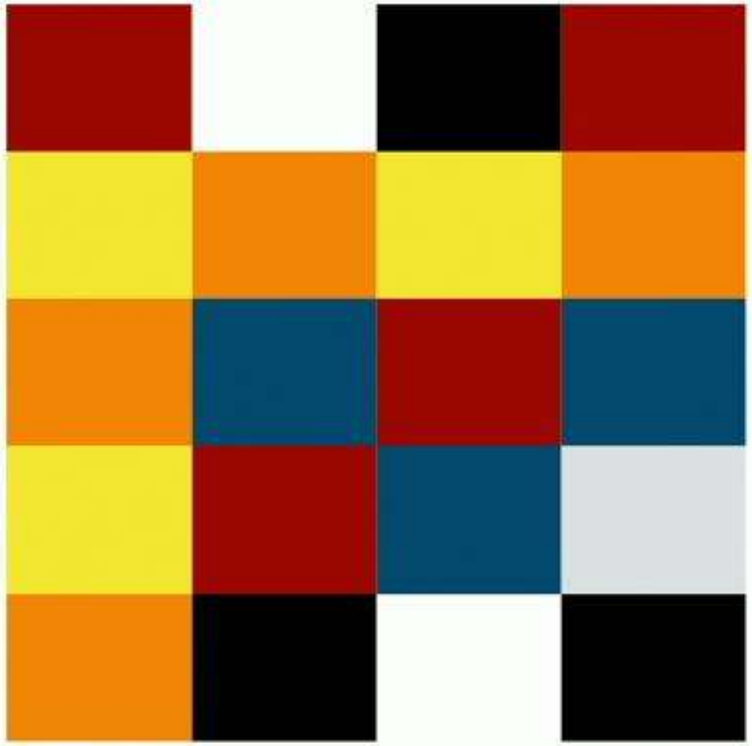
ML Perspective

dog

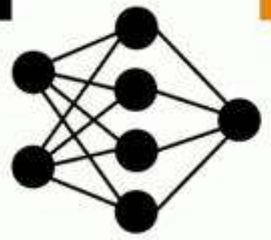


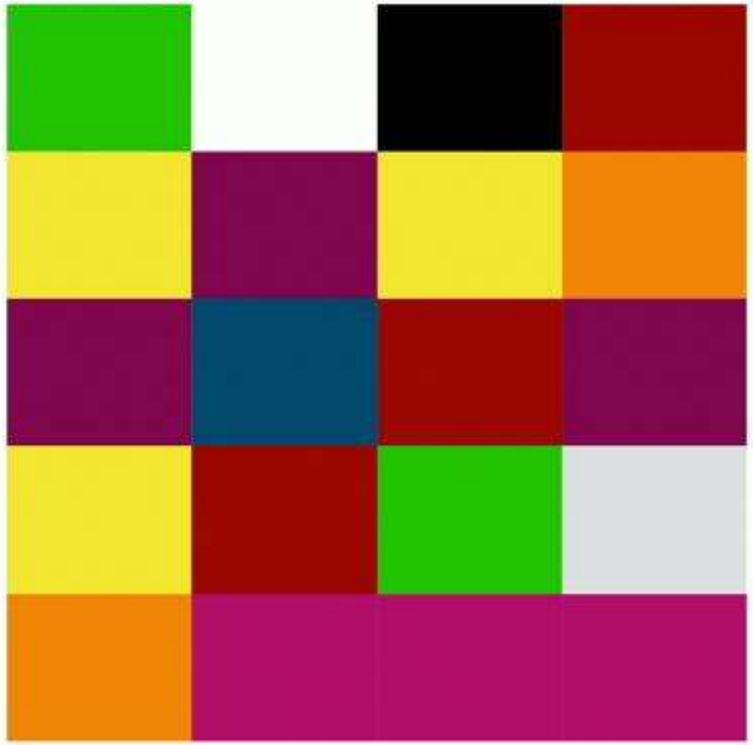
cat

ML Perspective



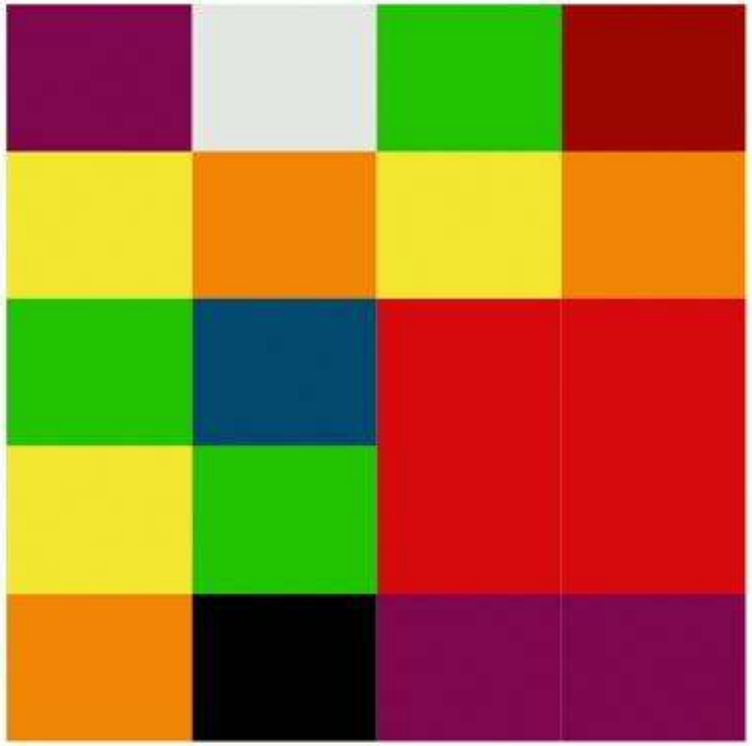
tap



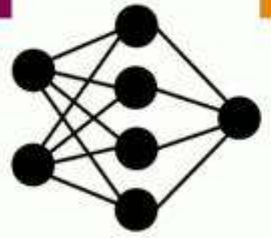


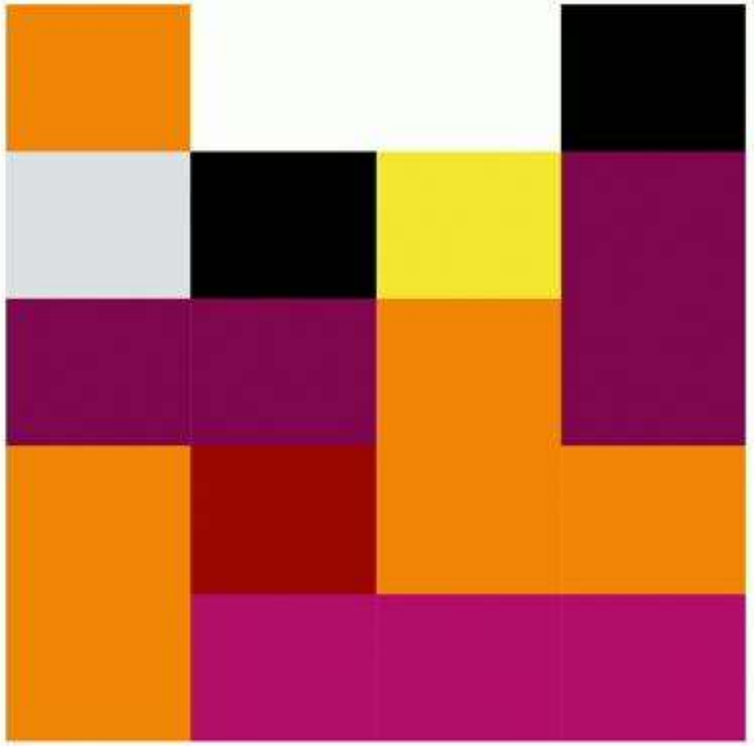
toc

ML Perspective



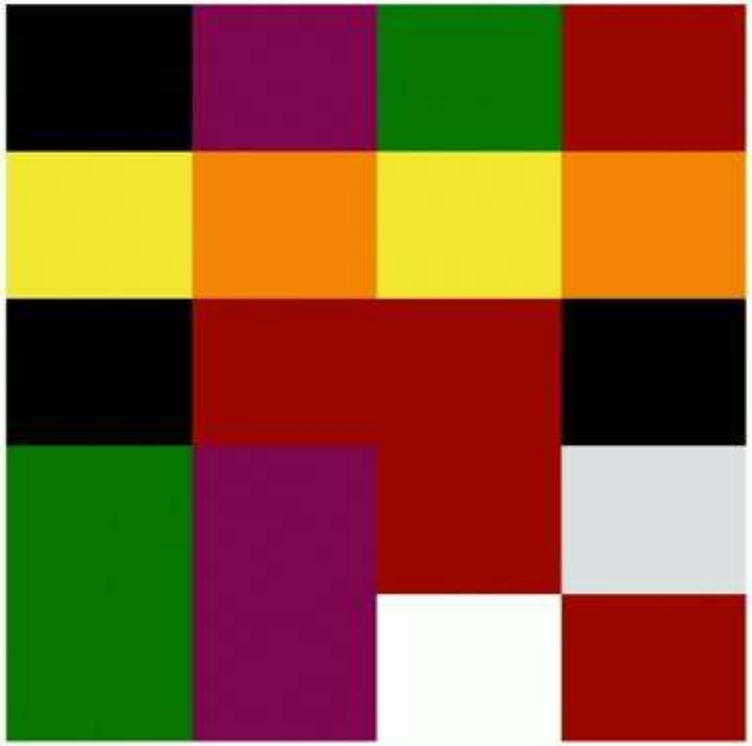
tap



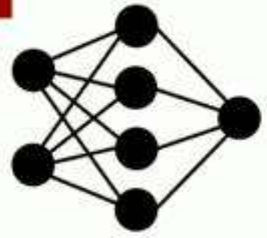


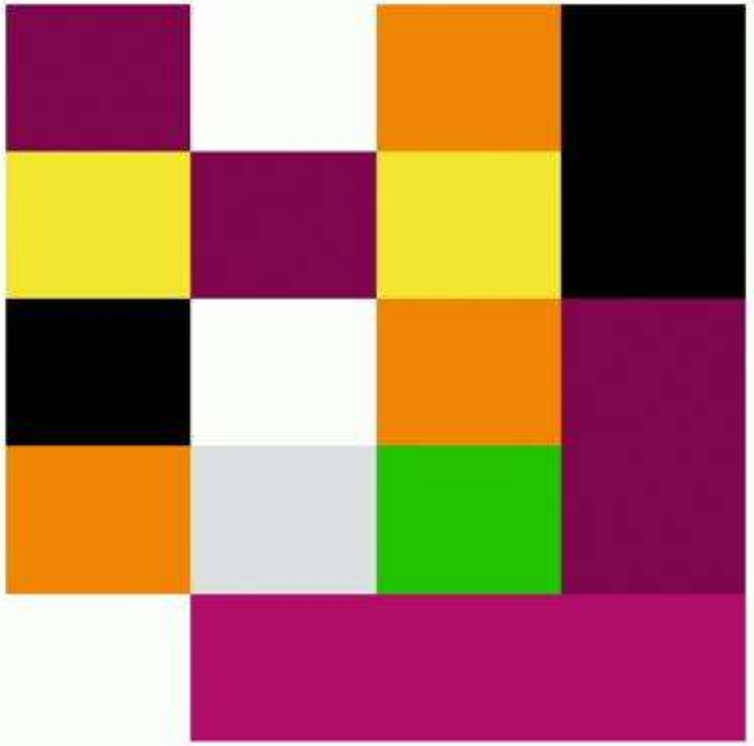
toc

ML Perspective



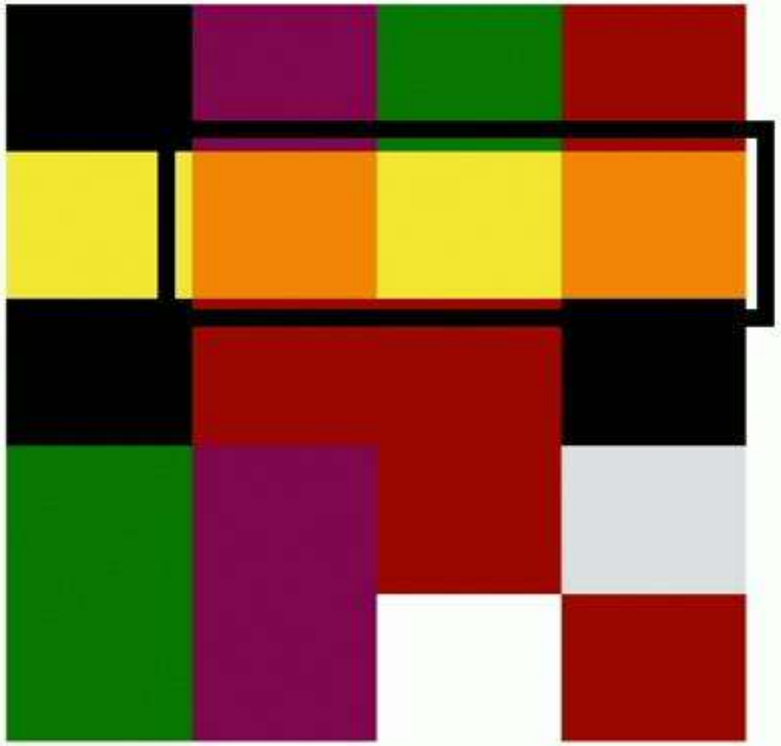
tap



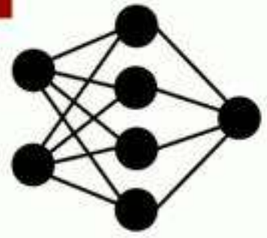


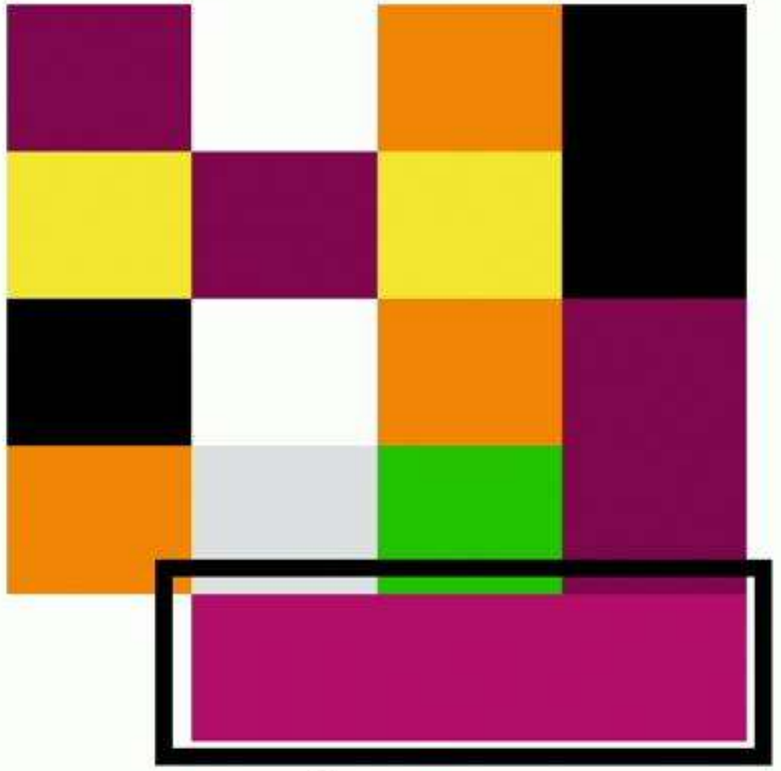
toc

ML Perspective



tap



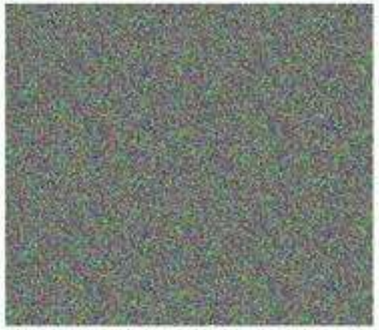


toc

ML Perspective

dog

+



meaningless
perturbation

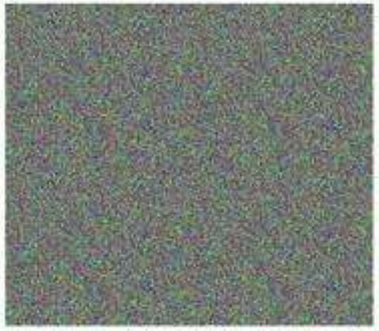
=

cat

ML Perspective

dog

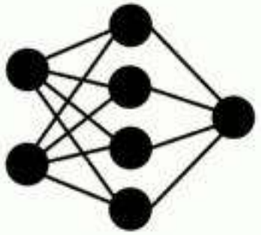
+



meaningless
perturbation

=

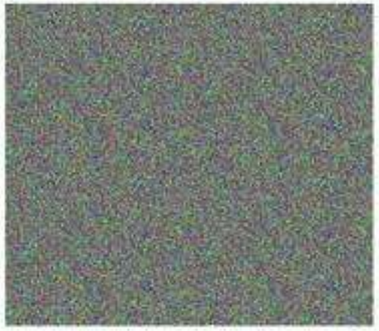
cat



ML Perspective

dog

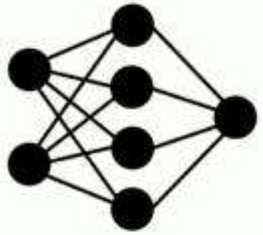
+

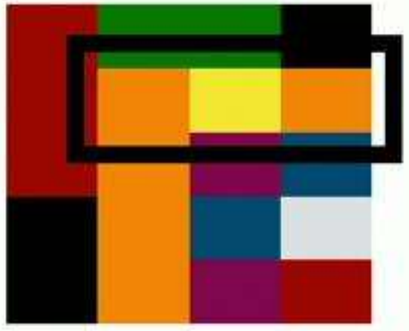


meaningless perturbation

=

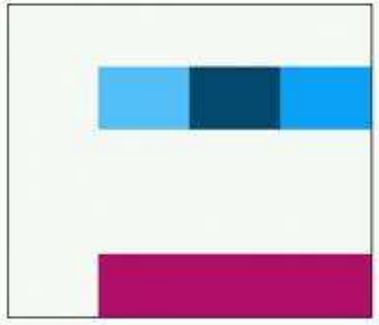
cat





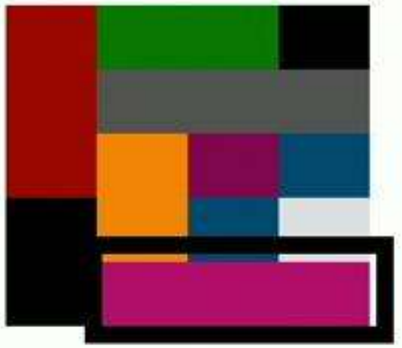
tap

+



meaningless perturbation

=

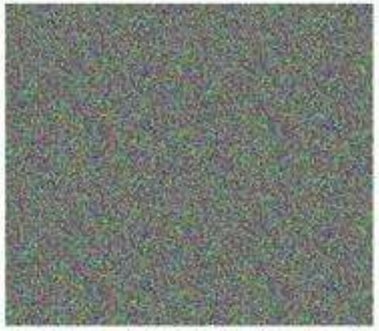


toc

ML Perspective

dog

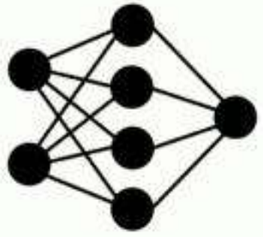
+

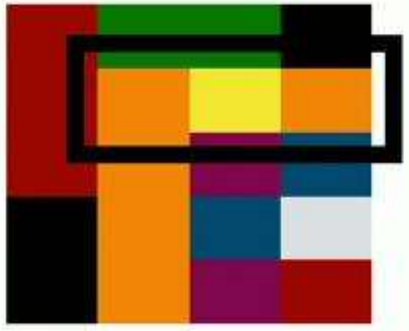


meaningless perturbation

=

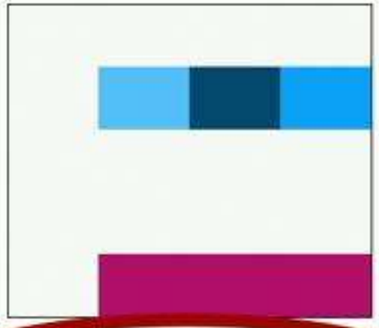
cat





tap

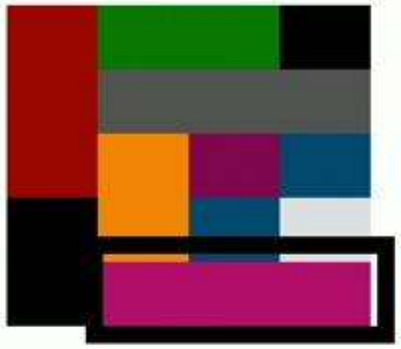
+



meaningless perturbation

?

=



toc

Are adversarial perturbations just meaningless artifacts?

[Ilyas Santurkar Tsipras Engstrom Tran M '19]

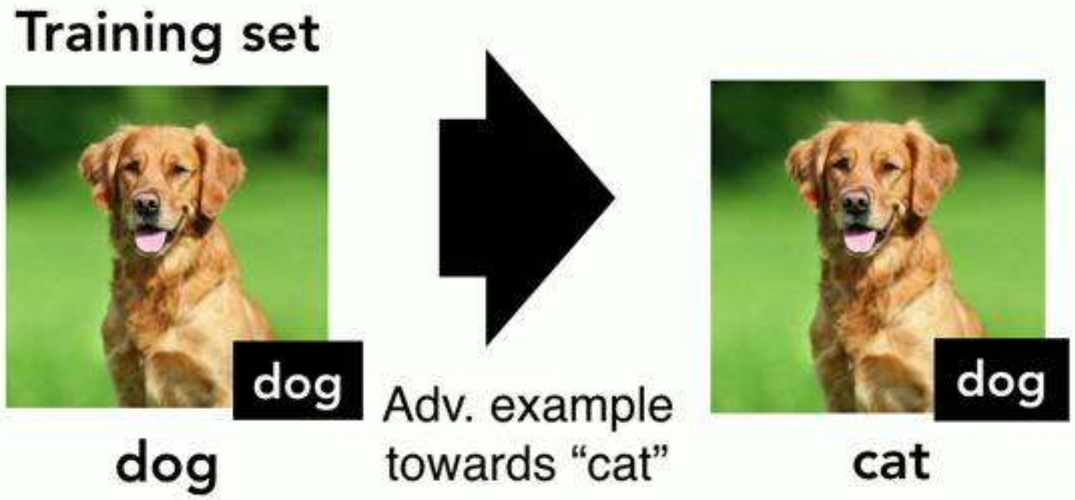
A Simple Experiment

A Simple Experiment

Training set

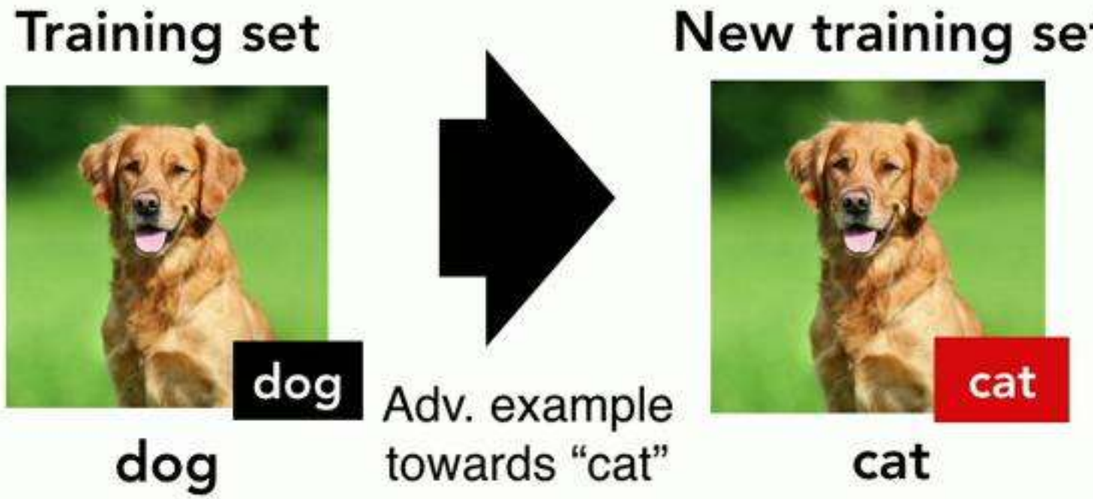
dog

A Simple Experiment



1. **Make adversarial example** towards the other class

A Simple Experiment



1. **Make adversarial example** towards the other class
2. **Relabel** the image as the target class

A Simple Experiment

1. **Make adversarial example** towards the other class
2. **Relabel** the image as the target class
3. Train with **new** dataset but test on the **original** test set

A Simple Experiment

So: We train on a “totally mislabeled” dataset but expect performance on a “correct” dataset

A Simple Experiment

So: We train on a “totally mislabeled” dataset but expect performance on a “correct” dataset

What will happen?

A Simple Experiment

Result: We get a **nontrivial accuracy** on the **original** classification task

A Simple Experiment

Result: We get a **nontrivial accuracy** on the **original** classification task

(For example, 78% on the CIFAR dog vs cat)

What's going on?

What's going on?

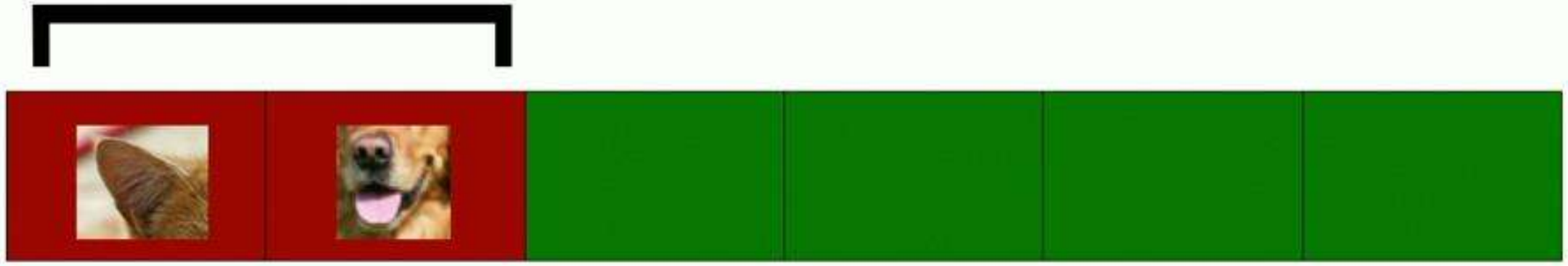
What if adversarial perturbations are **not** aberrations but **features**?

The Robust Features Model

The Robust Features Model

Robust features

Correlated with label
even with adversary



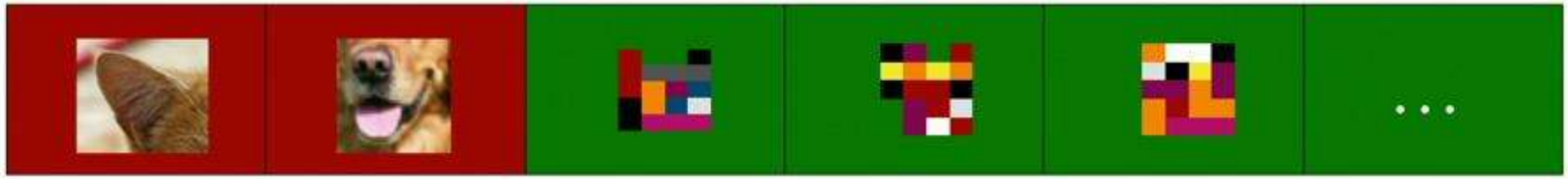
The Robust Features Model

Robust features

Correlated with label
even with adversary

Non-robust features

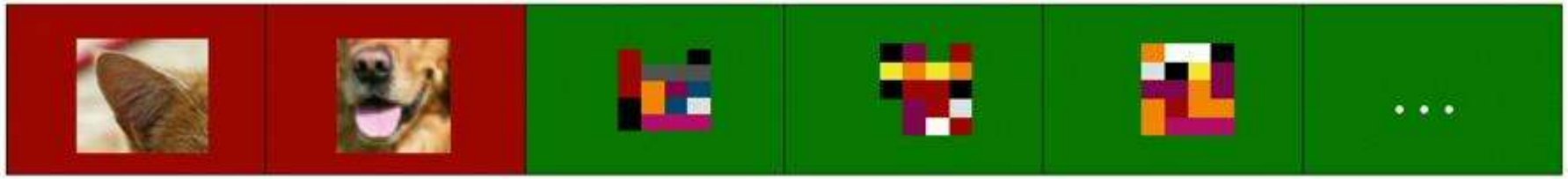
Correlated with label on average,
but can be flipped within, e.g., ℓ_2 ball



The Robust Features Model

Robust features
Correlated with label
even with adversary

Non-robust features
Correlated with label on average,
but can be flipped within, e.g., ℓ_2 ball

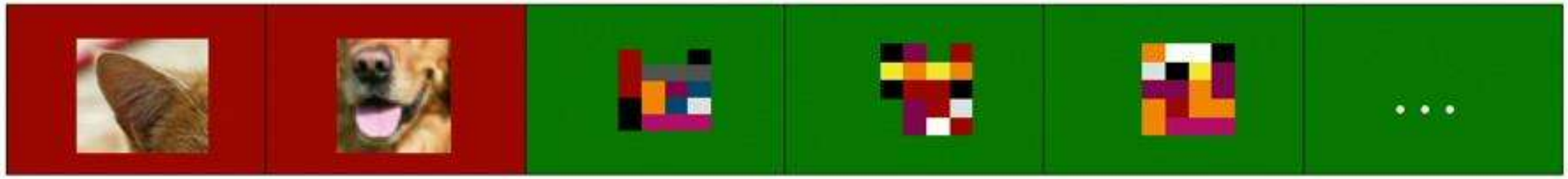


When maximizing (test) accuracy: All features are good

The Robust Features Model

Robust features
Correlated with label
even with adversary

Non-robust features
Correlated with label on average,
but can be flipped within, e.g., ℓ_2 ball



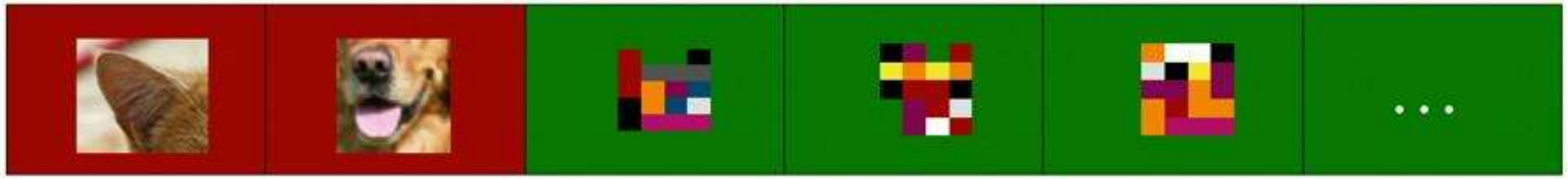
When maximizing (test) accuracy: All features are good

And: Non-robust features are often great!

The Robust Features Model

Robust features
Correlated with label
even with adversary

Non-robust features
Correlated with label on average,
but can be flipped within, e.g., ℓ_2 ball



When maximizing (test) accuracy: All features are good

And: Non-robust features are often great!

That's why our models pick on them
(and **become vulnerable to adversarial perturbations**)

The Simple Experiment: A Second Look

The Simple Experiment: A Second Look

Training set

dog

The Simple Experiment: A Second Look

Training set

dog

Robust features: dog

Non-robust features: dog

The Simple Experiment: A Second Look

Training set

dog

Adversarial example towards “cat”

New training set

cat

Robust features: dog
Non-robust features: dog

Robust features: dog
Non-robust features: cat

The Simple Experiment: A Second Look

Training set

New training set

All robust features are **misleading**

Robust features: dog
Non-robust features: dog

Robust features: dog
Non-robust features: cat

The Simple Experiment: A Second Look

Training set

New training set

All robust features are **misleading**

But: Non-robust features suffice for good generalization

The Simple Experiment: A Second Look

New training set

cat

Robust features: dog

Non-robust features: cat

The Simple Experiment: A Second Look

New training set

cat

Train

Test set

Robust features: dog

Non-robust features: cat

The Simple Experiment: A Second Look

New training set

cat

Train

Test set

Robust features: dog
Non-robust features: cat

Good test accuracy on original test set

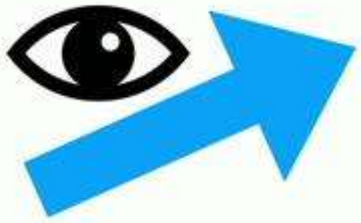
Human vs ML Model Priors

Human vs ML Model Priors

dog

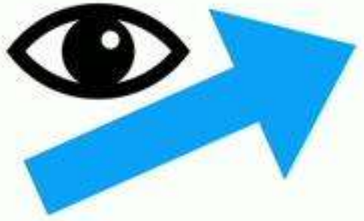
Human vs ML Model Priors

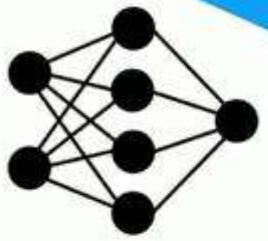
dog

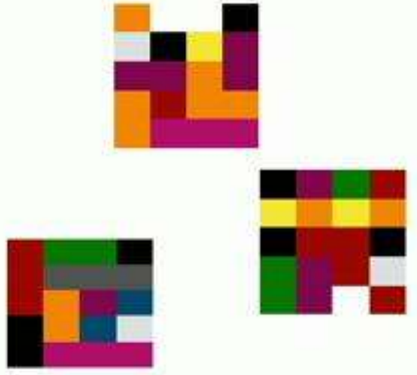


Human vs ML Model Priors

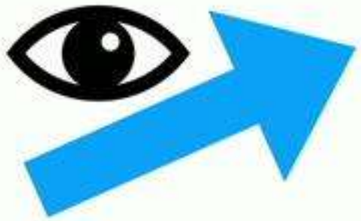
dog







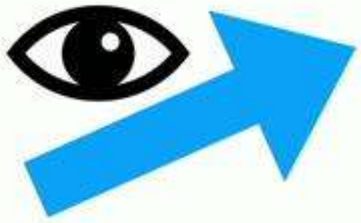
Human vs ML Model Priors

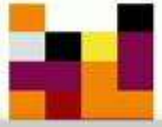


These are **equally valid** classification methods

dog

Human vs ML Model Priors





These are **equally valid** classification methods

No reason to expect our models to use the first one

Human vs ML Model Priors

Adversarial examples are a **human** phenomenon

Human vs ML Model Priors

Adversarial examples are a **human** phenomenon

No hope for interpretable models without intervention **at training time** (instead of post-hoc)

Human vs ML Model Priors

Adversarial examples are a **human** phenomenon

No hope for interpretable models without intervention **at training time** (instead of post-hoc)

Need **additional restrictions (priors)** on what features models should use to make predictions

What now?

What now?

A (new) perspective on adversarial robustness

What now?

A (new) perspective on adversarial robustness

(Provides insights into other questions too)

New capability: Robustification

New capability: Robustification

Training set

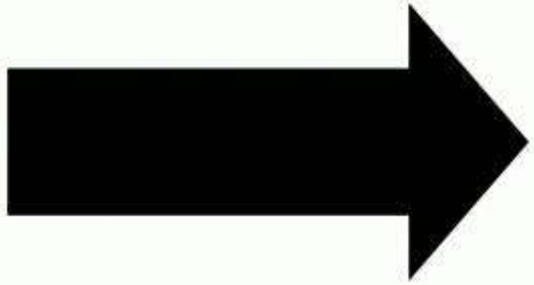
frog

New capability: Robustification

Training set

frog

Restrict to features
of robust model



New capability: Robustification

Training set

frog

Restrict to features of robust model

New training set

"robustified" frog

New capability: Robustification

Test set

Standard training

New training set

"robustified" frog

New capability: Robustification

Test set

Standard training

New training set

"robustified" frog

We get both standard and **robust** accuracy

New capability: Robustification

Test set

Standard training

New training set

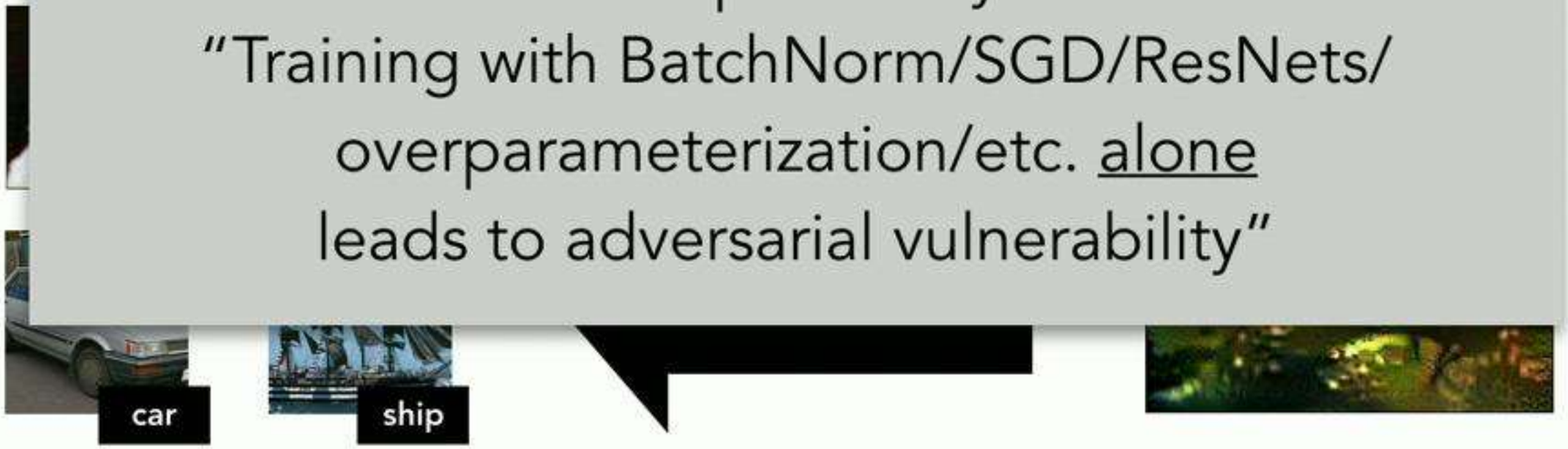
"robustified" frog

We get both standard and **robust** accuracy

So: It really is about features

New capability: Robustification

Also: Counterexample to any statement that "Training with BatchNorm/SGD/ResNets/ overparameterization/etc. alone leads to adversarial vulnerability"



"robustified" frog

We get both standard and **robust** accuracy

So: It really is about features

Some Direct Consequences

Some Direct Consequences

Transferability: Features = property of **datasets** (not models)

Some Direct Consequences

Transferability: Features = property of **datasets** (not models)

Effectiveness of Robust Training:
Makes features that are non-robust w.r.t. Δ **useless**

Some Direct Consequences

Transferability: Features = property of **datasets** (not models)

Effectiveness of Robust Training:
Makes features that are non-robust w.r.t. Δ **useless**

Effectiveness of Randomized Smoothing:
Overwhelms non-robust (w.r.t. Δ) features with noise

Robustness and Data Efficiency

Robustness and Data Efficiency

Robust models can only leverage **robust** features

Robustness and Data Efficiency

Robust models can only leverage **robust** features

(Even though non-robust features **do** help with generalization)

Robustness and Data Efficiency

Robust models can only leverage **robust** features

(Even though non-robust features **do** help with generalization)

→ Need **more data** to get a given (robust) accuracy
(vide **[Schmidt Santurkar Tsipras Talwar M '18]**)

Robustness and Data Efficiency

Robust models can only leverage **robust** features

(Even though non-robust features **do** help with generalization)

→ Need **more data** to get a given (robust) accuracy
(vide **[Schmidt Santurkar Tsipras Talwar M '18]**)

→ Will get a **lower standard accuracy**
(vide **[Tsipras Santurkar Engstrom Turner M '18]**)

Robustness and Data Efficiency

Robust models can only leverage **robust** features

(Even though non-robust features **do** help with generalization)

→ Need **more data** to get a given (robust) accuracy
(vide **[Schmidt Santurkar Tsipras Talwar M '18]**)

→ Will get a **lower standard accuracy**
(vide **[Tsipras Santurkar Engstrom Turner M '18]**)

But: Is leveraging non-robust features good?

A Simple Theoretical Setting:

Robust Max Likelihood Gaussian Classification

Robustness and Data Efficiency

Some Direct Consequences

Robustness and Data Efficiency

Robust models can only leverage **robust** features

(Even though non-robust features **do** help with generalization)

A Simple Theoretical Setting:

Robust Max Likelihood Gaussian Classification

Some Direct Consequences

Robustness and Data Efficiency

Robust models can only leverage **robust** features

(Even though non-robust features **do** help with generalization)

→ Need **more data** to get a given (robust) accuracy
(vide **[Schmidt Santurkar Tsipras Talwar M '18]**)

→ Will get a **lower standard accuracy**
(vide **[Tsipras Santurkar Engstrom Turner M '18]**)

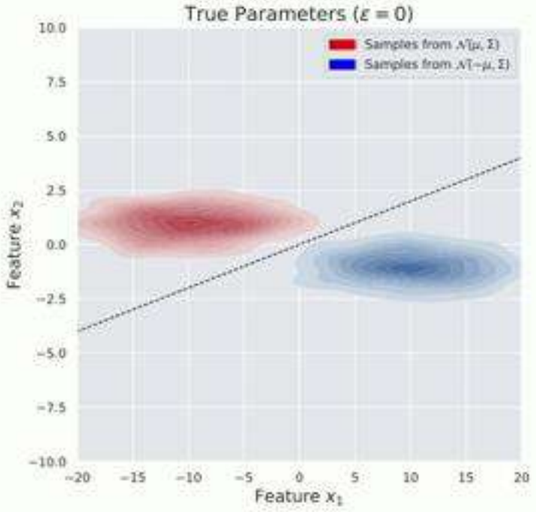
But: Is leveraging non-robust features good?

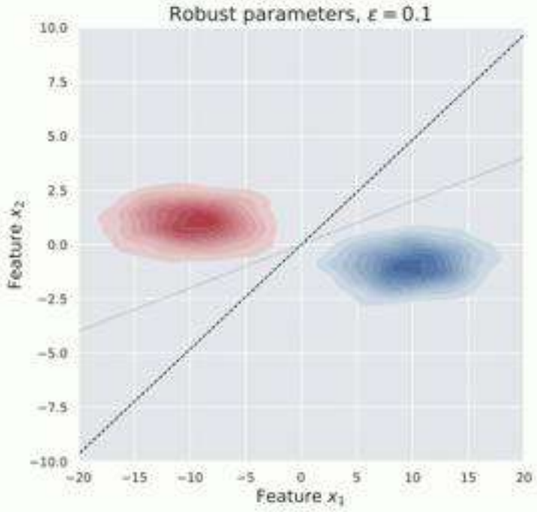
A Simple Theoretical Setting:

Robust Max Likelihood Gaussian Classification

A Simple Theoretical Setting:

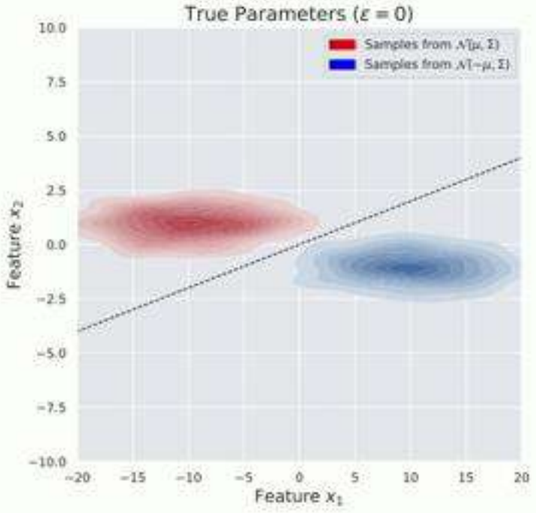
Robust Max Likelihood Gaussian Classification

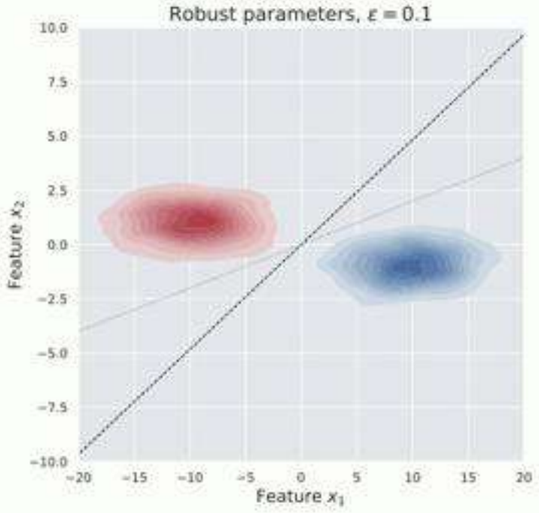


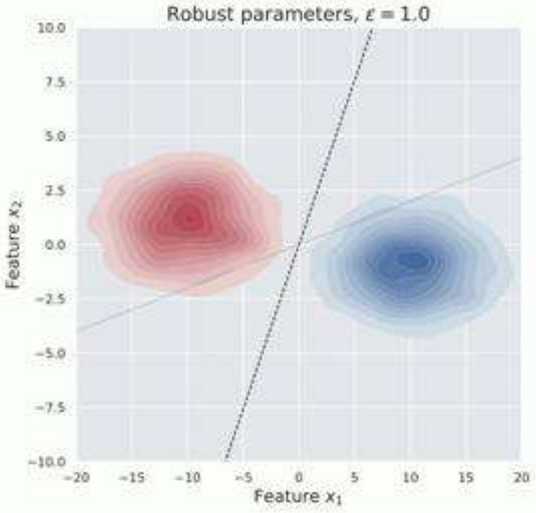


A Simple Theoretical Setting:

Robust Max Likelihood Gaussian Classification

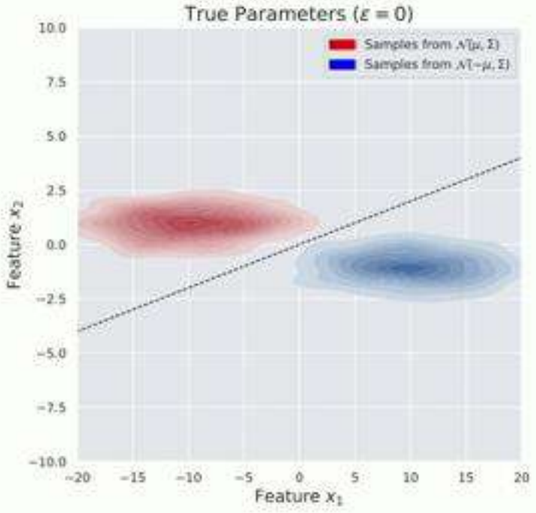


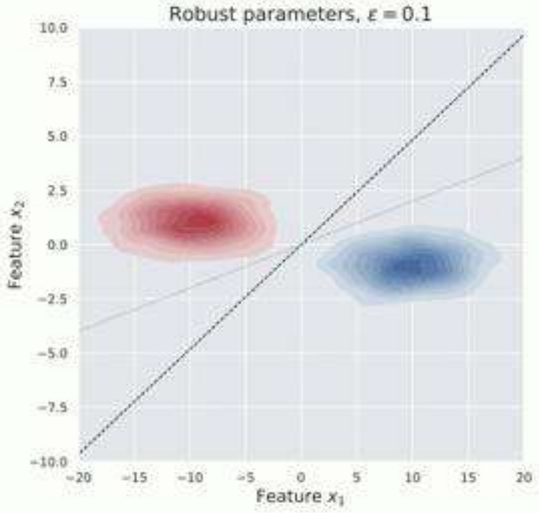


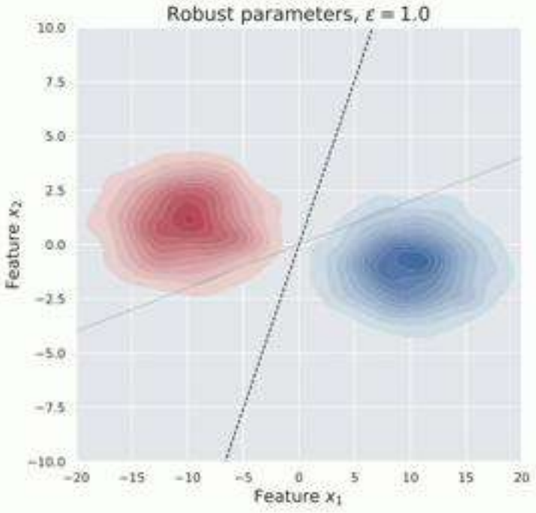


A Simple Theoretical Setting:

Robust Max Likelihood Gaussian Classification



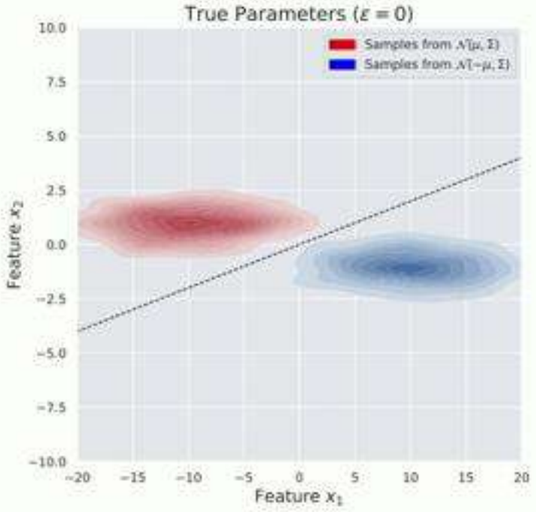


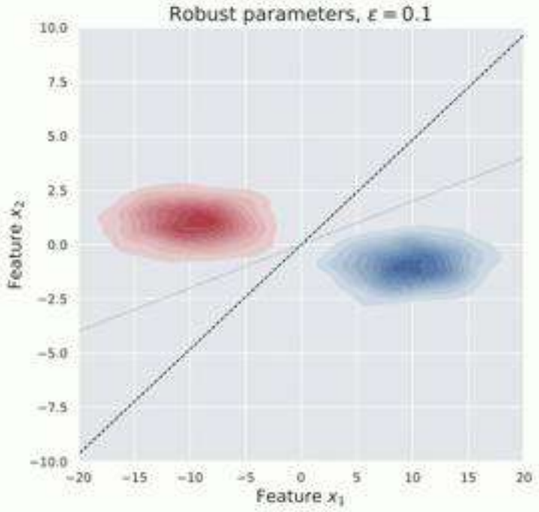


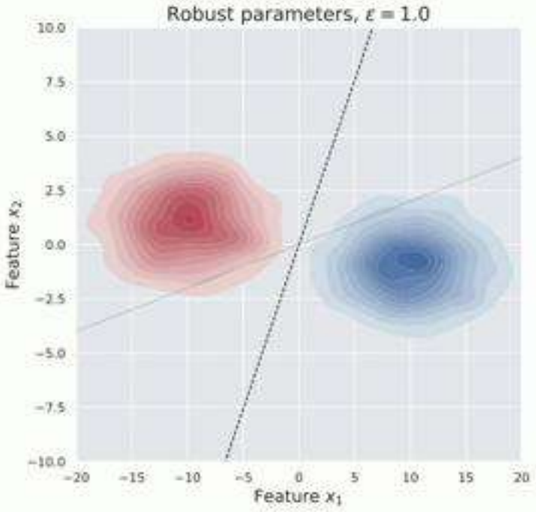
(Exact theorems in the paper)

A Simple Theoretical Setting:

Robust Max Likelihood Gaussian Classification





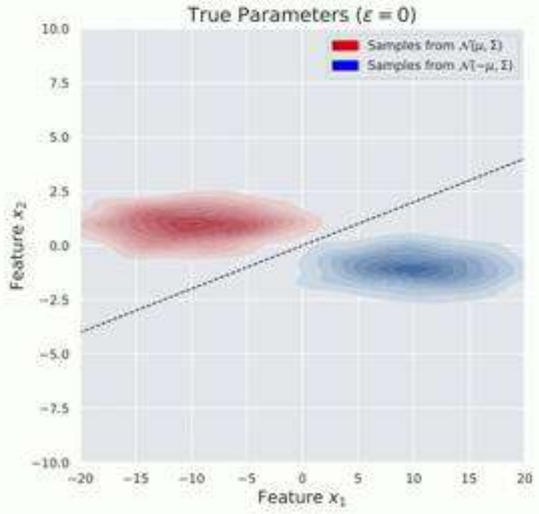


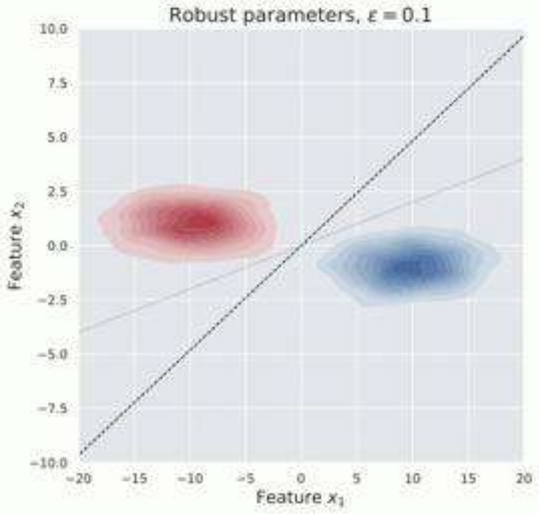
Things to observe:

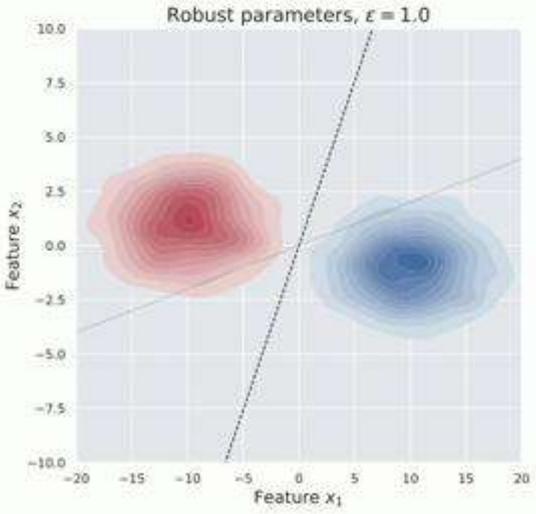
(Exact theorems in the paper)

A Simple Theoretical Setting:

Robust Max Likelihood Gaussian Classification







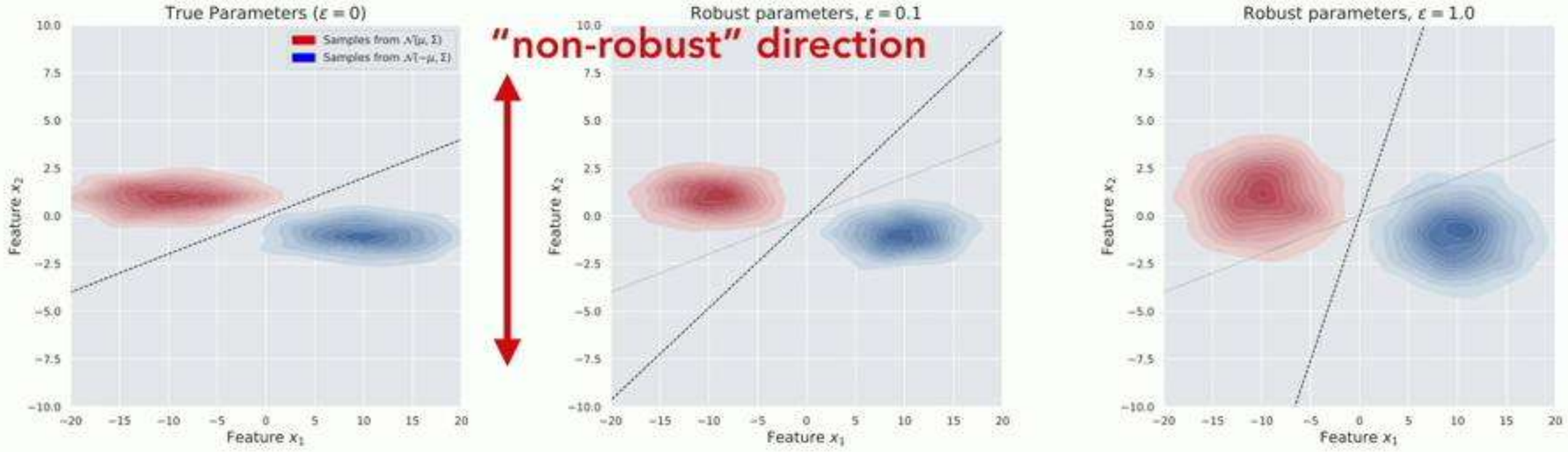
Things to observe:

→ Non-robust features are needed to get better standard accuracy but lead to vulnerability

(Exact theorems in the paper)

A Simple Theoretical Setting:

Robust Max Likelihood Gaussian Classification



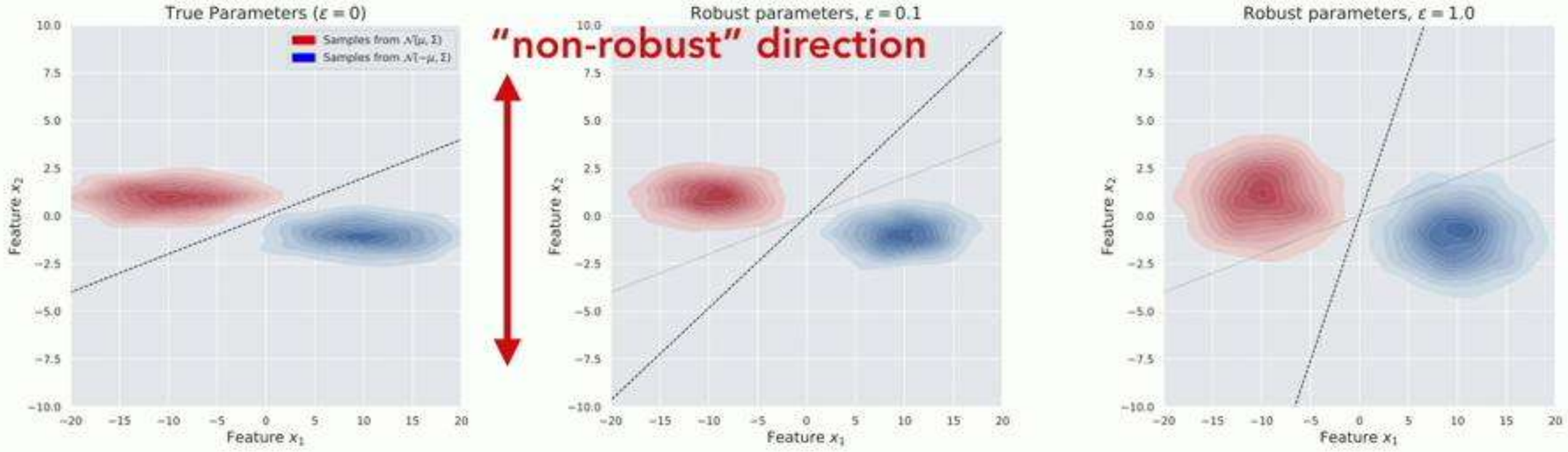
Things to observe:

→ Non-robust features are needed to get better standard accuracy but lead to vulnerability

(Exact theorems in the paper)

A Simple Theoretical Setting:

Robust Max Likelihood Gaussian Classification



Things to observe:

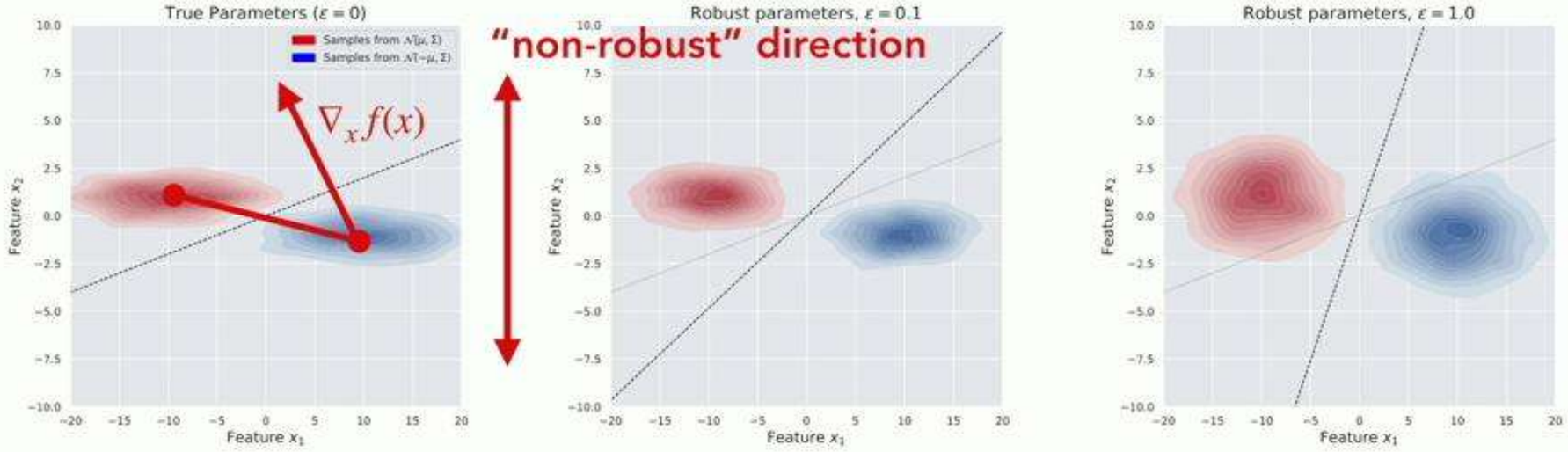
→ Non-robust features are needed to get better standard accuracy but lead to vulnerability

→ Gradient directions in robust models are more aligned with the"semantic"/human-preferred direction (will get back to this)

(Exact theorems in the paper)

A Simple Theoretical Setting:

Robust Max Likelihood Gaussian Classification



Things to observe:

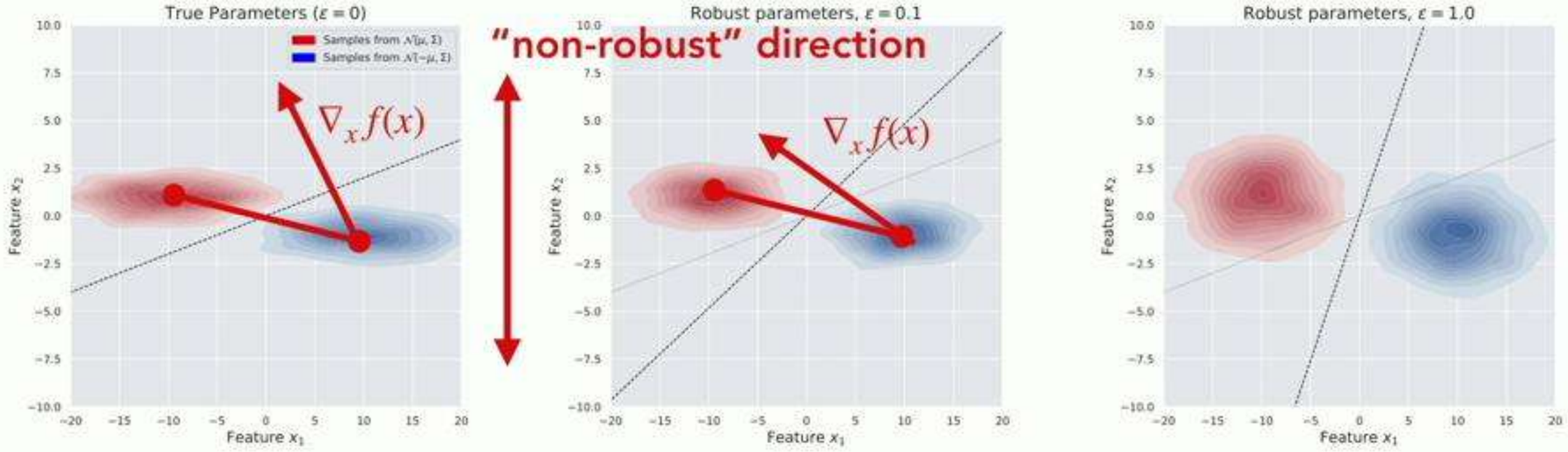
→ Non-robust features are needed to get better standard accuracy but lead to vulnerability

→ Gradient directions in robust models are more aligned with the"semantic"/human-preferred direction (will get back to this)

(Exact theorems in the paper)

A Simple Theoretical Setting:

Robust Max Likelihood Gaussian Classification



Things to observe:

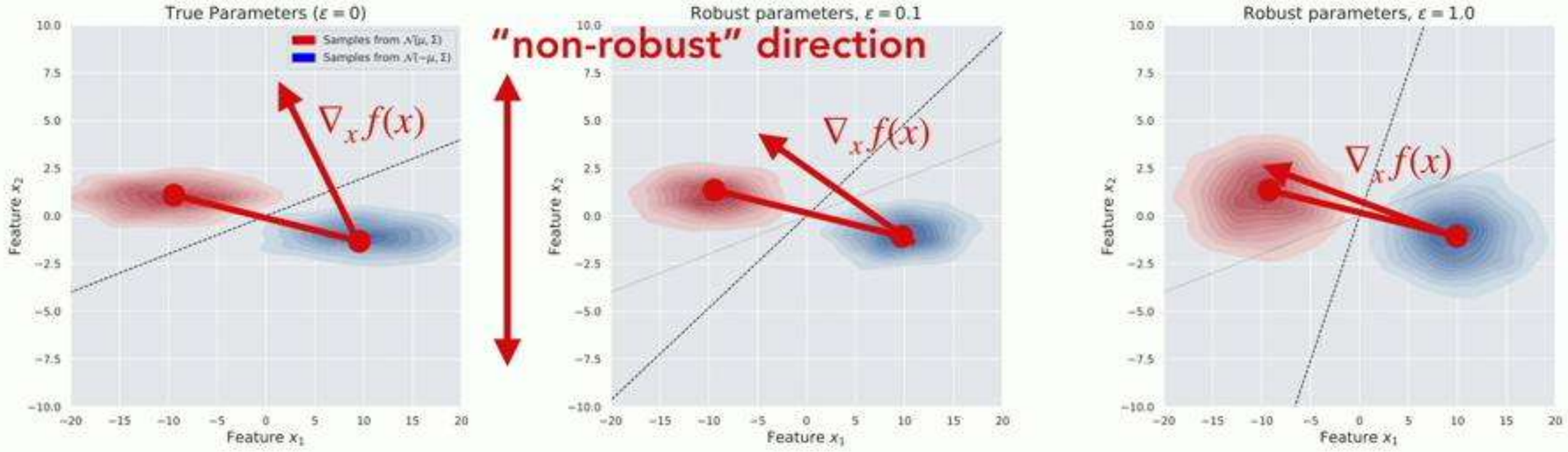
→ Non-robust features are needed to get better standard accuracy but lead to vulnerability

→ Gradient directions in robust models are more aligned with the"semantic"/human-preferred direction (will get back to this)

(Exact theorems in the paper)

A Simple Theoretical Setting:

Robust Max Likelihood Gaussian Classification



Things to observe:

→ Non-robust features are needed to get better standard accuracy but lead to vulnerability

→ Gradient directions in robust models are more aligned with the"semantic"/human-preferred direction (will get back to this)

(Exact theorems in the paper)

What if we **prevent** models from learning **non-robust** features?

[Tsipras Santurkar Engstrom Turner M '18]

[Engstrom Ilyas Santurkar Tsipras Tran M '19]

Robustness → Perception Alignment

Robustness → Perception Alignment

Models become more (human) perception aligned

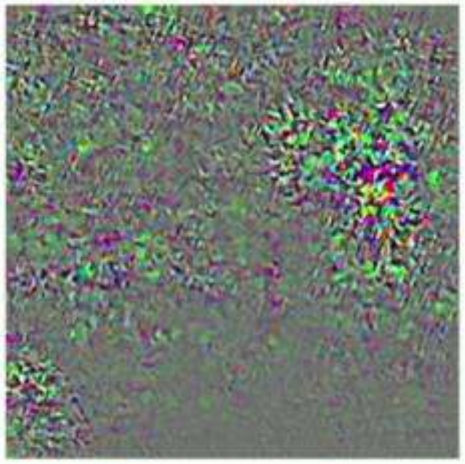
Robustness → Perception Alignment

Input

Models become more (human) perception aligned

Robustness → Perception Alignment

Input

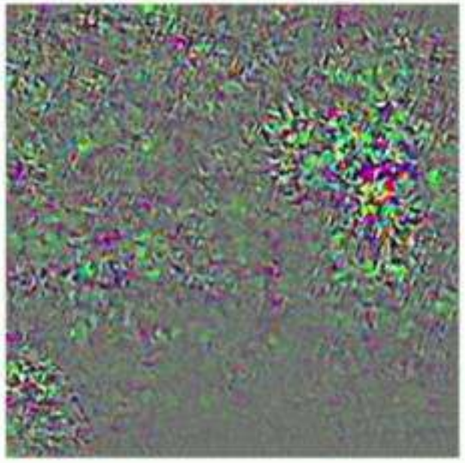


Gradient of
standard model

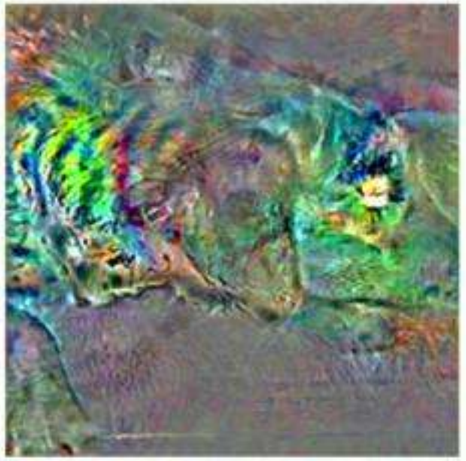
Models become more (human) perception aligned

Robustness → Perception Alignment

Input



Gradient of
standard model

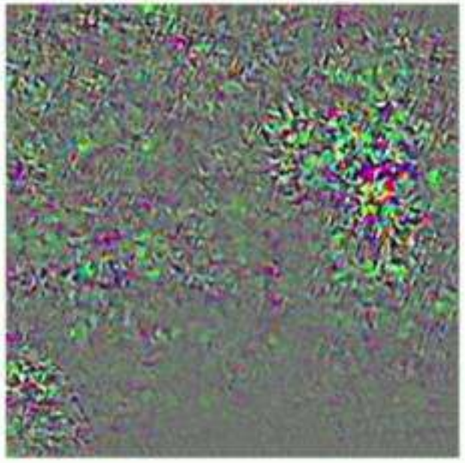


Gradient of
adv. robust model

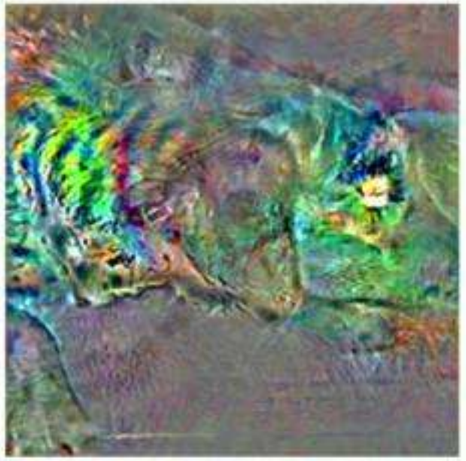
Models become more (human) perception aligned

Robustness → Perception Alignment

Input



Gradient of
standard model



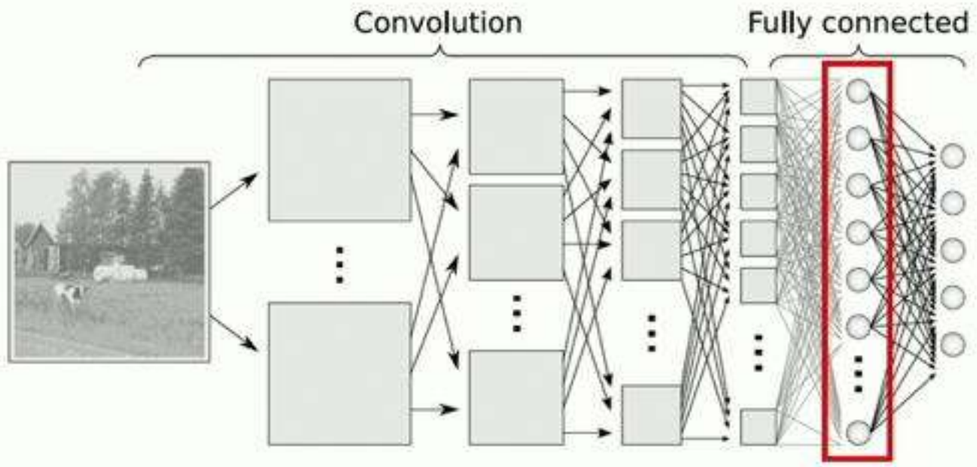
Gradient of
adv. robust model

Models become more (human) perception aligned

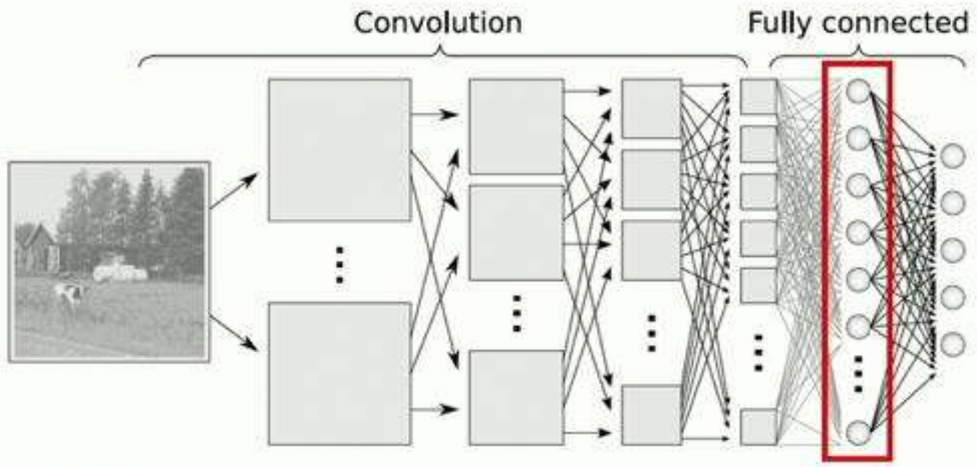
→ Robustness acts as a **prior** for "meaningful" features

Robustness → Better Representations

Robustness → Better Representations



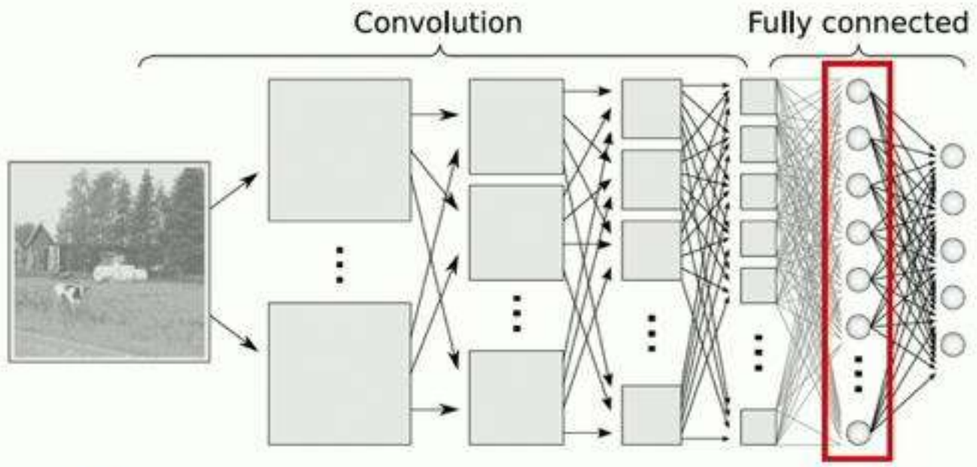
Robustness → Better Representations



≈

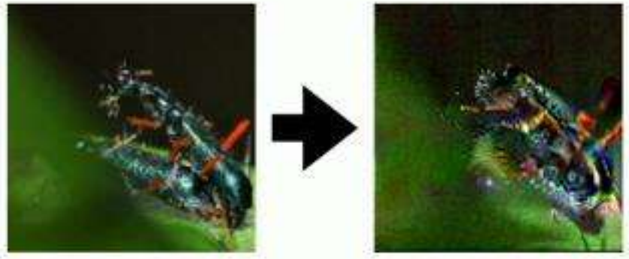
Standard Representation

Robustness → Better Representations



≈

Standard Representation



Robust Representation

Robustness → Better Representations

Robust representations enable a wide range of feature manipulations/visualizations in a **simple** way

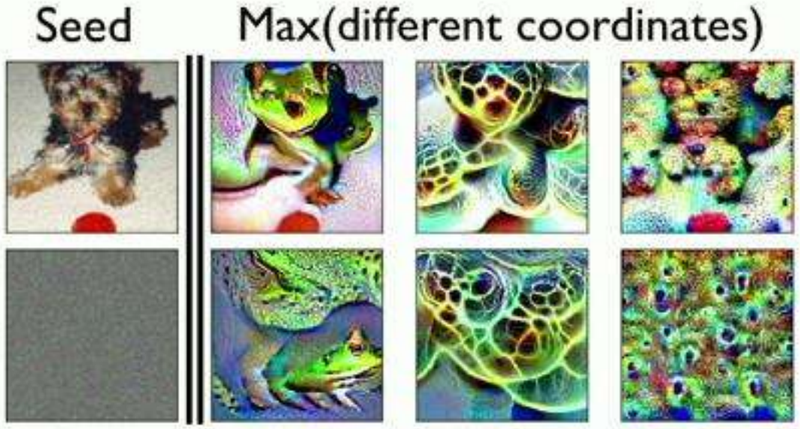
Robustness → Better Representations

Interpolation between **any** two inputs

Robustness → Better Representations

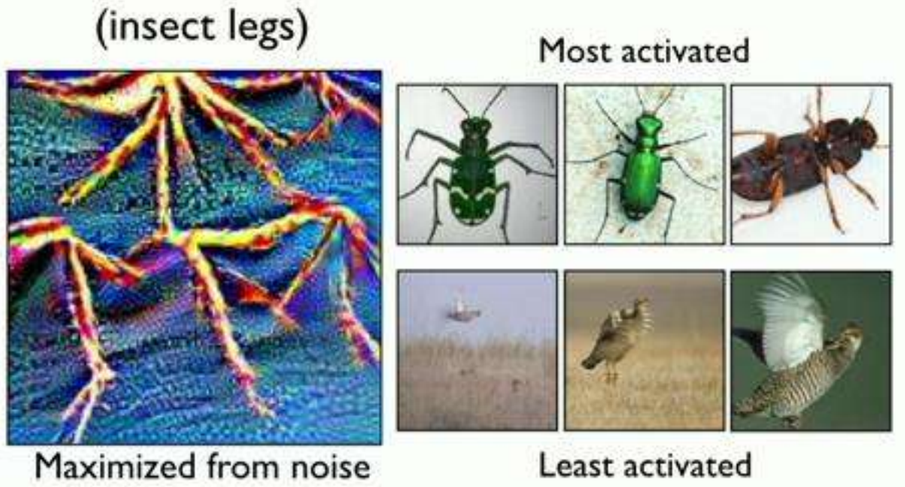
Direct feature visualization

Robustness → Better Representations



Direct feature visualization

Robustness → Better Representations



Direct feature visualization

Robustness → Better Representations

Add stripes

Direct feature manipulation

Robustness → Better Representations

Original image

label: “insect”; prediction: “dog”

Feature-level sensitivity analysis

Robustness → Better Representations

Original image

Top two features

label: "insect"; prediction: "dog"

Feature-level sensitivity analysis

Robustness → Better Representations

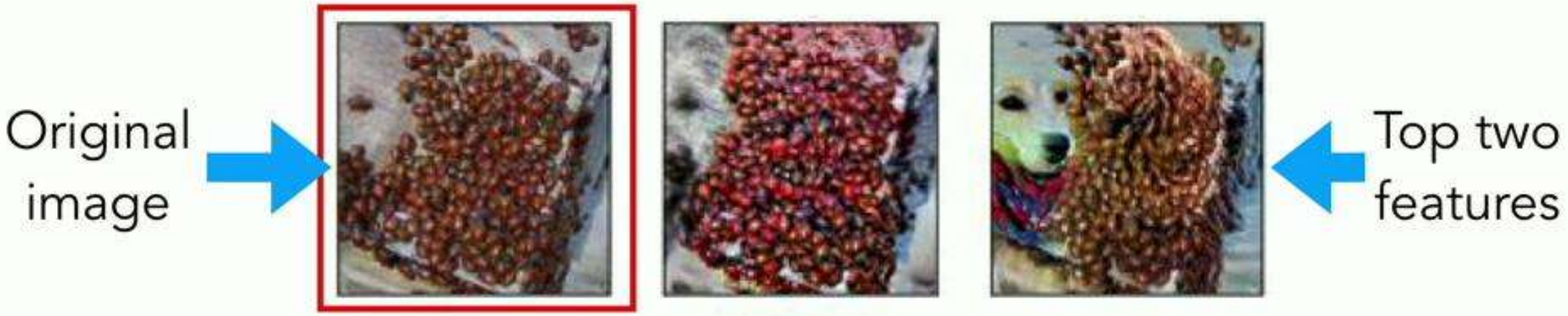
Original image

Top two features

label:“insect”; prediction:“dog”

Feature-level sensitivity analysis

Robustness → Better Representations



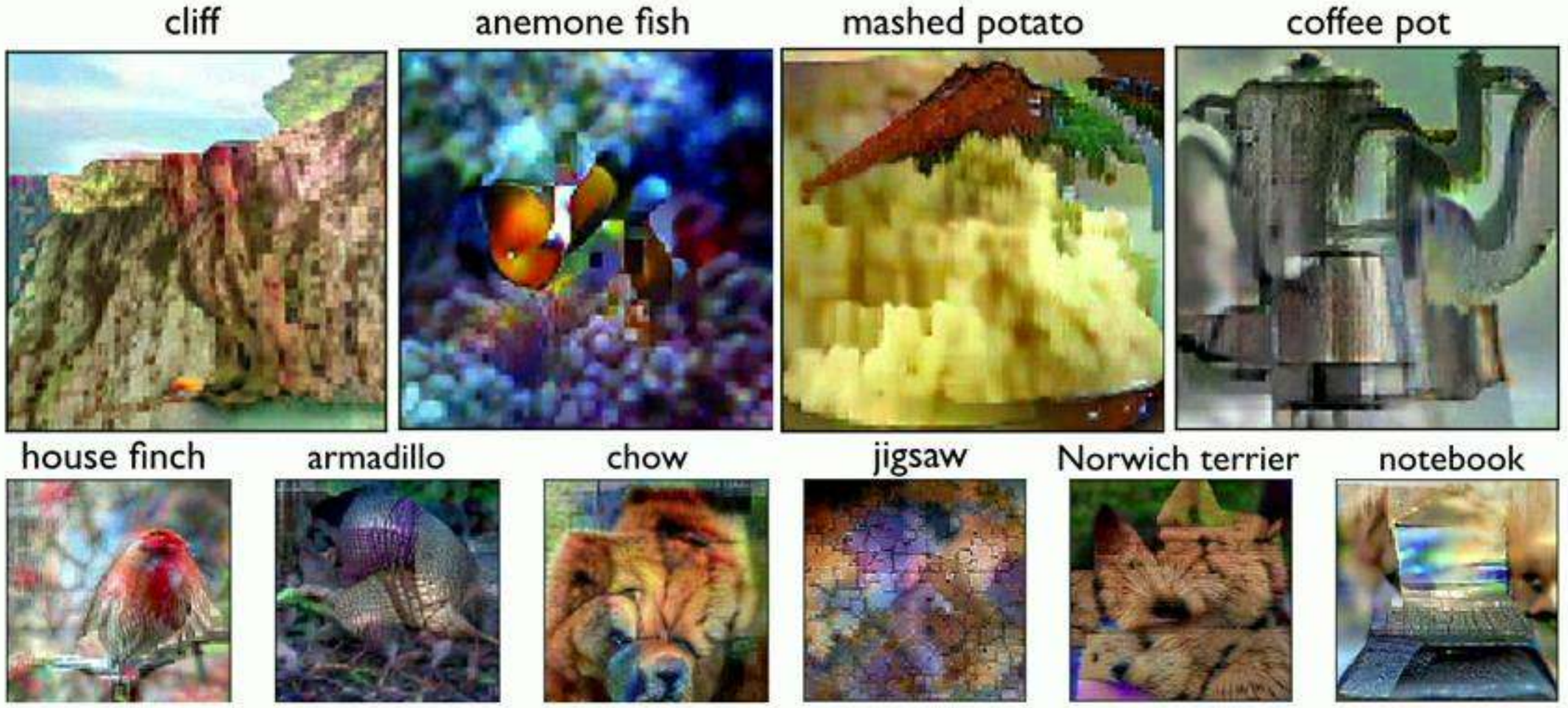
label: "insect"; prediction: "dog"

Feature-level sensitivity analysis

What else can we do?

[Santurkar Tsipras Tran Ilyas Engstrom M ’19]

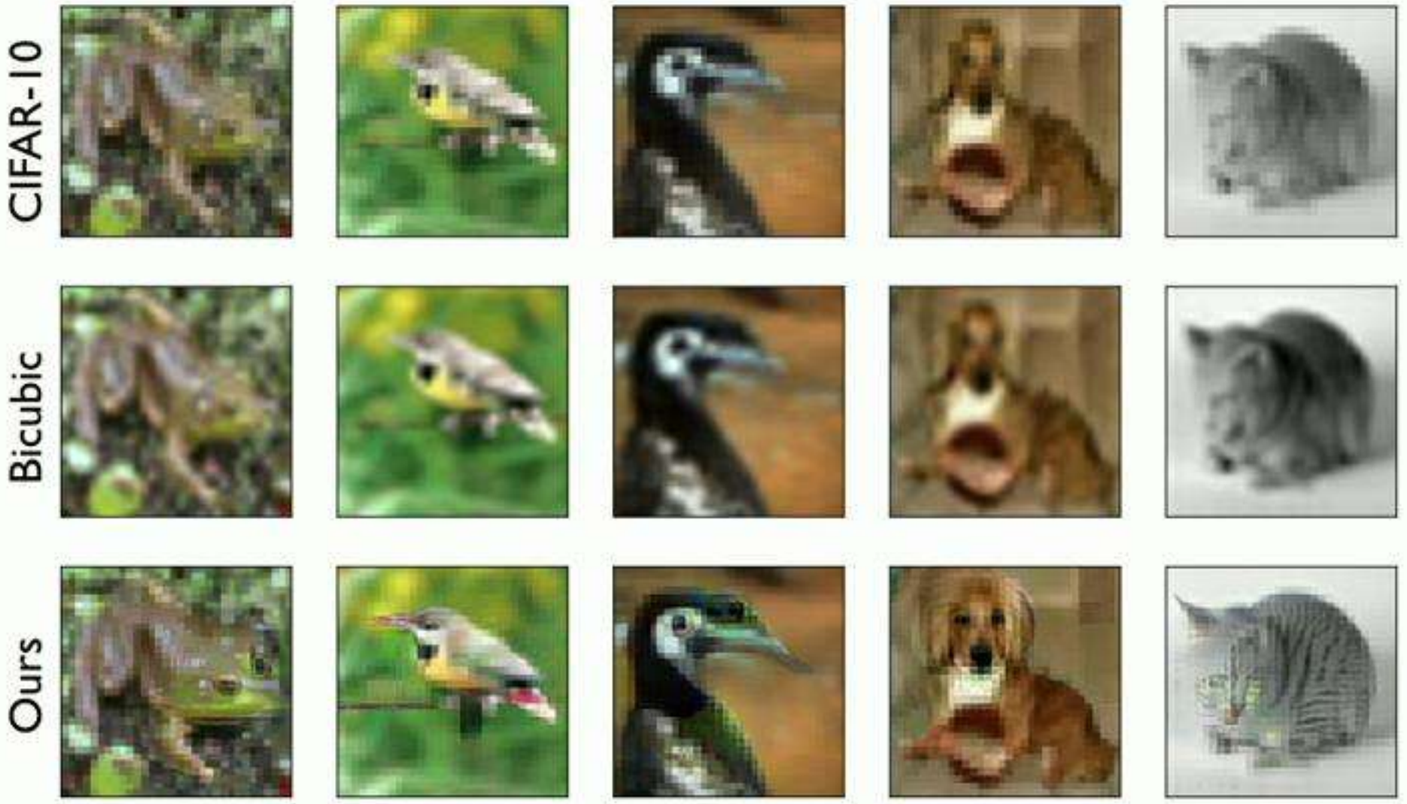
Robustness → CV Applications



(Random samples, 1K training images, no tuning)

Generative models (that work **better** on **large** datasets)

Robustness → CV Applications

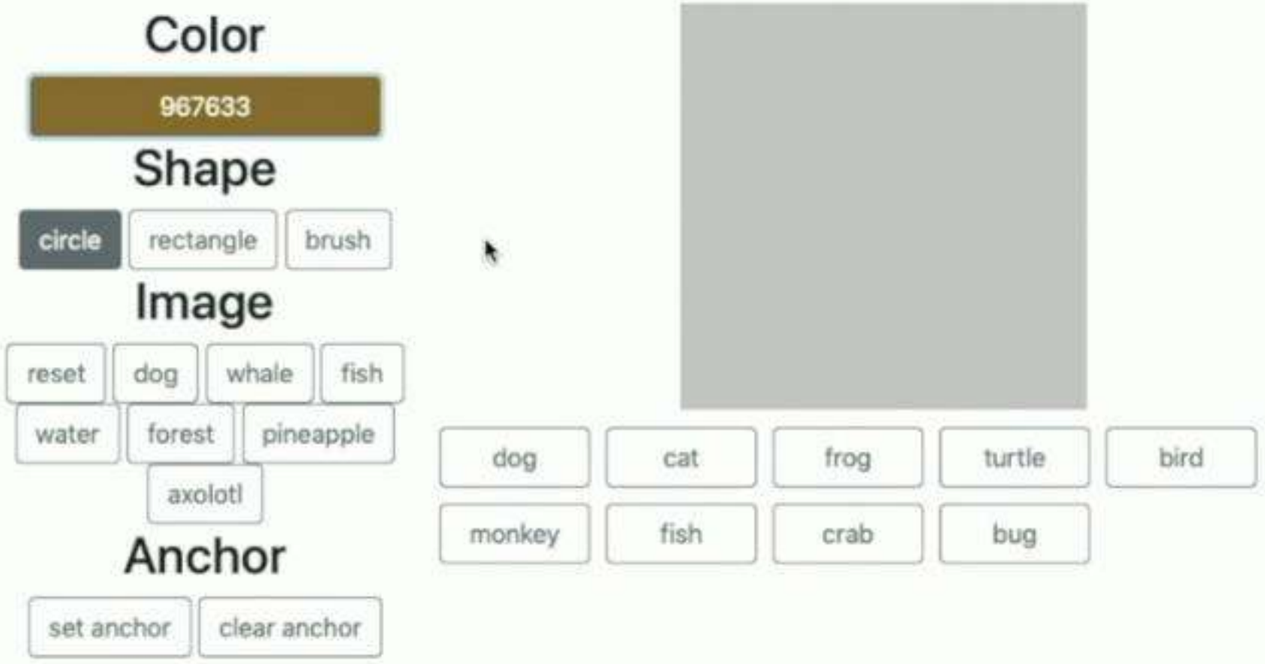


Super-Resolution

Robustness → CV Applications

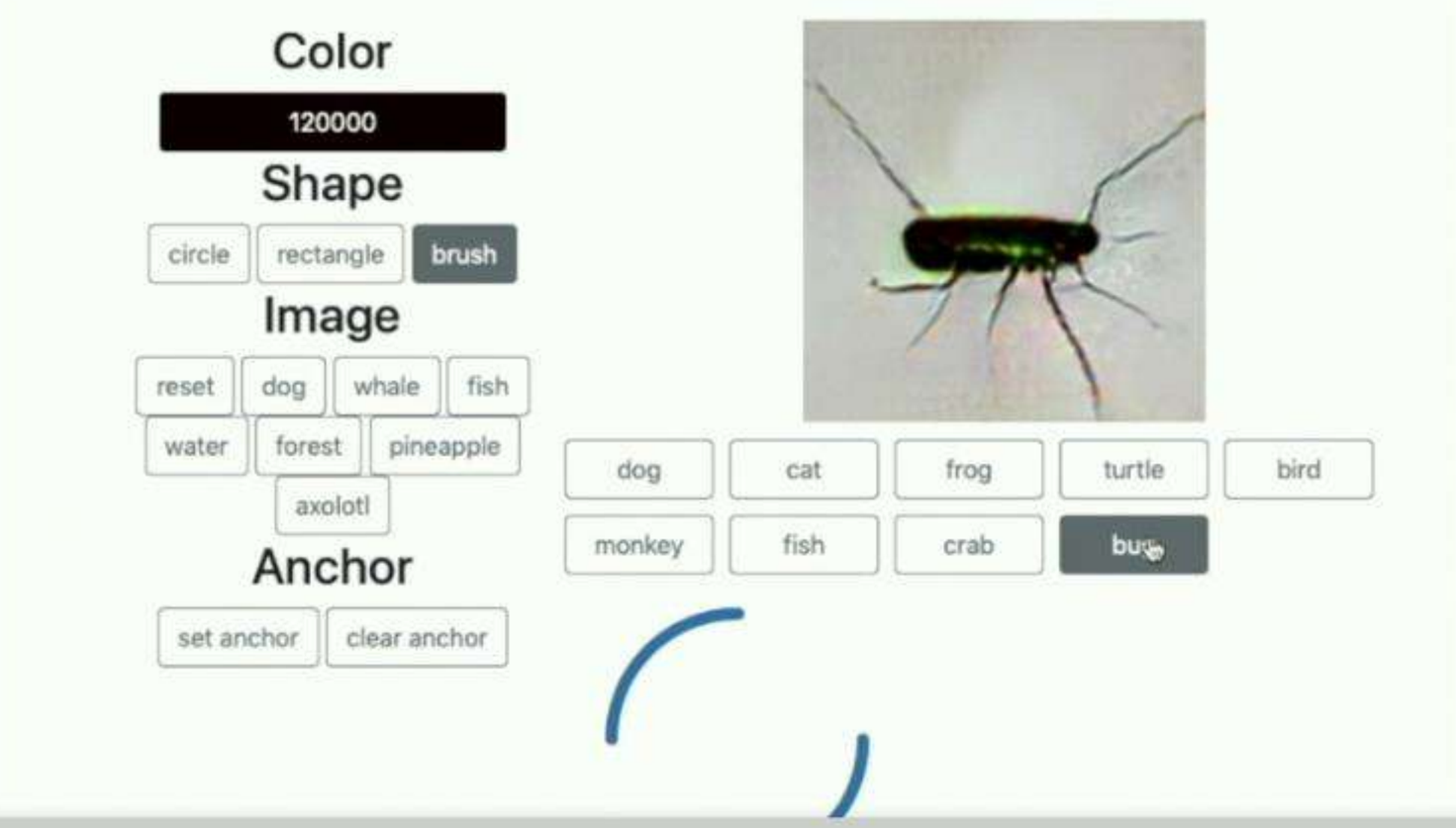
In-Painting

Robustness → CV Applications



Interactive **image class** manipulation

Robustness → CV Applications

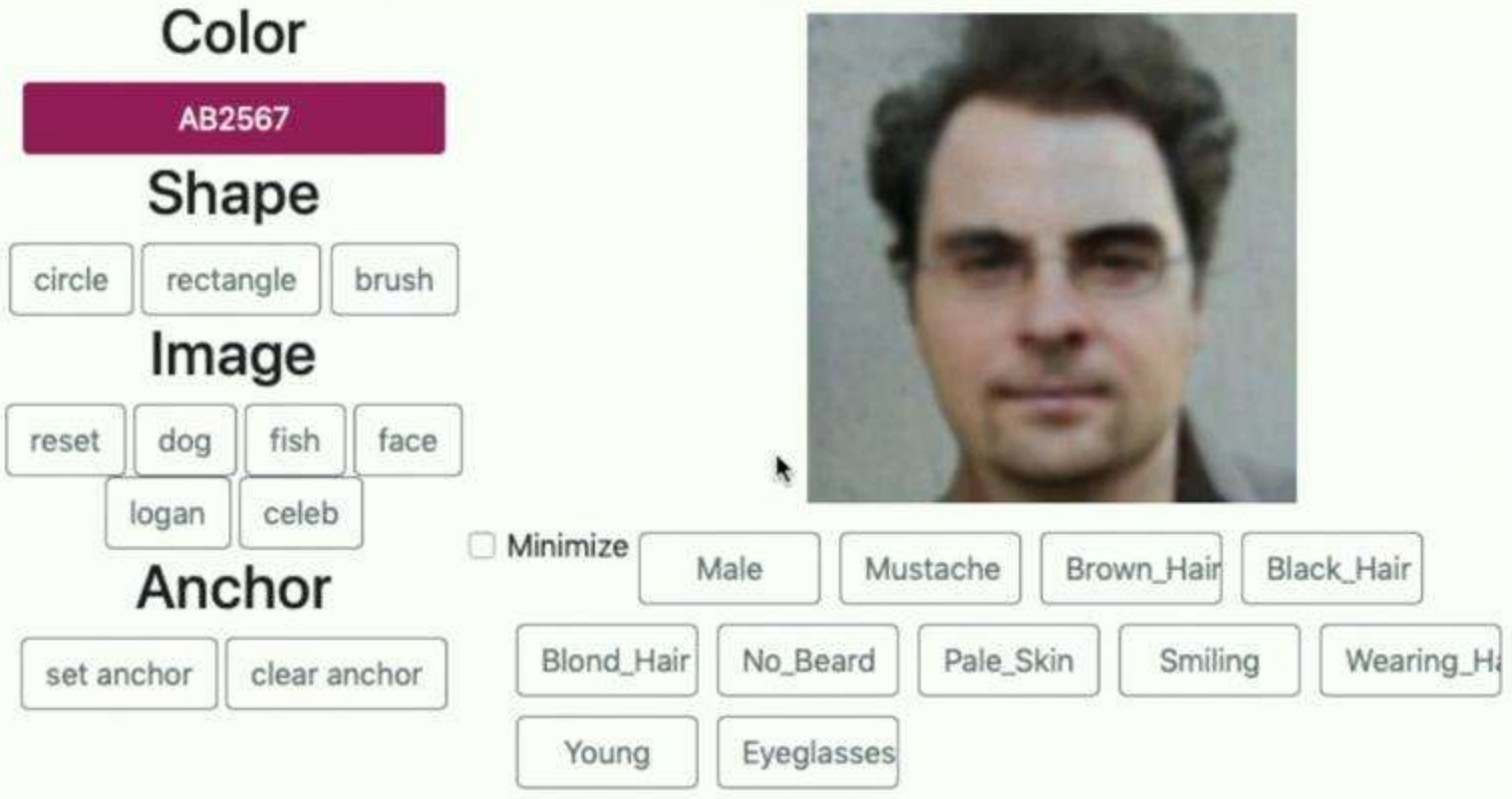


Interactive **image class** manipulation

Robustness → CV Applications

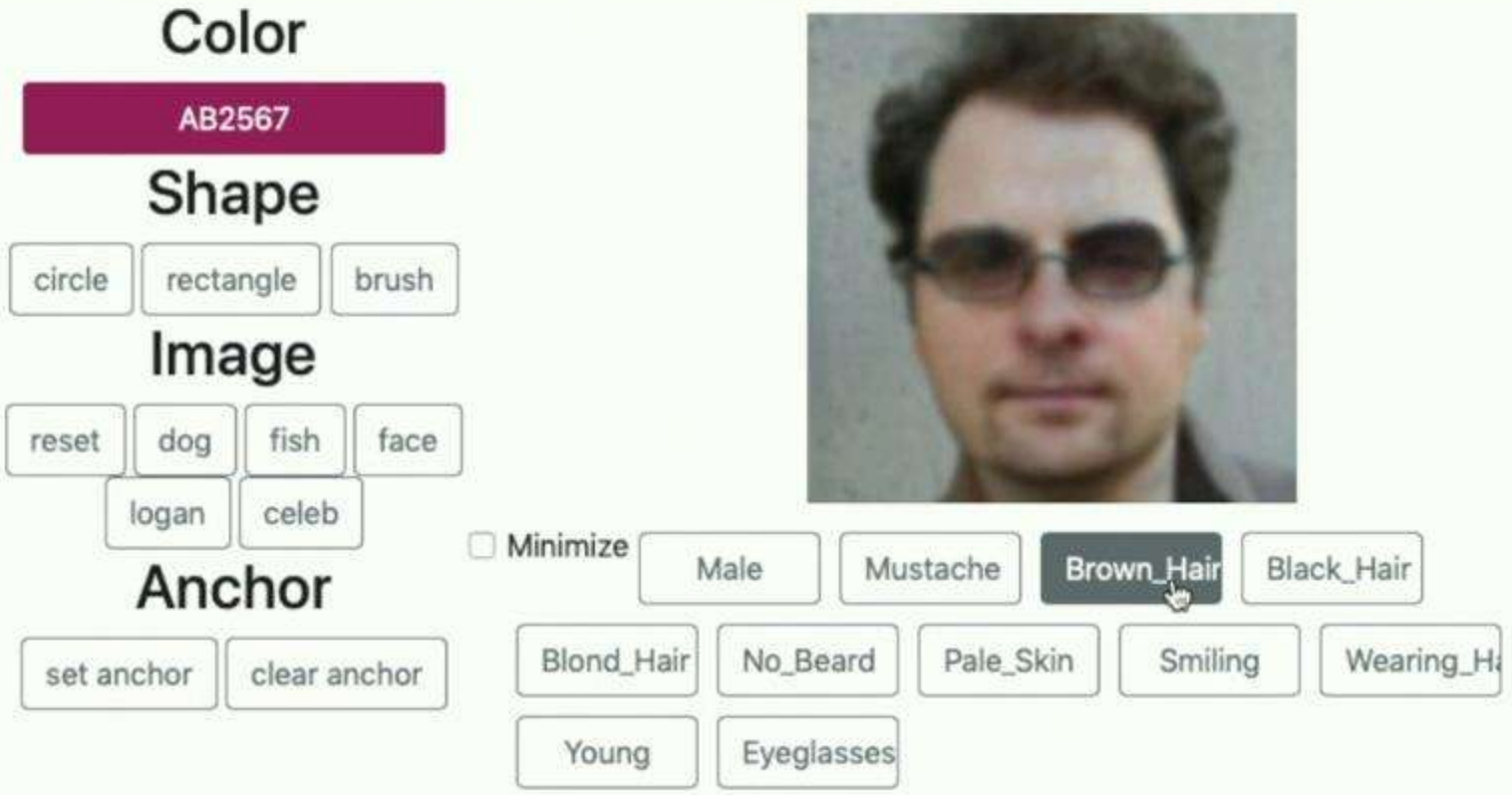
Enables exploration of data space

Robustness → CV Applications



Enables exploration of data space

Robustness → CV Applications



Enables exploration of data space

Robustness → CV Applications

Enables exploration of data space

Robustness → CV Applications

Enables exploration of data space

See: http://bit.ly/robustness_demo

Robustness → CV Applications

Enables exploration of data space

See: http://bit.ly/robustness_demo

Takeaways

But: Adv. robustness is not only about robustness to an adversary → it's about **how our models learn**

But: Adv. robustness is not only about robustness to an adversary → it's about **how our models learn**

→ What is the "right" notion of generalization?
Is it really about getting max accuracy possible?

→ How to measure distribution shift?
Shouldn't it be more about representations?

→ How much do we value human alignment/interpretability?

Adversarial robustness =
Framework for making our models better

But: Adv. robustness is not only about robustness to an adversary → it's about **how our models learn**

→ What is the "right" notion of generalization?
Is it really about getting max accuracy possible?

→ How to measure distribution shift?
Shouldn't it be more about representations?

→ How much do we value human alignment/interpretability?

Adversarial robustness =
Framework for making our models better

Here: "Adversary" corresponds to a "human critic"

But: Adv. robustness is not only about robustness to an adversary → it's about **how our models learn**

→ What is the "right" notion of generalization?
Is it really about getting max accuracy possible?

→ How to measure distribution shift?
Shouldn't it be more about representations?

→ How much do we value human alignment/interpretability?

Adversarial robustness =
Framework for making our models better

Here: "Adversary" corresponds to a "human critic"

gradientscience.org

Graph neural networks and graph isomorphism

Soledad Villar

Center for Data Science
Courant Institute of Mathematical Sciences

NEW YORK UNIVERSITY

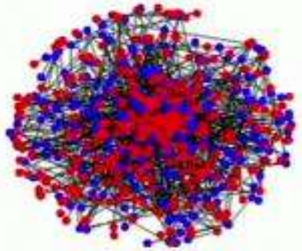
Geometry of Deep Learning
Microsoft, August 27 2019

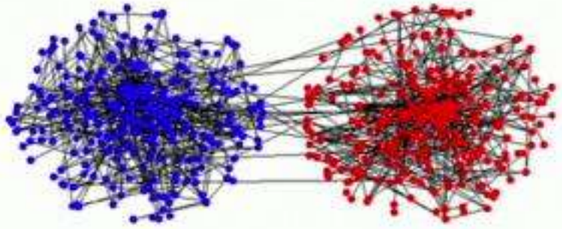
Motivating example 1

Clustering the stochastic block model

$A \sim SBM(p, q, n, 2)$ (two equal-sized communities):

$$\mathbb{P}(A_{ij} = 1) = \begin{cases} p & \text{if } i, j \text{ in the same community} \\ q & \text{if } i, j \text{ in different communities} \end{cases}$$



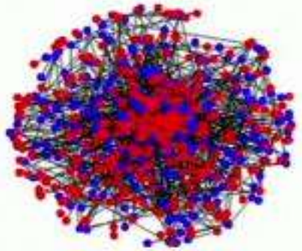


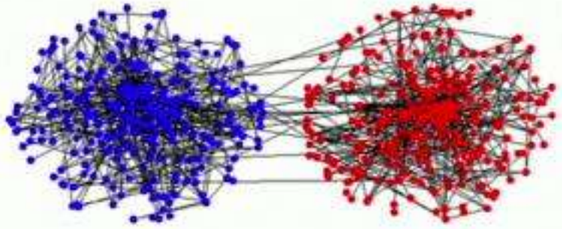
Motivating example 1

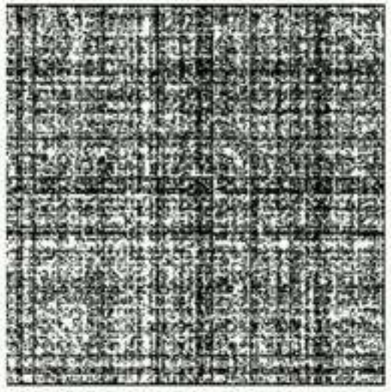
Clustering the stochastic block model

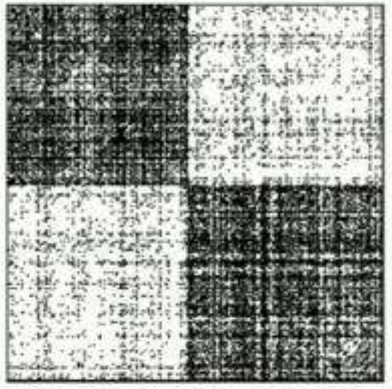
$A \sim SBM(p, q, n, 2)$ (two equal-sized communities):

$$\mathbb{P}(A_{ij} = 1) = \begin{cases} p & \text{if } i, j \text{ in the same community} \\ q & \text{if } i, j \text{ in different communities} \end{cases}$$





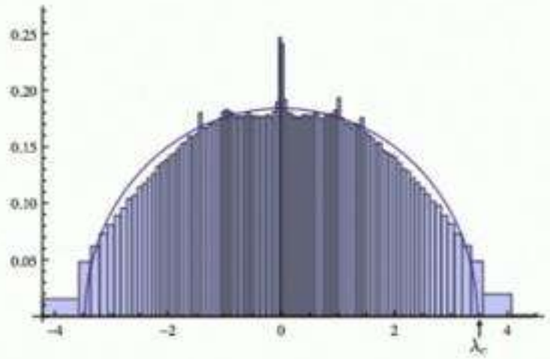




Clustering the stochastic block model

$A \sim SBM(a/n, b/n, n, 2)$ sparse.
Statistical threshold for detection: $(a-b)^2 > 2(a+b)$.

Spectrum doesn't concentrate (high degree vertices dominate it)
Laplacian is not useful for clustering



Other methods succeed. Example: semidefinite programming.

Krzakala, Moore, Mossel, Neeman, Sly, Zdeborová, Zhang, 2013

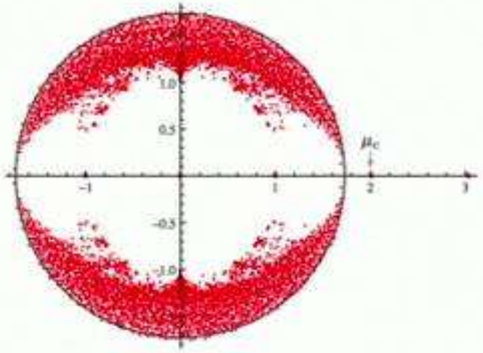
Deshpande, Abbe, Montanari, 2014

Abbe, Bandeira, Hall, 2014

Spectral redemption

Consider the non-backtracking operator (from linearized BP)

$$B_{(i\to j)(i'\to j')} = \begin{cases} 1 \text{ if } j = i' \text{ and } j' \neq i \\ 0 \text{ otherwise} \end{cases}$$



Second eigenvector of B reveals clustering structure

Krzakala, Moore, Mossel, Neeman, Sly, Zdeborová, Zhang, 2013

Bordenave, Lelarge, Massoulie, 2015

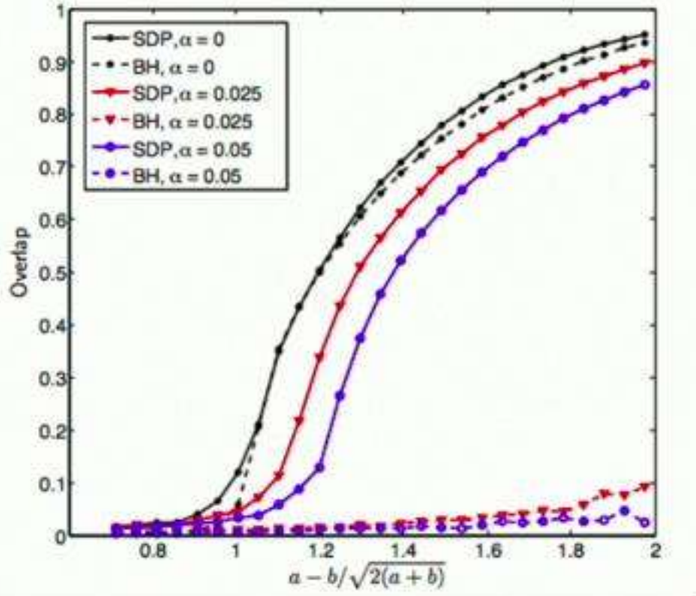
Bethe Hessian

$$BH(r) = (r^2 - 1)I - rA + D$$

Fixed points of BP $\longleftrightarrow$ Stationary points of Bethe free energy

Second eigenvector reveals clustering structure

Pitfall: highly dependent in the model.



Saade, Krzakala, Zdeborová, 2014

Javanmard, Montanari, Ricci-Tersenghi, 2015

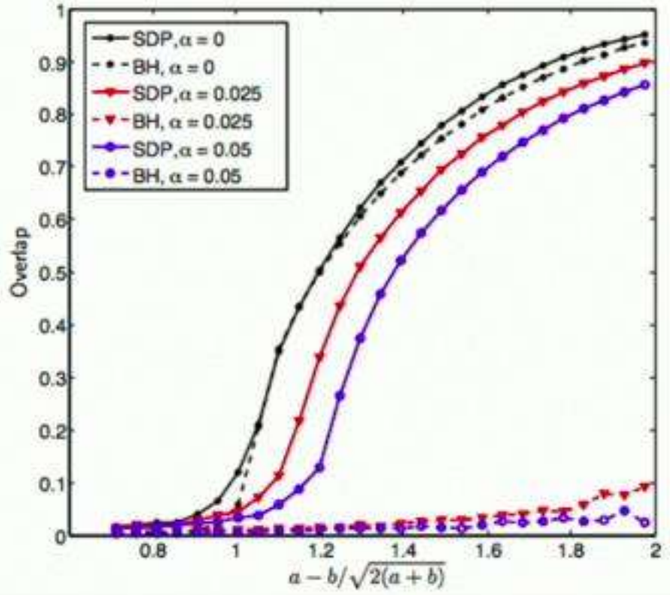
Bethe Hessian

$$BH(r) = (r^2 - 1)I - rA + D$$

Fixed points of BP $\longleftrightarrow$ Stationary points of Bethe free energy

Second eigenvector reveals clustering structure

Pitfall: highly dependent in the model.



Saade, Krzakala, Zdeborová, 2014

Javanmard, Montanari, Ricci-Tersenghi, 2015

Goal: Combine graph operators $I, D, A, \ldots$ to generate robust "data-driven spectral methods" for problems in graphs

Graph neural networks

sGNN($\mathcal{M}$)

Power method: $v^{t+1} = Mv^t \qquad t = 1, \ldots, T.$

Scarselli, Tsoi, Hagenbuchner, Monfardini, 2009

Chen, Li, Bruna, 2017

Graph neural networks

sGNN($\mathcal{M}$)

Power method: $v^{t+1} = Mv^t \qquad t = 1, \dots, T.$

Graph with adjacency matrix A. Set $\mathcal{M} = \{I_n, D, A, A^2, \dots, A^{2^J}\}$,

$$v^{t+1} = \left(\sum_{M \in \mathcal{M}} M v^t \theta_M \right),$$

with $v^t \in \mathbb{R}^{n \times d_t}$,
$\Theta = \{\theta_1^t, \dots, \theta_{|\mathcal{M}|}^t\}_t$, $\theta_M^t \in \mathbb{R}^{d_t \times d_{t+1}}$ trainable parameters.

Scarselli, Tsoi, Hagenbuchner, Monfardini, 2009

Chen, Li, Bruna, 2017

Graph neural networks

sGNN($\mathcal{M}$)

Power method: $v^{t+1} = Mv^t \qquad t = 1, \dots, T.$

Graph with adjacency matrix A. Set $\mathcal{M} = \{I_n, D, A, A^2, \dots, A^{2^J}\}$,

$$v_l^{t+1} = \left(\sum_{M \in \mathcal{M}} Mv^t \theta_{M,l} \right), \; l = 1, \dots, d_{t+1}$$

with $v^t \in \mathbb{R}^{n \times d_t}$,
$\Theta = \{\theta_1^t, \dots, \theta_{|\mathcal{M}|}^t\}_t$, $\theta_M^t \in \mathbb{R}^{d_t \times d_{t+1}}$ trainable parameters.

Scarselli, Tsoi, Hagenbuchner, Monfardini, 2009

Chen, Li, Bruna, 2017

Graph neural networks

sGNN($\mathcal{M}$)

Power method: $v^{t+1} = Mv^t \qquad t = 1, \dots, T.$

Graph with adjacency matrix A. Set $\mathcal{M} = \{I_n, D, A, A^2, \dots, A^{2^J}\}$,

$$v_l^{t+1} = \left(\sum_{M \in \mathcal{M}} M v^t \theta_{M,l}{}^t \right) , l = 1, \dots, d_{t+1}$$

with $v^t \in \mathbb{R}^{n \times d_t}$,
$\Theta = \{\theta_1^t, \dots, \theta_{|\mathcal{M}|}^t\}_t$, $\theta_M^t \in \mathbb{R}^{d_t \times d_{t+1}}$ trainable parameters.

Scarselli, Tsoi, Hagenbuchner, Monfardini, 2009

Chen, Li, Bruna, 2017

Graph neural networks

sGNN($\mathcal{M}$)

Power method: $v^{t+1} = Mv^t \qquad t = 1, \dots, T.$

Graph with adjacency matrix A. Set $\mathcal{M} = \{I_n, D, A, A^2, \dots, A^{2^J}\}$,

$$v_l^{t+1} = \rho\left(\sum_{M \in \mathcal{M}} Mv^t \theta_{M,l}{}^t\right) \quad , l = 1, \dots, d_{t+1}$$

with $v^t \in \mathbb{R}^{n \times d_t}$,
$\Theta = \{\theta_1^t, \dots, \theta_{|\mathcal{M}|}^t\}_t$, $\theta_M^t \in \mathbb{R}^{d_t \times d_{t+1}}$ trainable parameters.

Scarselli, Tsoi, Hagenbuchner, Monfardini, 2009

Chen, Li, Bruna, 2017

Graph neural networks

sGNN($\mathcal{M}$)

Power method: $v^{t+1} = Mv^t \qquad t = 1, \dots, T.$

Graph with adjacency matrix A. Set $\mathcal{M} = \{I_n, D, A, A^2, \dots, A^{2^J}\}$,

$$v_l^{t+1} = \rho\left(\sum_{M \in \mathcal{M}} Mv^t \theta_{M,l}{}^t\right) \ , l = 1, \dots, d_{t+1}$$

with $v^t \in \mathbb{R}^{n \times d_t}$,
$\Theta = \{\theta_1^t, \dots, \theta_{|\mathcal{M}|}^t\}_t$, $\theta_M^t \in \mathbb{R}^{d_t \times d_{t+1}}$ trainable parameters.

- Extension to line graph (GNN with non-backtracking).
- Extension to power graph $\min(1, A^t)$

Scarselli, Tsoi, Hagenbuchner, Monfardini, 2009

Chen, Li, Bruna, 2017

Graph neural networks

sGNN($\mathcal{M}$)

Power method: $v^{t+1} = Mv^t \qquad t = 1, \dots, T.$

Graph with adjacency matrix A. Set $\mathcal{M} = \{I_n, D, A, A^2, \dots, A^{2^J}\}$,

$$v_l^{t+1} = \rho\left(\sum_{M \in \mathcal{M}} Mv^t \theta_{M,l}{}^t\right) \ , l = 1, \dots, d_{t+1}$$

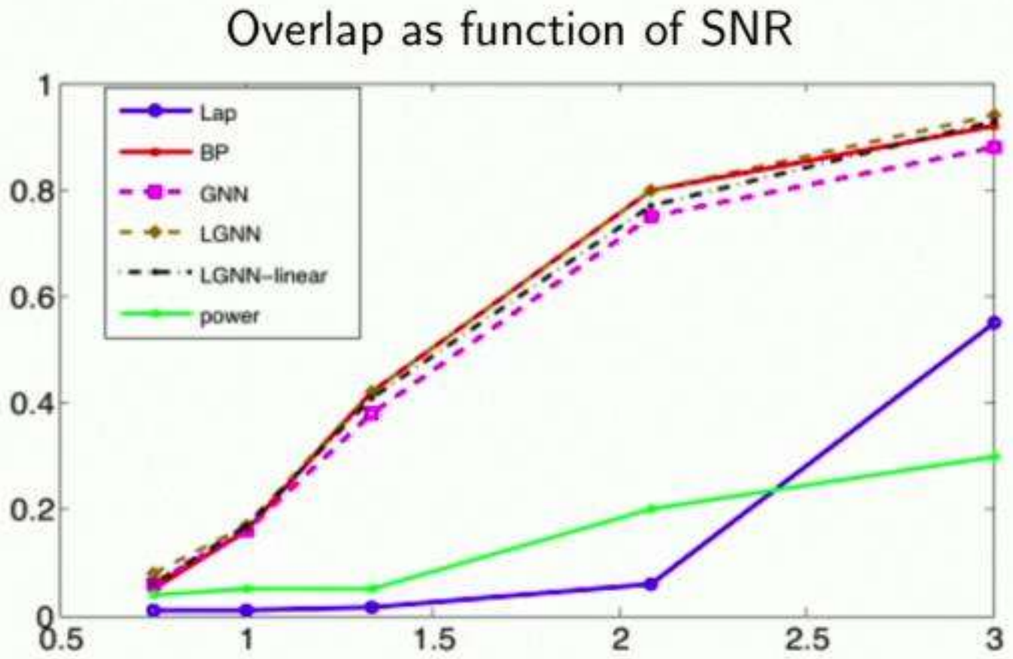
with $v^t \in \mathbb{R}^{n \times d_t}$,
$\Theta = \{\theta_1^t, \dots, \theta_{|\mathcal{M}|}^t\}_t$, $\theta_M^t \in \mathbb{R}^{d_t \times d_{t+1}}$ trainable parameters.

- Extension to line graph (GNN with non-backtracking).
- Extension to power graph $\min(1, A^t)$
- Equivariant wrt permutations $G \mapsto \phi(G)$ then $G_\Pi \mapsto \Pi\phi(G)$.

Scarselli, Tsoi, Hagenbuchner, Monfardini, 2009

Chen, Li, Bruna, 2017

Numerical performance. SBM $k = 2$

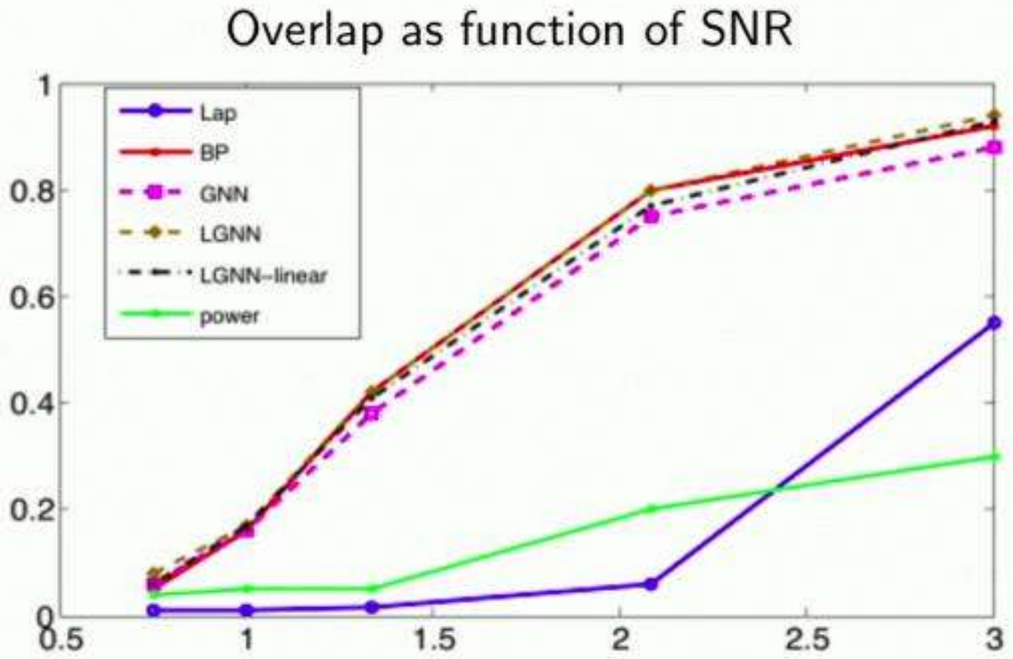


Theoretical result: Under simplifications one can show that all local minima have small loss.

Chen, Li, Bruna, 2017

Yao, V., Bandeira, 2019

Numerical performance. SBM $k = 2$



Theoretical result: Under simplifications one can show that all local minima have small loss.

Extension to unsupervised setting: Max-cut on random regular graphs.

Chen, Li, Bruna, 2017

Yao, V., Bandeira, 2019

Motivating example 2

Quadratic assignment problem

A, B $n \times n$ matrices. Π : set of $n \times n$ permutation matrices.

$$\text{Quadratic assignment} : \max_{X \in \Pi} \text{Trace}(AXBX^\top)$$

Motivating example 2

Quadratic assignment problem

A, B $n \times n$ matrices. Π : set of $n \times n$ permutation matrices.

$$\text{Quadratic assignment} : \max_{X \in \Pi} \text{Trace}(AXBX^\top)$$

It includes many relevant problem as particular cases:

- Graph matching: $\min_{X \in \Pi} \|AX - XB\|_F^2$

Motivating example 2

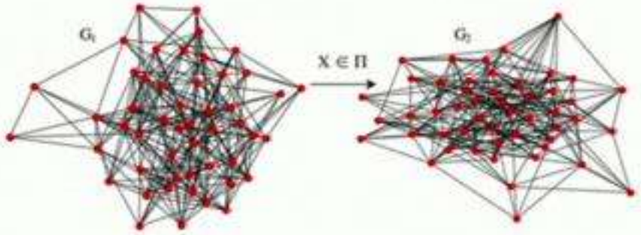
Quadratic assignment problem

A, B $n \times n$ matrices. Π : set of $n \times n$ permutation matrices.

$$\text{Quadratic assignment} : \max_{X \in \Pi} \text{Trace}(AXBX^\top)$$

It includes many relevant problem as particular cases:

- Graph matching: $\min_{X \in \Pi} \|AX - XB\|_F^2 = \|AX\|^2 + \|XB\|^2 - 2\langle AX, XB \rangle$



Motivating example 2

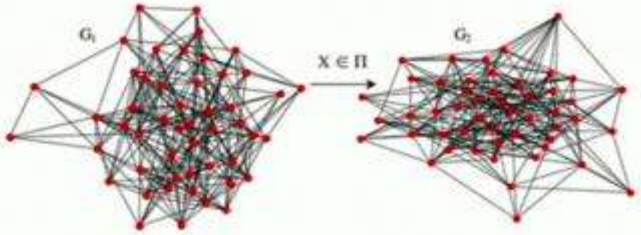
Quadratic assignment problem

A, B $n \times n$ matrices. Π : set of $n \times n$ permutation matrices.

$$\text{Quadratic assignment} : \max_{X \in \Pi} \text{Trace}(AXBX^\top)$$

It includes many relevant problem as particular cases:

- Graph matching: $\min_{X \in \Pi} \|AX - XB\|_F^2 = \|AX\|^2 + \|XB\|^2 - 2\langle AX, XB \rangle$



- Graph isomorphism

Motivating example 2

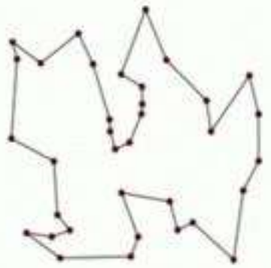
Quadratic assignment problem

A, B $n \times n$ matrices. Π : set of $n \times n$ permutation matrices.

$$\text{Quadratic assignment} : \max_{X \in \Pi} \text{Trace}(AXBX^\top)$$

It includes many relevant problem as particular cases:

- Graph matching: $\min_{X \in \Pi} \|AX - XB\|_F^2$.
- Graph isomorphism.
- Traveling salesman problem.



Motivating example 2

Quadratic assignment problem

A, B $n \times n$ matrices. Π : set of $n \times n$ permutation matrices.

$$\text{Quadratic assignment} : \max_{X \in \Pi} \text{Trace}(AXBX^\top)$$

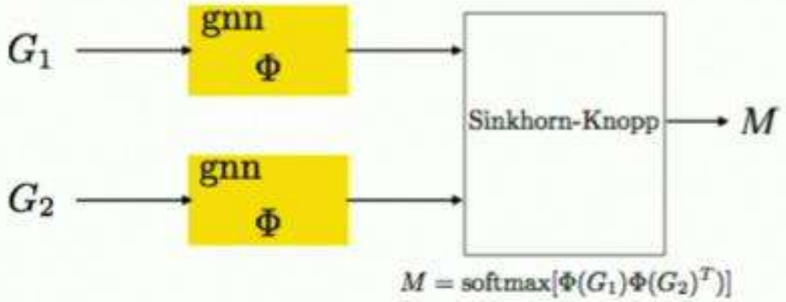
It includes many relevant problem as particular cases:

- Graph matching: $\min_{X \in \Pi} \|AX - XB\|_F^2$.
- Graph isomorphism.
- Traveling salesman problem.
- Gromov-Hausdorff distance of finite metric spaces.

It is NP-hard, even to approximate it.

GNN approach to quadratic assignment

Siamese neural network:

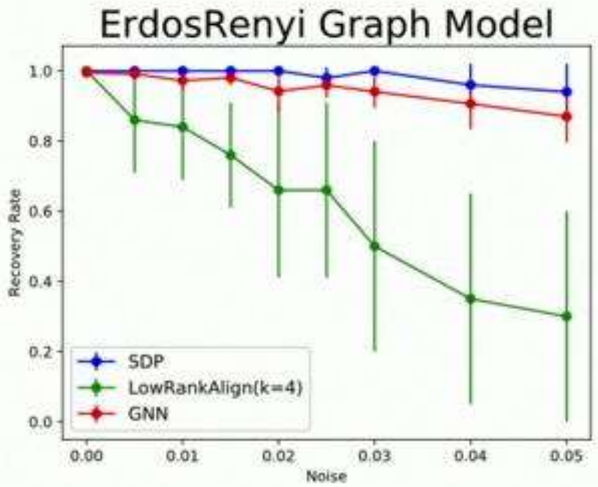


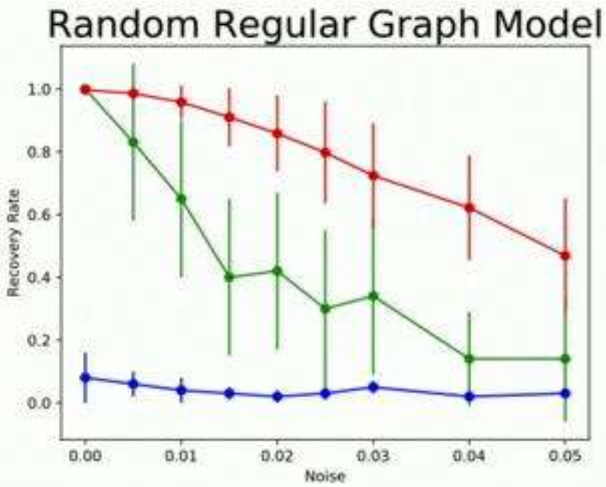
$G_2 = \pi \star G_1 \oplus N \qquad N \sim$ i.i.d. bit flip

$G_1 \sim$ Erdos-Renyi

$G_1 \sim$ Random regular

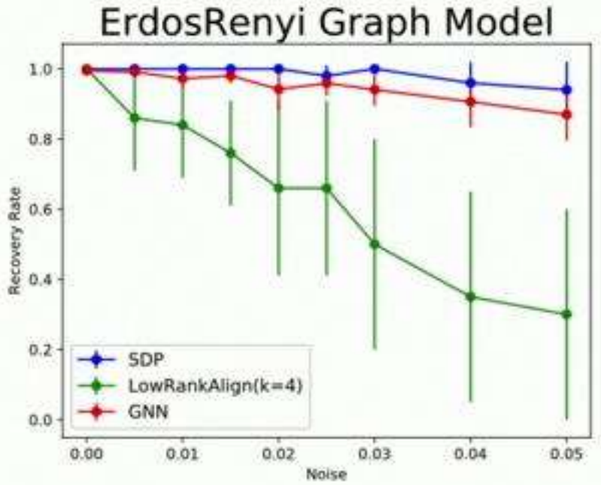
Numerical experiments

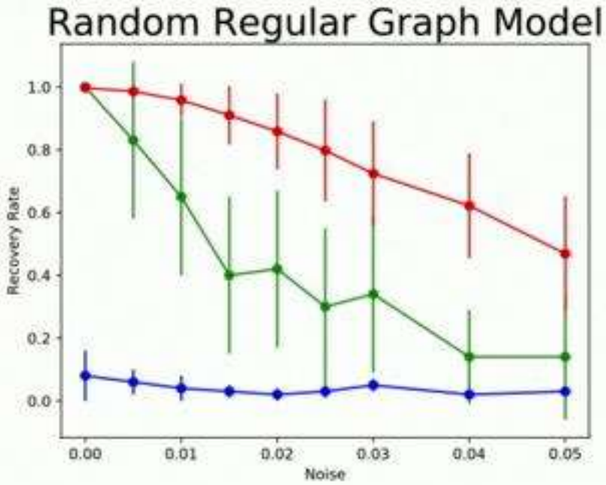


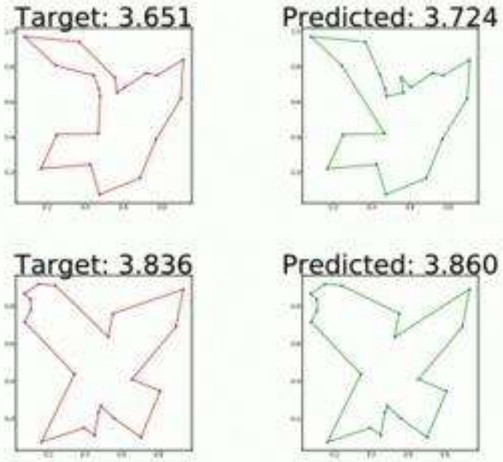


Nowak, V., Bandeira, Bruna, 2017

Numerical experiments







Nowak, V., Bandeira, Bruna, 2017

Other GNN formulation

Message passing neural network

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\{h_u^{(k-1)} : u \in \mathcal{N}(u)\}\right)$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right)$$

Hamilton, Ying, Leskovec, 2017

Other GNN formulation

Message passing neural network

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\{h_u^{(k-1)} : u \in \mathcal{N}(u)\}\right)$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right)$$

- There exist many formulations of GNN
- All satisfy one essential property:
  - invariance or equivariance with respect to permutations
  - node labels are not intrinsic

Hamilton, Ying, Leskovec, 2017

How powerful are graph neural networks?

Q: How good are they at distinguishing non-isomorphic graphs?

Xu, Hu, Leskovec, Jekelga 2019

How powerful are graph neural networks?

Q: How good are they at distinguishing non-isomorphic graphs?

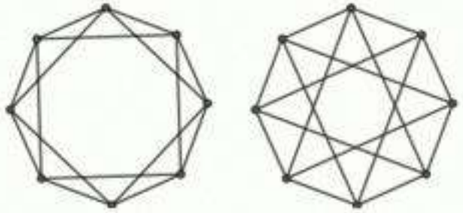
A: MPNN can be as powerful as the Weisfeler-Lehman test (1968).
W-L test is as powerful as the LP relaxation (Ullman et al 1994).

How powerful are graph neural networks?

Q: How good are they at distinguishing non-isomorphic graphs?

A: MPNN can be as powerful as the Weisfeler-Lehman test (1968).
W-L test is as powerful as the LP relaxation (Ullman et al 1994).

In particular MPNN cannot distinguish between non-isomorphic regular graphs with the same degree.



Xu, Hu, Leskovec, Jekelga 2019

Invariant functions on graphs

- Linear case:
  - If $L : \mathbb{R}^{n^k} \to \mathbb{R}$ invariant, then $vec(L) = \pi^{\otimes k} vec(L)$.
  - If $L : \mathbb{R}^{n^k} \to \mathbb{R}^{n^k}$ equivariant, then $vec(L) = \pi^{\otimes 2k} vec(L)$

  - The space of invariant [equivariant] linear functions on k-tensors has dimension $b(k)$ $[b(2k)]$.
    ($b(k)$ denotes Bell Number: number of partitions of a size k set).

Maron, Ben-Hamu, Shamir, Lipman, 2019

Maron, Fetaya, Segol, Lipman, 2019

Keriven, Peyré, 2019

Invariant functions on graphs

- Linear case:
  - If $L : \mathbb{R}^{n^k} \to \mathbb{R}$ invariant, then $vec(L) = \pi^{\otimes k} vec(L)$.
  - If $L : \mathbb{R}^{n^k} \to \mathbb{R}^{n^k}$ equivariant, then $vec(L) = \pi^{\otimes 2k} vec(L)$
  - The space of invariant [equivariant] linear functions on k-tensors has dimension $b(k)$ $[b(2k)]$.
    ($b(k)$ denotes Bell Number: number of partitions of a size k set).
- Universal approximation:
  - Invariant networks constructed by composition of linear invariant layers $L_t : \mathbb{R}^{n^k \times a} \to \mathbb{R}^b$ with ReLU or sigmoid activation functions universally approximate the space of invariant functions.
  - Extension to equivariant functions.

Maron, Ben-Hamu, Shamir, Lipman, 2019

Maron, Fetaya, Segol, Lipman, 2019

Keriven, Peyré, 2019

Invariant functions on graphs

- Linear case:
  - If $L : \mathbb{R}^{n^k} \to \mathbb{R}$ invariant, then $vec(L) = \pi^{\otimes k} vec(L)$.
  - If $L : \mathbb{R}^{n^k} \to \mathbb{R}^{n^k}$ equivariant, then $vec(L) = \pi^{\otimes 2k} vec(L)$
  - The space of invariant [equivariant] linear functions on k-tensors has dimension $b(k)$ $[b(2k)]$.
    ($b(k)$ denotes Bell Number: number of partitions of a size k set).
- Universal approximation:
  - Invariant networks constructed by composition of linear invariant layers $L_t : \mathbb{R}^{n^k \times a} \to \mathbb{R}^b$ with ReLU or sigmoid activation functions universally approximate the space of invariant functions.
  - Extension to equivariant functions.

    Arbitrary high order tensors are needed.

    Rates of convergence are not known.

Maron, Ben-Hamu, Shamir, Lipman, 2019

Maron, Fetaya, Segol, Lipman, 2019

Keriven, Peyré, 2019

Invariant functions on graphs

- Linear case:
  - If $L : \mathbb{R}^{n^k} \to \mathbb{R}$ invariant, then $vec(L) = \pi^{\otimes k} vec(L)$.
  - If $L : \mathbb{R}^{n^k} \to \mathbb{R}^{n^k}$ equivariant, then $vec(L) = \pi^{\otimes 2k} vec(L)$
  - The space of invariant [equivariant] linear functions on k-tensors has dimension $b(k)$ $[b(2k)]$.
    ($b(k)$ denotes Bell Number: number of partitions of a size k set).
- Universal approximation:
  - Invariant networks constructed by composition of linear invariant layers $L_t : \mathbb{R}^{n^k \times a} \to \mathbb{R}^b$ with ReLU or sigmoid activation functions universally approximate the space of invariant functions.
  - Extension to equivariant functions.

    Arbitrary high order tensors are needed.

    Rates of convergence are not known.

Maron, Ben-Hamu, Shamir, Lipman, 2019

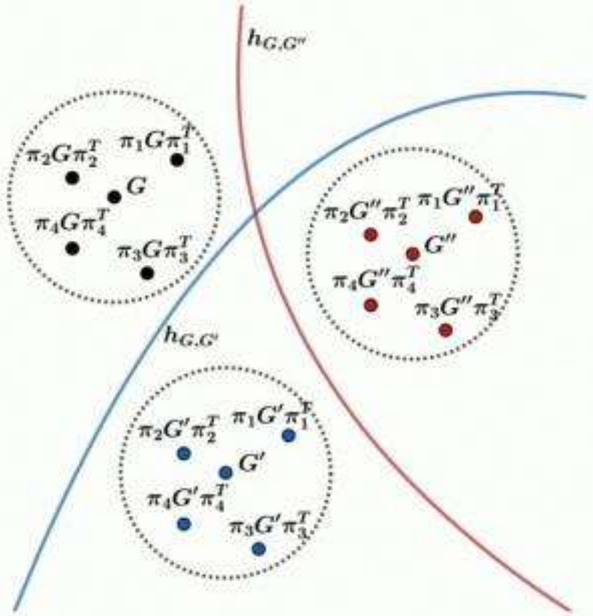
Maron, Fetaya, Segol, Lipman, 2019

Keriven, Peyré, 2019

Graph isomorphism equivalence to universal approximation

GIso-discriminating class of functions

A class $\mathcal{C}$ of permutation-invariant functions from $\mathcal{X}^{n\times n}$ to $\mathbb{R}$ so that for all pairs $G_1 \not\simeq G_2 \in \mathcal{X}^{n\times n}$, there exists $h \in \mathcal{C}$ such that $h(G_1) \neq h(G_2)$.



Chen, V., Chen, Bruna, 2019

Graph isomorphism equivalence to universal approximation

Universally approximating

A class $\mathcal{C}$ of permutation-invariant functions from $\mathcal{X}^{n\times n}$ to $\mathbb{R}$ so that for all permutation-invariant function f from $\mathcal{X}^{n\times n}$ to $\mathbb{R}$, and for all $\epsilon > 0$, there exists $h_{f,\epsilon} \in \mathcal{C}$ such that
$\|f - h_{f,\epsilon}\|_\infty := \sup_{G\in\mathcal{X}^{n\times n}} |f(G) - h_{f,\epsilon}(G)| < \epsilon$

Remark

Universally approximating classes of functions are also GIso-discriminating.

Chen, V., Chen, Bruna, 2019

Graph isomorphism equivalence to universal approximation

$\mathcal{C}^{+L}$

If $\mathcal{C}$ is a collection of functions from $\mathcal{X}^{n\times n}$ to $\mathbb{R}$, consider the set of functions from graphs G to $\mathcal{NN}([h_1(G), ..., h_d(G)])$ for some finite d and $h_1, ..., h_d \in \mathcal{C}$, where $\mathcal{NN}$ is a feed-forward neural network with ReLU and L layers.

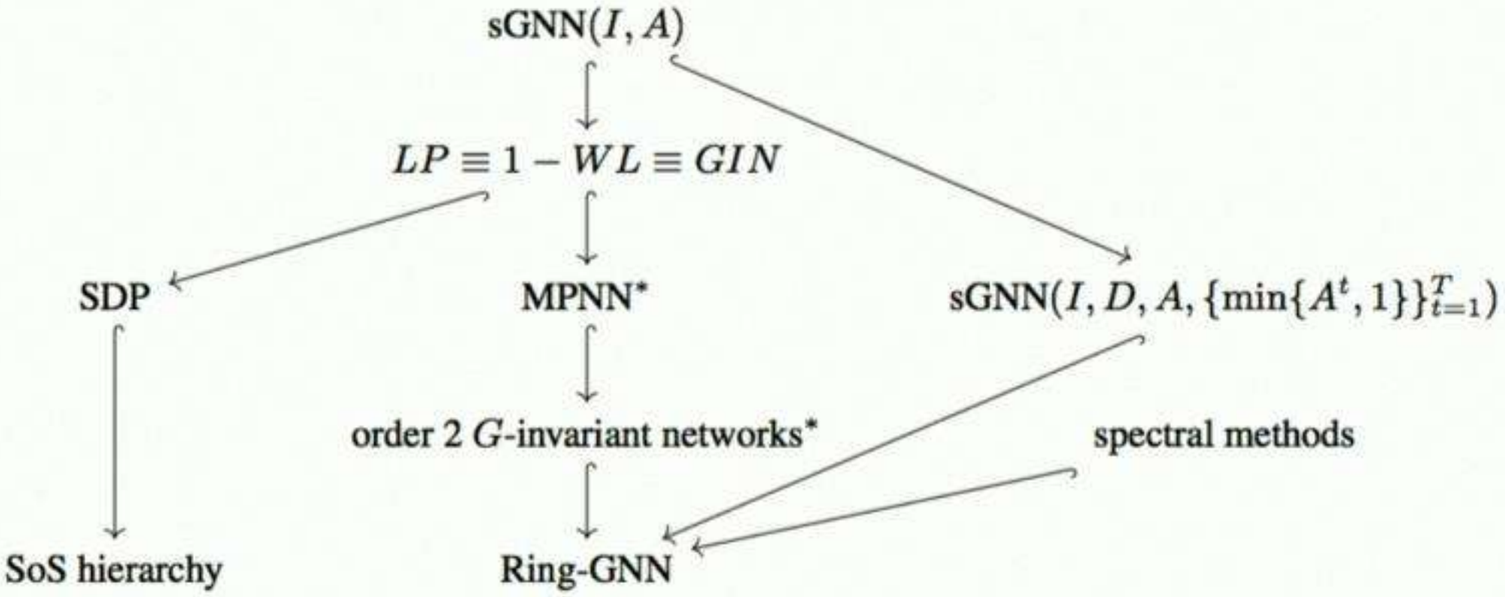
Theorem

If $\mathcal{C}$ is GIso-discriminating $\mathcal{C}^{+2}$ is universally approximating.

Chen, V., Chen, Bruna, 2019

Comparison of classes of functions through GIso

$\mathcal{C} \subseteq \mathcal{C}'$ if for all pairs of non-isomorphic graphs G_1, G_2, if there exists $h \in \mathcal{C}$ so that $h(G_1) \neq h(G_2)$ then there exists $h' \in \mathcal{C}'$ so that $h'(G_1) \neq h'(G_2)$.



Ring GNN

Input: Graph with n nodes and d features: $A \in \mathbb{R}^{n \times n \times d}$.

Equivariant linear layer from $\mathbb{R}^{n \times n \times d}$ to $\mathbb{R}^{n \times n \times d'}$. For $\theta \in \mathbb{R}^{d \times d' \times 17}$:
$L_\theta(A)_{\cdot,\cdot,k'} = \sum_{k=1}^{d} \sum_{i=1}^{15} \theta_{k,k',i} L_i(A_{\cdot,\cdot,i}) + \sum_{i=16}^{17} \theta_{k,k',i} \overline{L}_i$.
Set $A^{(0)} = A$.

$$\begin{aligned} B_1^{(t)} &= \rho(L_{\alpha^{(t)}}(A^{(t)})) \\ B_2^{(t)} &= \rho(L_{\beta^{(t)}}(A^{(t)}) \cdot L_{\gamma^{(t)}}(A^{(t)})) \\ A^{(t+1)} &= k_1^{(t)} B_1^{(t)} + k_2^{(t)} B_2^{(t)} \end{aligned}$$

where $k_1^{(t)}, k_2^{(t)} \in \mathbb{R}$, $\alpha^{(t)}, \beta^{(t)}, \gamma^{(t)} \in \mathbb{R}^{d^{(t)} \times d'^{(t)} \times 17}$ are learnable parameters.

Scalar output: $\theta_S \sum_{i,j} A_{ij}^{(T)} + \theta_D \sum_{i,i} A_{ii}^{(T)} + \sum_i \theta_i \lambda_i(A^{(T)})$, where $\theta_S, \theta_D, \theta_1, \ldots, \theta_n \in \mathbb{R}$ are trainable parameters, and $\lambda_i(A^{(T)})$ is the i-th eigenvalue of $A^{(T)}$.

Extensions - Future work

- Explicit rates:
  Connect GNN depth/architecture with classes of graphs they separate.

- Optimization landscape of GNNs:
  Current analysis of optimization landscape relies in simplified models to show that all local minima are confined in low-energy configurations.

- Connection with SoS:
  For some classes of "detecting hidden structures problems" existence of degree-d SoS refutations implies success of certain (typically non-explicit) spectral methods.
  - Can we express such class of spectral methods with GNNs.
  - Can we learn them?

Hopkins, Kothari, Potechin, Raghavendra, Schramm, Steurer, 2017

References

Supervised Community Detection with Hierarchical Graph Neural Networks
Z. Chen, L. Li, J. Bruna, ICLR 2019
arXiv:1705.08415

A note on learning algorithms for quadratic assignment with Graph Neural Networks
A. Nowak, S. Villar, A. S. Bandeira, J. Bruna, ICML (PADL) 2017
arXiv:1706.07450

On the equivalence between graph isomorphism testing and function approximation with GNNs
Z. Chen, S. Villar, L. Chen, J. Bruna
arXiv:1905.12560

SIMONS FOUNDATION