Regularization in the Age of Machine Learning

Gabriel Clara

Department of Applied Mathematics
University of Twente

June 25, 2025

About Me

- PhD student at UTwente (almost done!)

About Me

- PhD student at UTwente (almost done!)
- Research focus: mathematical foundations of machine learning methods

- PhD student at UTwente (almost done!)
- Research focus: mathematical foundations of machine learning methods
- My bosses: Sophie Langer (RU Bochum) and Johannes Schmidt-Hieber (UTwente)

A Supervised Learning Example

- Observe data $\mathbf{X}_i \in \mathbb{R}^{d_x}$, $i = 1, \dots, n$ with labels $\mathbf{Y}_i \in \mathbb{R}^{d_y}$

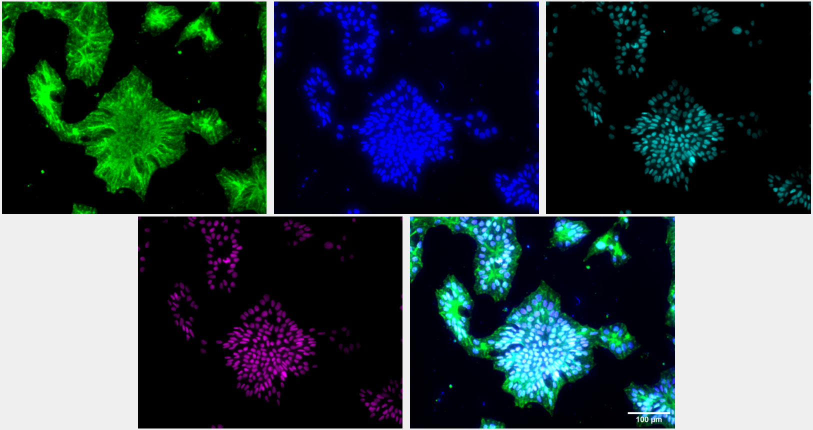
A Supervised Learning Example

- Observe data $\mathbf{X}_i \in \mathbb{R}^{d_x}$, $i = 1, \ldots, n$ with labels $\mathbf{Y}_i \in \mathbb{R}^{d_y}$
- Want to predict labels on previously unseen and unlabeled data

A Supervised Learning Example

- Observe data $\mathbf{X}_i \in \mathbb{R}^{d_x}$, $i = 1, \ldots, n$ with labels $\mathbf{Y}_i \in \mathbb{R}^{d_y}$
- Want to predict labels on previously unseen and unlabeled data
- Example: $\mathbf{X}_i$ a medical scan and $\mathbf{Y}_i$ a corresponding diagnosis

A Supervised Learning Example



The Model Class:

- Want to learn functional relationship $\mathbf{Y}_i \approx f(\mathbf{X}_i)$
- Must choose class $\mathcal{F}$ of proposed functions f

The Three Ingredients of Regression I

The Model Class:

- Want to learn functional relationship $\mathbf{Y}_i \approx f(\mathbf{X}_i)$
- Must choose class $\mathcal{F}$ of proposed functions f
- Modern ML uses tremendously complex model classes

The Model Class:

- Want to learn functional relationship $\mathbf{Y}_i \approx f(\mathbf{X}_i)$
- Must choose class $\mathcal{F}$ of proposed functions f
- Modern ML uses tremendously complex model classes
- GPT-3: 175 billion trainable parameters[1]

[1] Brown, T. B. et al *Language Models are Few-Shot Learners* (2020)

A Simpler Model Class:

- Fix $L \geq 2$, rewrite $d_{L+1} = d_x$ and $d_0 = d_y$, and pick d_ℓ for each $\ell = 1, \dots L$

A Simpler Model Class:

- Fix $L \geq 2$, rewrite $d_{L+1} = d_x$ and $d_0 = d_y$, and pick d_ℓ for each $\ell = 1, \dots L$
- Pick $\sigma : \mathbb{R} \to \mathbb{R}$ and write $\sigma_{W_\ell, \mathbf{v}_\ell}(\mathbf{z}) = \sigma\big(W_\ell \mathbf{x} + \mathbf{v}_\ell\big)$, with $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ and $\mathbf{v}_\ell \in \mathbb{R}^{d_\ell}$

A Simpler Model Class:

- Fix $L \geq 2$, rewrite $d_{L+1} = d_x$ and $d_0 = d_y$, and pick d_ℓ for each $\ell = 1, \ldots L$
- Pick $\sigma : \mathbb{R} \to \mathbb{R}$ and write $\sigma_{W_\ell, \mathbf{v}_\ell}(\mathbf{z}) = \sigma\big(W_\ell \mathbf{x} + \mathbf{v}_\ell\big)$, with $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ and $\mathbf{v}_\ell \in \mathbb{R}^{d_\ell}$
- Neural network:

$$f(\mathbf{x}) = W_{L+1} \circ \sigma_{W_L, \mathbf{v}_L} \circ \cdots \circ \sigma_{W_1, \mathbf{v}_1}(\mathbf{x})$$

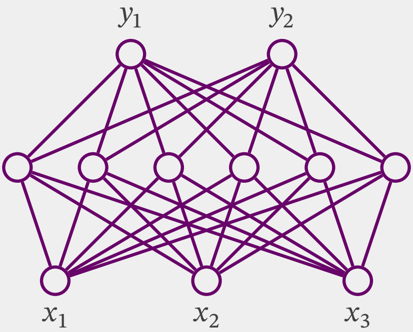
A Simpler Model Class:

- Fix $L \geq 2$, rewrite $d_{L+1} = d_x$ and $d_0 = d_y$, and pick d_ℓ for each $\ell = 1, \ldots L$
- Pick $\sigma : \mathbb{R} \to \mathbb{R}$ and write $\sigma_{W_\ell, \mathbf{v}_\ell}(\mathbf{z}) = \sigma\big(W_\ell \mathbf{x} + \mathbf{v}_\ell\big)$, with $W_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ and $\mathbf{v}_\ell \in \mathbb{R}^{d_\ell}$
- Neural network:

$$f(\mathbf{x}) = W_{L+1} \circ \sigma_{W_L, \mathbf{v}_L} \circ \cdots \circ \sigma_{W_1, \mathbf{v}_1}(\mathbf{x})$$

- Alternates affine transformations with (usually) non-linear function σ

Feed-Forward Neural Networks



Symbol	Terminology
L	Network Depth
d_0	Input Dimension/No. of Features
d_{L+1}	Output Dimension
σ	Activation Function
$\sigma_{W_\ell,\mathbf{v}_\ell}$	Hidden Layer
W_{L+1}	Output Layer
d_ℓ, $\ell = 1, \ldots, L$	Hidden Layer Widths

Why Choose Neural Networks as a Model Class?

- Neural networks with a non-polynomial activation function are dense in the space of continuous functions with respect to compact convergence.

Why Choose Neural Networks as a Model Class?

- Neural networks with a non-polynomial activation function are dense in the space of continuous functions with respect to compact convergence.[2]

[2] Leshno, M. et al *Multilayer Feedforward Networks with a Nonpolynomial Activation Function can Approximate any Function* (1993)

Why Choose Neural Networks as a Model Class?

- Neural networks with a non-polynomial activation function are dense in the space of continuous functions with respect to compact convergence.
- Can adapt to arbitrarily complex patterns in the data

The Empirical Risk:

- To pick the optimal f in our class, need to quantify predictive performance

The Empirical Risk:

- To pick the optimal f in our class, need to quantify predictive performance
- Let $\mathcal{L}_i(f)$ measure fit on i^{th} data point, for example $\mathcal{L}_i(f) = \|\mathbf{Y}_i - f(\mathbf{X}_i)\|_2^2$

The Empirical Risk:

- Let $\mathcal{L}_i(f)$ measure fit on i^{th} data point, for example $\mathcal{L}_i(f) = \|\mathbf{Y}_i - f(\mathbf{X}_i)\|_2^2$
- If $\mathbf{Y}_i = f(\mathbf{X}_i)$ for each i, then f minimizes the empirical risk

$$\widehat{\mathcal{L}}_n(f) = \frac{1}{n} \cdot \sum_{i=1}^{n} \mathcal{L}_i(f)$$

The Empirical Risk:

- Let $\mathcal{L}_i(f)$ measure fit on i^{th} data point, for example $\mathcal{L}_i(f) = \|\mathbf{Y}_i - f(\mathbf{X}_i)\|_2^2$
- If $\mathbf{Y}_i = f(\mathbf{X}_i)$ for each i, then f minimizes the empirical risk

$$\widehat{\mathcal{L}}_n(f) = \frac{1}{n} \cdot \sum_{i=1}^{n} \mathcal{L}_i(f)$$

- With (hypothetical) access to the whole data distribution μ, can compute the population risk

$$\mathcal{L}_\mu(f) = \int \mathcal{L}_s(f) \, \mathrm{d}\mu(s)$$

How do we Estimate the Optimal Parameters?

- Ideal model would satisfy $\mathcal{L}(f) = 0$, but we cannot access the population risk

How do we Estimate the Optimal Parameters?

- Ideal model would satisfy $\mathcal{L}(f) = 0$, but we cannot access the population risk
- Can only hope to compute empirical risk minimizer

$$\widehat{f} \in \arg\min_{f \in \mathcal{F}} \widehat{\mathcal{L}}_n(f)$$

How do we Estimate the Optimal Parameters?

- Ideal model would satisfy $\mathcal{L}(f) = 0$, but we cannot access the population risk
- Can only hope to compute empirical risk minimizer

$$\widehat{f} \in \arg\min_{f \in \mathcal{F}} \widehat{\mathcal{L}}_n(f)$$

- To achieve a robust estimate, must both minimize $\widehat{\mathcal{L}}_n$ (data fit) and the gap $\widehat{\mathcal{L}}_n - \mathcal{L}$ (generalization error)

Potential Problems:

- The empirical risk $\widehat{\mathcal{L}}_n$ may feature many local and global minima, not all of which generalize well

Exercise

Consider the linear regression loss

$$\boldsymbol{\beta} \mapsto \|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2$$

with $X \in \mathbb{R}^{n \times d}$ *having linearly independent columns and* $d \gg n$.

Exercise

Consider the linear regression loss

$$\boldsymbol{\beta} \mapsto \|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2$$

with $X \in \mathbb{R}^{n \times d}$ *having linearly independent columns and* $d \gg n$*. Are there any "bad" solutions to this problem?*

Potential Problems:

- The empirical risk $\widehat{\mathcal{L}}_n$ may feature many local and global minima, not all of which generalize well
- Must be careful when computing empirical risk minimizer $\widehat{f} \in \arg\min_{f\in\mathcal{F}} \widehat{\mathcal{L}}_n(f)$

The Training Algorithm:

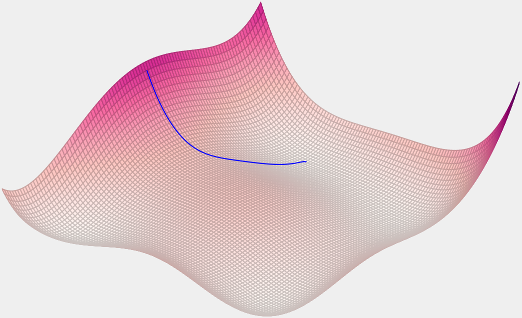
- Typically, cannot directly compute empirical risk minimizer, especially difficult if $\mathcal{F}$ is a class of neural networks

The Training Algorithm:

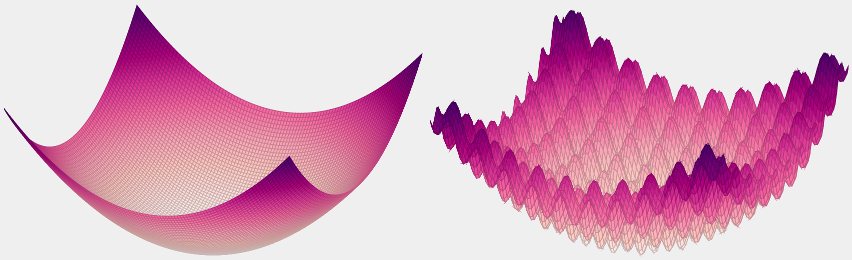
- Typically, cannot directly compute empirical risk minimizer, especially difficult if $\mathcal{F}$ is a class of neural networks
- Approximate iteratively: pick initial guesses $W_\ell(0)$ and $\mathbf{v}_\ell(0)$, then use gradient descent recursion

$$\begin{bmatrix} W_1(k+1) \\ \vdots \\ W_{L+1}(k+1) \\ \mathbf{v}_1(k+1) \\ \vdots \\ \mathbf{v}_L(k+1) \end{bmatrix} = \begin{bmatrix} W_1(k) \\ \vdots \\ W_{L+1}(k) \\ \mathbf{v}_1(k) \\ \vdots \\ \mathbf{v}_L(k) \end{bmatrix} - \alpha_k \cdot \begin{bmatrix} \nabla_{W_1(k)} \widehat{\mathcal{L}}_n(f) \\ \vdots \\ \nabla_{W_{L+1}(k)} \widehat{\mathcal{L}}_n(f) \\ \nabla_{\mathbf{v}_1(k)} \widehat{\mathcal{L}}_n(f) \\ \vdots \\ \nabla_{\mathbf{v}_L(k)} \widehat{\mathcal{L}}_n(f) \end{bmatrix}$$

The Three Ingredients of Regression III



The Three Ingredients of Regression III



What does this mean for us?

- Despite all the potential problems, gradient descent works well in practice

What does this mean for us?

- Despite all the potential problems, gradient descent works well in practice
- Example: GD in over-parametrized linear regression yields norm-minimal solutions

What does this mean for us?

- Despite all the potential problems, gradient descent works well in practice
- Example: GD in over-parametrized linear regression yields norm-minimal solutions
- Regularization implicit to the choices made during training may explain how models generalize

How does Randomness Enter the Picture?

How does Randomness Enter the Picture?

- Stochastic gradient descent:

$$W_\ell(k+1) = W_\ell(k) - \alpha_k \cdot \nabla_{W_\ell(k)} \mathcal{L}_{i_k}(f), \qquad \ell = 1, \ldots, L$$

with $i_k \sim \mathrm{Unif}(1, \ldots, n)$

How does Randomness Enter the Picture?

- Stochastic gradient descent:

$$W_\ell(k+1) = W_\ell(k) - \alpha_k \cdot \nabla_{W_\ell(k)} \mathcal{L}_{i_k}(f), \qquad \ell = 1, \dots, L$$

  with $i_k \sim \mathrm{Unif}(1, \dots, n)$
- Speeds up gradient computation

How does Randomness Enter the Picture?

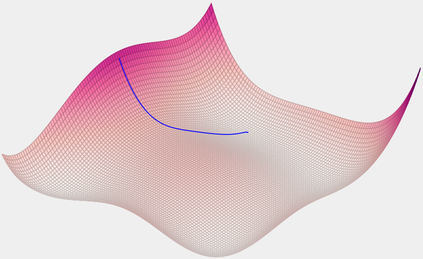
- Stochastic gradient descent:

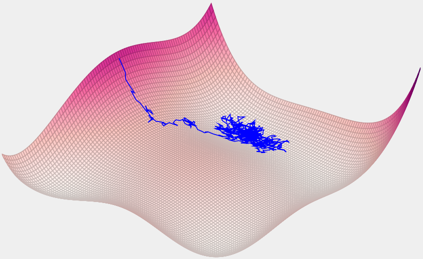
$$W_\ell(k+1) = W_\ell(k) - \alpha_k \cdot \nabla_{W_\ell(k)} \mathcal{L}_{i_k}(f), \qquad \ell = 1, \dots, L$$

  with $i_k \sim \mathrm{Unif}(1, \dots, n)$
- Speeds up gradient computation
- Can help escape sub-optimal minima[2]

[2] Ibayashi, H. et al *Why does SGD Prefer Flat Minima?* (2023)

Algorithmic Randomness





General Stochastic Approximation:

- SGD is an instance of the general algorithm

$$W_\ell(k+1) = W_\ell(k) - \alpha_k \cdot \nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f), \qquad \ell = 1, \dots, L$$

with $\widetilde{\mathcal{L}}_k \sim \widetilde{\mathcal{L}}$ a sample from a random function $f \mapsto \widetilde{\mathcal{L}}(f)$

General Stochastic Approximation:

- SGD is an instance of the general algorithm

$$W_\ell(k+1) = W_\ell(k) - \alpha_k \cdot \nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f), \qquad \ell = 1, \dots, L$$

  with $\widetilde{\mathcal{L}}_k \sim \widetilde{\mathcal{L}}$ a sample from a random function $f \mapsto \widetilde{\mathcal{L}}(f)$
- In general, iterates converge to distribution concentrated near critical points of

$$f \mapsto \mathbb{E}\big[\widetilde{\mathcal{L}}(f)\big]$$

  with step-sizes α_k determining the variance[2]

[2] Robbins, H. et al *A Stochastic Approximation Method* (1951)

General Stochastic Approximation:

- SGD is an instance of the general algorithm

$$W_\ell(k+1) = W_\ell(k) - \alpha_k \cdot \nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f), \qquad \ell = 1, \dots, L$$

  with $\widetilde{\mathcal{L}}_k \sim \widetilde{\mathcal{L}}$ a sample from a random function $f \mapsto \widetilde{\mathcal{L}}(f)$

- In general, iterates converge to distribution concentrated near critical points of

$$f \mapsto \mathbb{E}\big[\widetilde{\mathcal{L}}(f)\big]$$

  with step-sizes α_k determining the variance

- What if we choose noise such that

$$\mathbb{E}\big[\nabla \widetilde{\mathcal{L}}(f)\big] \neq \nabla \widehat{\mathcal{L}}_n(f)$$

  and why would we do so?

Example: Dropout

- During training neurons may correlate with each other and lose expressiveness

Example: Dropout

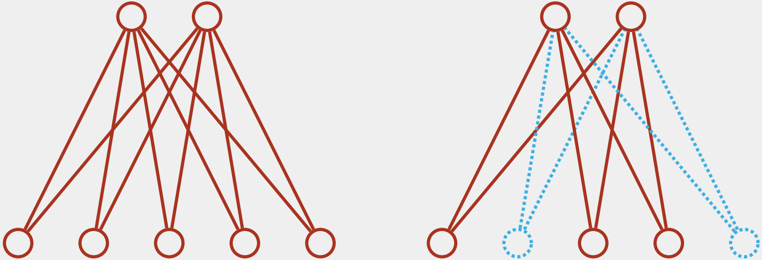
- During training neurons may correlate with each other and lose expressiveness
- To help, may randomly omit connections from the network during training

Example: Dropout

- During training neurons may correlate with each other and lose expressiveness
- To help, may randomly omit connections from the network during training[2]

[2] Srivastava, N. et al *Dropout: A Simple Way to Prevent Neural Networks from Overfitting* (2014)

Noisy Algorithmic Regularization Methods



Example: Dropout

- During training neurons may correlate with each other and lose expressiveness
- To help, may randomly omit connections from the network during training
- Randomized loss $\widetilde{\mathcal{L}}(f) = \widehat{\mathcal{L}}_n(\widetilde{f})$ with $\widetilde{f}$ having randomly deleted connections

Example: Stochastic Sharpness-Aware Minimization:

- Flat regions of the empirical risk are thought to generalize well

Example: Stochastic Sharpness-Aware Minimization:

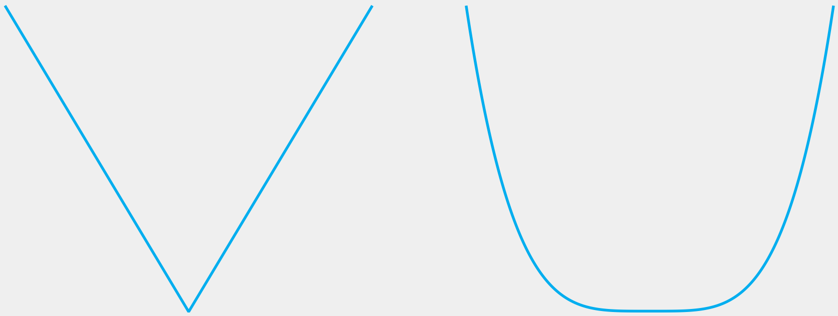
- Flat regions of the empirical risk are thought to generalize well[2]

[2] Hochreiter, S. et al *Simplifying Neural Nets by Discovering Flat Minima* (1994) Foret, P. et al *Sharpness-Aware Minimization for Efficiently Improving Generalization* (2021)

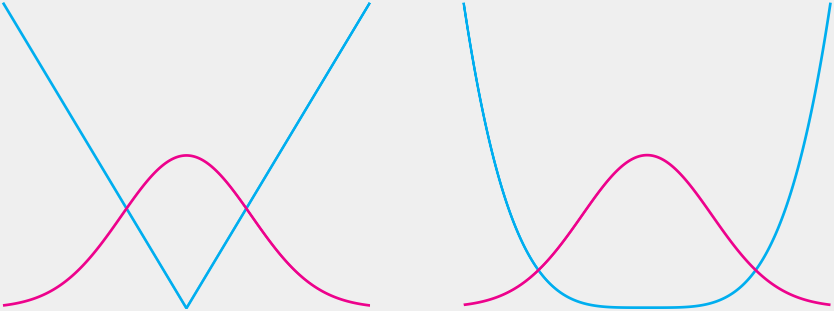
Example: Stochastic Sharpness-Aware Minimization:

- Flat regions of the empirical risk are thought to generalize well
- Flatness of a function $f : \mathbb{R}^d \to \mathbb{R}$ at a point $\mathbf{w}$ can be quantified via

$$\mathbf{w} \mapsto \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{N}(0, I_d)}\big[f(\mathbf{w} + \boldsymbol{\xi})\big] - f(\mathbf{w})$$



Noisy Algorithmic Regularization Methods



Example: Stochastic Sharpness-Aware Minimization:

- Flat regions of the empirical risk are thought to generalize well
- Flatness of a function $f : \mathbb{R}^d \to \mathbb{R}$ at a point $\mathbf{w}$ can be quantified via

$$\mathbf{w} \mapsto \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{N}(0, I_d)}\big[f(\mathbf{w} + \boldsymbol{\xi})\big] - f(\mathbf{w})$$

- To jointly optimize loss and flatness, must find

$$\mathbf{w} \in \operatorname*{arg\,min}_{\mathbf{w} \in \mathbb{R}^d} \left\{ \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{N}(0, \eta^2 \cdot I_d)}\big[f(\mathbf{w} + \boldsymbol{\xi})\big] \right\}$$

Example: Stochastic Sharpness-Aware Minimization:

- Flat regions of the empirical risk are thought to generalize well
- Flatness of a function $f : \mathbb{R}^d \to \mathbb{R}$ at a point $\mathbf{w}$ can be quantified via

$$\mathbf{w} \mapsto \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{N}(0, I_d)}\left[f(\mathbf{w} + \boldsymbol{\xi})\right] - f(\mathbf{w})$$

- To jointly optimize loss and flatness, must find

$$\mathbf{w} \in \arg\min_{\mathbf{w} \in \mathbb{R}^d} \left\{ f(\mathbf{w}) + \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{N}(0, \eta^2 \cdot I_d)}\left[f(\mathbf{w} + \boldsymbol{\xi})\right] - f(\mathbf{w}) \right\}$$

Example: Stochastic Sharpness-Aware Minimization:

- Flat regions of the empirical risk are thought to generalize well
- Flatness of a function $f : \mathbb{R}^d \to \mathbb{R}$ at a point $\mathbf{w}$ can be quantified via

$$\mathbf{w} \mapsto \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{N}(0, I_d)}\big[f(\mathbf{w} + \boldsymbol{\xi})\big] - f(\mathbf{w})$$

- To jointly optimize loss and flatness, must find

$$\mathbf{w} \in \arg\min_{\mathbf{w} \in \mathbb{R}^d} \Big\{ \mathbb{E}_{\boldsymbol{\xi} \sim \mathcal{N}(0, \eta^2 \cdot I_d)}\big[f(\mathbf{w} + \boldsymbol{\xi})\big] \Big\}$$

- Implies stochastic approximation algorithm

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \cdot \nabla f\big(\mathbf{w}_k + \boldsymbol{\xi}_k\big)$$

- Recall the general stochastic approximation algorithm

$$W_\ell(k+1) = W_\ell(k) - \alpha_k \cdot \nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f), \qquad \ell = 1, \dots, L$$

- Recall the general stochastic approximation algorithm, which can be rewritten into

$$
\begin{aligned}
&W_\ell(k+1) \\
&= W_\ell(k) - \alpha_k \cdot \mathbb{E}\big[\nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f) \mid \text{prev. iteration}\big] \\
&\quad + \alpha_k \cdot \Big(\mathbb{E}\big[\nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f) \mid \text{prev. iteration}\big] - \nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f)\Big)
\end{aligned}
$$

- Recall the general stochastic approximation algorithm, which can be rewritten into

$$
\begin{aligned}
&W_\ell(k+1) \\
&= W_\ell(k) - \alpha_k \cdot \mathbb{E}\big[\nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f) \mid \text{prev. iteration}\big] \\
&\quad + \alpha_k \cdot \Big(\mathbb{E}\big[\nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f) \mid \text{prev. iteration}\big] - \nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f)\Big)
\end{aligned}
$$

- Separates algorithm into deterministic part (expected gradient) and stochastic fluctuations around expectation

- Recall the general stochastic approximation algorithm, which can be rewritten into

$$\begin{aligned} &W_\ell(k+1) \\ &= W_\ell(k) - \alpha_k \cdot \mathbb{E}\big[\nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f) \mid \text{prev. iteration}\big] \\ &\quad + \alpha_k \cdot \Big(\mathbb{E}\big[\nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f) \mid \text{prev. iteration}\big] - \nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f)\Big) \end{aligned}$$

- Separates algorithm into deterministic part (expected gradient) and stochastic fluctuations around expectation
- Change in expected landscape may induce regularization

- Recall the general stochastic approximation algorithm, which can be rewritten into

$$
\begin{aligned}
&W_\ell(k+1) \\
&= W_\ell(k) - \alpha_k \cdot \mathbb{E}\big[\nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f) \mid \text{prev. iteration}\big] \\
&\quad + \alpha_k \cdot \Big(\mathbb{E}\big[\nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f) \mid \text{prev. iteration}\big] - \nabla_{W_\ell(k)} \widetilde{\mathcal{L}}_k(f)\Big)
\end{aligned}
$$

- Separates algorithm into deterministic part (expected gradient) and stochastic fluctuations around expectation
- Change in expected landscape may induce regularization
- Challenging analysis, due to many interlinked components

A Classical Example of Regularization

- Linear regression loss:

$$\boldsymbol{\beta} \mapsto \frac{1}{n} \cdot \sum_{i=1}^{n} \left(Y_i - \mathbf{X}_i^{\mathsf{t}} \boldsymbol{\beta}\right)^2$$

- Linear regression loss:

$$\boldsymbol{\beta} \mapsto \frac{1}{n} \cdot \sum_{i=1}^{n} \left(Y_i - \mathbf{X}_i^{\mathrm{t}} \boldsymbol{\beta}\right)^2 \propto \|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2$$

with

$$\mathbf{Y} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} \quad \text{and} \quad X = \begin{bmatrix} \mathbf{X}_1^{\mathrm{t}} \\ \vdots \\ \mathbf{X}_n^{\mathrm{t}} \end{bmatrix}$$

- Linear regression loss:

$$\boldsymbol{\beta} \mapsto \|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2$$

- Linear regression loss:

$$\boldsymbol{\beta} \mapsto \|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2$$

- If $X^{\mathrm{t}}X$ invertible, unique minimizer $\hat{\boldsymbol{\beta}} = (X^{\mathrm{t}}X)^{-1}X^{\mathrm{t}}\mathbf{Y}$

A Classical Example of Regularization

- Linear regression loss:

$$\boldsymbol{\beta} \mapsto \|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2$$

- If $X^{\mathsf{t}}X$ invertible, unique minimizer $\hat{\boldsymbol{\beta}} = (X^{\mathsf{t}}X)^{-1}X^{\mathsf{t}}\mathbf{Y}$
- What happens if $X^{\mathsf{t}}X$ is close to singular?

A Classical Example of Regularization

- Linear regression loss:

$$\boldsymbol{\beta} \mapsto \|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2$$

- If $X^{\mathsf{t}}X$ invertible, unique minimizer $\hat{\boldsymbol{\beta}} = (X^{\mathsf{t}}X)^{-1}X^{\mathsf{t}}\mathbf{Y}$
- What happens if $X^{\mathsf{t}}X$ is close to singular?
- Suppose $X = \sum_{j=1}^{d} \sigma_j \cdot \mathbf{u}_j\mathbf{v}_j^{\mathsf{t}}$ (SVD)

A Classical Example of Regularization

- Linear regression loss:

$$\boldsymbol{\beta} \mapsto \|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2$$

- If $X^{\mathsf{t}}X$ invertible, unique minimizer $\hat{\boldsymbol{\beta}} = (X^{\mathsf{t}}X)^{-1}X^{\mathsf{t}}\mathbf{Y}$
- What happens if $X^{\mathsf{t}}X$ is close to singular?
- Suppose $X = \sum_{j=1}^{d} \sigma_j \cdot \mathbf{u}_j\mathbf{v}_j^{\mathsf{t}}$ (SVD), then

$$\hat{\boldsymbol{\beta}} = \left(\left(\sum_{j=1}^{d} \sigma_j \cdot \mathbf{v}_j\mathbf{u}_j^{\mathsf{t}}\right)\left(\sum_{j=1}^{d} \sigma_j \cdot \mathbf{u}_j\mathbf{v}_j^{\mathsf{t}}\right)\right)^{-1} \left(\sum_{j=1}^{d} \sigma_j \cdot \mathbf{v}_j\mathbf{u}_j^{\mathsf{t}}\right)\mathbf{Y}$$

A Classical Example of Regularization

- Linear regression loss:

$$\boldsymbol{\beta} \mapsto \|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2$$

- If $X^{\mathrm{t}}X$ invertible, unique minimizer $\hat{\boldsymbol{\beta}} = (X^{\mathrm{t}}X)^{-1}X^{\mathrm{t}}\mathbf{Y}$
- What happens if $X^{\mathrm{t}}X$ is close to singular?
- Suppose $X = \sum_{j=1}^{d} \sigma_j \cdot \mathbf{u}_j\mathbf{v}_j^{\mathrm{t}}$ (SVD), then

$$\hat{\boldsymbol{\beta}} = \left(\sum_{j=1}^{d} \sigma_j^2 \cdot \mathbf{v}_j\mathbf{v}_j^{\mathrm{t}}\right)^{-1} \left(\sum_{j=1}^{d} \sigma_j \cdot \mathbf{v}_j\mathbf{u}_j^{\mathrm{t}}\right)\mathbf{Y}$$

A Classical Example of Regularization

- Linear regression loss:

$$\boldsymbol{\beta} \mapsto \|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2$$

- If $X^{\mathsf{t}}X$ invertible, unique minimizer $\hat{\boldsymbol{\beta}} = (X^{\mathsf{t}}X)^{-1}X^{\mathsf{t}}\mathbf{Y}$
- What happens if $X^{\mathsf{t}}X$ is close to singular?
- Suppose $X = \sum_{j=1}^{d} \sigma_j \cdot \mathbf{u}_j \mathbf{v}_j^{\mathsf{t}}$ (SVD), then

$$\hat{\boldsymbol{\beta}} = \left(\sum_{j=1}^{d} \frac{1}{\sigma_j^2} \cdot \mathbf{v}_j \mathbf{v}_j^{\mathsf{t}}\right)\left(\sum_{j=1}^{d} \sigma_j \cdot \mathbf{v}_j \mathbf{u}_j^{\mathsf{t}}\right)\mathbf{Y}$$

- Linear regression loss:

$$\boldsymbol{\beta} \mapsto \|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2$$

- If $X^{\mathrm{t}}X$ invertible, unique minimizer $\hat{\boldsymbol{\beta}} = (X^{\mathrm{t}}X)^{-1}X^{\mathrm{t}}\mathbf{Y}$
- What happens if $X^{\mathrm{t}}X$ is close to singular?
- Suppose $X = \sum_{j=1}^{d} \sigma_j \cdot \mathbf{u}_j\mathbf{v}_j^{\mathrm{t}}$ (SVD), then

$$\hat{\boldsymbol{\beta}} = \left(\sum_{j=1}^{d} \frac{1}{\sigma_j} \cdot \mathbf{v}_j\mathbf{u}_j^{\mathrm{t}}\right)\mathbf{Y}$$

A Classical Example of Regularization

- Linear regression loss:

$$\boldsymbol{\beta} \mapsto \|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2$$

- If $X^t X$ invertible, unique minimizer $\hat{\boldsymbol{\beta}} = (X^t X)^{-1} X^t \mathbf{Y}$
- What happens if $X^t X$ is close to singular?
- Suppose $X = \sum_{j=1}^{d} \sigma_j \cdot \mathbf{u}_j \mathbf{v}_j^t$ (SVD), then

$$\mathrm{Cov}(\hat{\boldsymbol{\beta}}) = \left(\sum_{j=1}^{d} \frac{1}{\sigma_j} \cdot \mathbf{v}_j \mathbf{u}_j^t \right) \mathrm{Cov}(\mathbf{Y}) \left(\sum_{j=1}^{d} \frac{1}{\sigma_j} \cdot \mathbf{u}_j \mathbf{v}_j^t \right)$$

- Variance diverges as $\sigma_j \to 0$

What can be done?

- Replace $X^{t}X$ with $X^{t}X + \lambda \cdot I_d$, λ to make it "less singular", so

$$\hat{\boldsymbol{\beta}}_{\lambda} = \left(X^{t}X + \lambda \cdot I_d\right)^{-1} X^{t}\mathbf{Y}$$

What can be done?

- Replace $X^{\mathrm{t}}X$ with $X^{\mathrm{t}}X + \lambda \cdot I_d$, λ to make it "less singular", so

$$\hat{\boldsymbol{\beta}}_\lambda = \left(X^{\mathrm{t}}X + \lambda \cdot I_d\right)^{-1} X^{\mathrm{t}}\mathbf{Y} = \left(\sum_{j=1}^{d} \frac{\sigma_j}{\sigma_j^2 + \lambda} \cdot \mathbf{v}_j \mathbf{u}_j^{\mathrm{t}}\right)\mathbf{Y}$$

What can be done?

- Replace $X^{\mathrm{t}}X$ with $X^{\mathrm{t}}X + \lambda \cdot I_d$, λ to make it "less singular", so

$$\hat{\boldsymbol{\beta}}_\lambda = \left(X^{\mathrm{t}}X + \lambda \cdot I_d\right)^{-1} X^{\mathrm{t}}\mathbf{Y} = \left(\sum_{j=1}^{d} \frac{\sigma_j}{\sigma_j^2 + \lambda} \cdot \mathbf{v}_j \mathbf{u}_j^{\mathrm{t}}\right)\mathbf{Y}$$

- Working backwards, we find that

$$\hat{\boldsymbol{\beta}}_\lambda = \arg\min_{\boldsymbol{\beta}} \left\{\|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2 + \lambda \cdot \|\boldsymbol{\beta}\|_2^2\right\}$$

A Simple Non-Convex Problem

- Consider the linear regression loss

$$\boldsymbol{\beta} \mapsto \frac{1}{2} \cdot \|\mathbf{Y} - X\boldsymbol{\beta}\|_2^2$$

- A deep version:

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \left\| \mathbf{Y} - X(\mathbf{w}_2 \odot \mathbf{w}_1) \right\|_2^2$$

($\mathbf{w}_2 \odot \mathbf{w}_1$ denotes the element-wise product)

A Simple Non-Convex Problem

- Diagonal linear network:

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \left\| \mathbf{Y} - X(\mathbf{w}_2 \odot \mathbf{w}_1) \right\|_2^2$$

- Diagonal linear network:

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \left\| \mathbf{Y} - X(\mathbf{w}_2 \odot \mathbf{w}_1) \right\|_2^2$$

- Suppose $\mathbf{Y} = X\mathbf{w}_*$ and X is an orthogonal matrix, then the loss turns into

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \|\mathbf{w}_* - \mathbf{w}_2 \odot \mathbf{w}_1\|_2^2$$

A Simple Non-Convex Problem

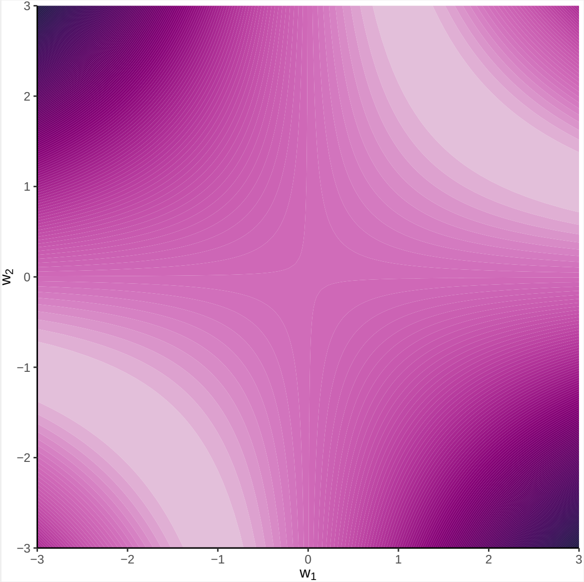
Exercise

How many critical points does the function

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \|\mathbf{w}_* - \mathbf{w}_2 \odot \mathbf{w}_1\|_2^2$$

have, and can you describe them?

A Simple Non-Convex Problem



Stochastic Sharpness-Aware Minimization

- Recall that flat regions are thought to generalize well, so want to minimize

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \mathbb{E}_{\xi_1, \xi_2 \sim \mathcal{N}(0, \eta^2 I_d)} \left[\left\| \mathbf{w}_* - (\mathbf{w}_2 + \boldsymbol{\xi}_2) \odot (\mathbf{w}_1 + \boldsymbol{\xi}_1) \right\|_2^2 \right]$$

- Recall that flat regions are thought to generalize well, so want to minimize

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \mathbb{E}_{\xi_1, \xi_2 \sim \mathcal{N}(0, \eta^2 I_d)} \left[\left\| \mathbf{w}_* - (\mathbf{w}_2 + \boldsymbol{\xi}_2) \odot (\mathbf{w}_1 + \boldsymbol{\xi}_1) \right\|_2^2 \right]$$

- Use stochastic approximation algorithm

$$\begin{bmatrix} \mathbf{w}_1(k+1) \\ \mathbf{w}_2(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1(k) \\ \mathbf{w}_2(k) \end{bmatrix} - \frac{\alpha_k}{2} \cdot \begin{bmatrix} \nabla_{\mathbf{w}_1(k)} \left\| \mathbf{w}_* - \big(\mathbf{w}_2(k) + \boldsymbol{\xi}_2(k)\big) \odot \big(\mathbf{w}_1(k) + \boldsymbol{\xi}_1(k)\big) \right\|_2^2 \\ \nabla_{\mathbf{w}_2(k)} \left\| \mathbf{w}_* - \big(\mathbf{w}_2(k) + \boldsymbol{\xi}_2(k)\big) \odot \big(\mathbf{w}_1(k) + \boldsymbol{\xi}_1(k)\big) \right\|_2^2 \end{bmatrix}$$

- Recall that flat regions are thought to generalize well, so want to minimize

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \mathbb{E}_{\xi_1, \xi_2 \sim \mathcal{N}(0, \eta^2 I_d)} \left[\left\| \mathbf{w}_* - (\mathbf{w}_2 + \boldsymbol{\xi}_2) \odot (\mathbf{w}_1 + \boldsymbol{\xi}_1) \right\|_2^2 \right]$$

- Use stochastic approximation algorithm

$$\begin{bmatrix} \mathbf{w}_1(k+1) \\ \mathbf{w}_2(k+1) \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1(k) \\ \mathbf{w}_2(k) \end{bmatrix} - \frac{\alpha_k}{2} \cdot \begin{bmatrix} \nabla_{\mathbf{w}_1(k)} \left\| \mathbf{w}_* - \left(\mathbf{w}_2(k) + \boldsymbol{\xi}_2(k)\right) \odot \left(\mathbf{w}_1(k) + \boldsymbol{\xi}_1(k)\right) \right\|_2^2 \\ \nabla_{\mathbf{w}_2(k)} \left\| \mathbf{w}_* - \left(\mathbf{w}_2(k) + \boldsymbol{\xi}_2(k)\right) \odot \left(\mathbf{w}_1(k) + \boldsymbol{\xi}_1(k)\right) \right\|_2^2 \end{bmatrix}$$

- Induced ℓ_2-regularizer

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \|\mathbf{w}_* - \mathbf{w}_2 \odot \mathbf{w}_1\|_2^2 + \frac{\eta^2}{2} \cdot \left(\|\mathbf{w}_1\|_2^2 + \|\mathbf{w}_2\|_2^2 \right)$$

Exercise

How many critical points does the function

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \|\mathbf{w}_* - \mathbf{w}_2 \odot \mathbf{w}_1\|_2^2 + \frac{\eta^2}{2} \cdot \left(\|\mathbf{w}_1\|_2^2 + \|\mathbf{w}_2\|_2^2\right)$$

have, and can you describe them?

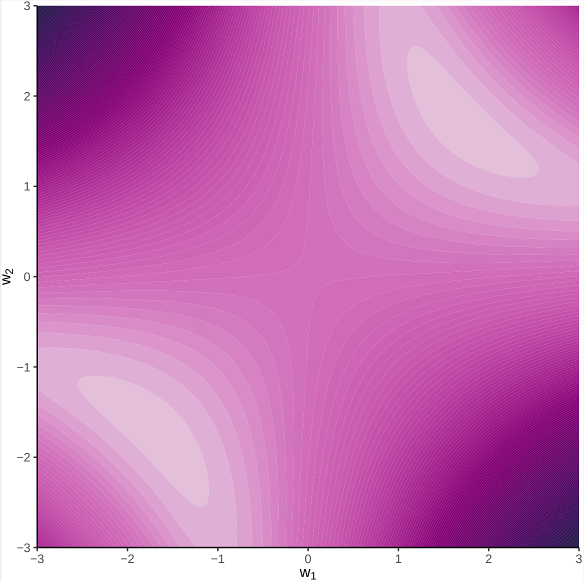
Theorem

Each critical point of the ℓ_2-penalized loss has the following form: pick $S \subset \{1, \dots, d\}$ and set

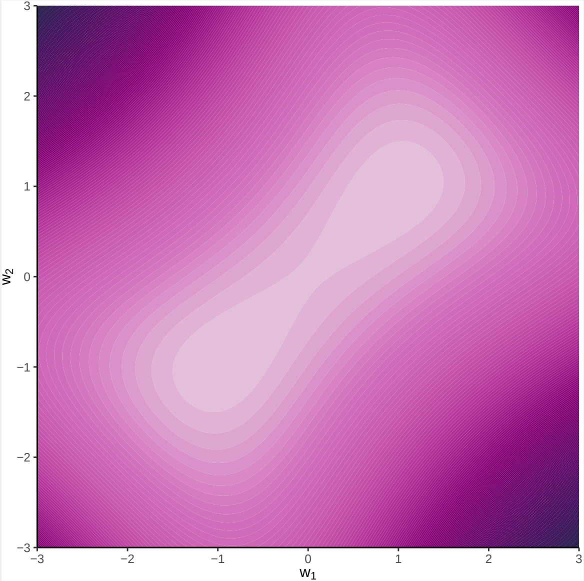
$$|\mathbf{w}_{1,i}| = |\mathbf{w}_{2,i}| = \begin{cases} \sqrt{|\mathbf{w}_{*,i}| - \eta^2}, & \textit{if } i \in S \textit{ and } |\mathbf{w}_{*,i}| \geq \eta^2 \\ 0, & \textit{otherwise} \end{cases}$$

with $\operatorname{sign}(\mathbf{w}_{1,i}) \cdot \operatorname{sign}(\mathbf{w}_{2,i}) = \operatorname{sign}(\mathbf{w}_{*,i})$.

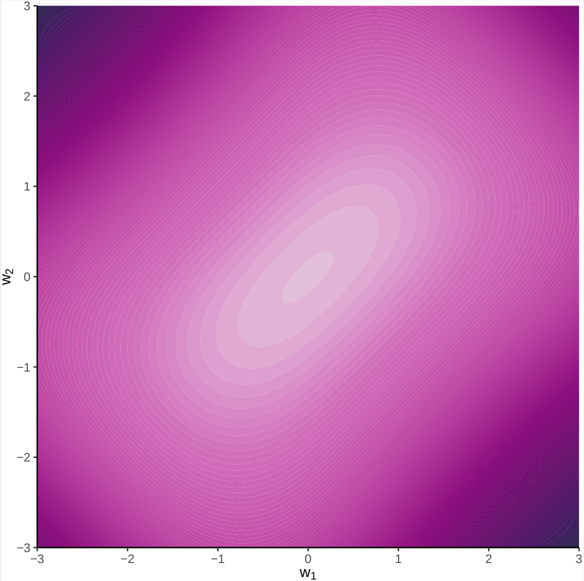
Diagonal Linear Networks with ℓ_2-Penalty



Diagonal Linear Networks with ℓ_2-Penalty



Diagonal Linear Networks with ℓ_2-Penalty



- Induced ℓ_2-regularizer

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \|\mathbf{w}_* - \mathbf{w}_2 \odot \mathbf{w}_1\|_2^2 + \frac{\eta^2}{2} \cdot \left(\|\mathbf{w}_1\|_2^2 + \|\mathbf{w}_2\|_2^2\right)$$

Diagonal Linear Networks with ℓ_2-Penalty

- Induced ℓ_2-regularizer

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \|\mathbf{w}_* - \mathbf{w}_2 \odot \mathbf{w}_1\|_2^2 + \frac{\eta^2}{2} \cdot \left(\|\mathbf{w}_1\|_2^2 + \|\mathbf{w}_2\|_2^2\right)$$

- Want to study the ℓ_2-penalized iterates

$$\begin{aligned}
&\begin{bmatrix} \mathbf{w}_1(k+1) \\ \mathbf{w}_2(k+1) \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{w}_1(k) \\ \mathbf{w}_2(k) \end{bmatrix} + \frac{\alpha_k}{2} \cdot \begin{bmatrix} \nabla_{\mathbf{w}_1(k)} \|\mathbf{w}_* - \mathbf{w}_2(k) \odot \mathbf{w}_1(k)\|_2^2 \\ \nabla_{\mathbf{w}_2(k)} \|\mathbf{w}_* - \mathbf{w}_2(k) \odot \mathbf{w}_1(k)\|_2^2 \end{bmatrix} \\
&\quad - \frac{\alpha_k \eta^2}{2} \cdot \begin{bmatrix} \nabla_{\mathbf{w}_1(k)} \|\mathbf{w}_1(k)\|_2^2 \\ \nabla_{\mathbf{w}_2(k)} \|\mathbf{w}_2(k)\|_2^2 \end{bmatrix}
\end{aligned}$$

Diagonal Linear Networks with ℓ_2-Penalty

- Induced ℓ_2-regularizer

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \|\mathbf{w}_* - \mathbf{w}_2 \odot \mathbf{w}_1\|_2^2 + \frac{\eta^2}{2} \cdot \left(\|\mathbf{w}_1\|_2^2 + \|\mathbf{w}_2\|_2^2\right)$$

- Want to study the ℓ_2-penalized iterates

$$\begin{aligned} &\begin{bmatrix} \mathbf{w}_1(k+1) \\ \mathbf{w}_2(k+1) \end{bmatrix} \\ = &\begin{bmatrix} \mathbf{w}_1(k) \\ \mathbf{w}_2(k) \end{bmatrix} - \alpha_k \cdot \left(\mathbf{w}_* - \mathbf{w}_2(k) \odot \mathbf{w}_1(k)\right) \cdot \begin{bmatrix} \mathbf{w}_2(k) \\ \mathbf{w}_1(k) \end{bmatrix} \\ &- \alpha_k \eta^2 \cdot \begin{bmatrix} \mathbf{w}_1(k) \\ \mathbf{w}_2(k) \end{bmatrix} \end{aligned}$$

- Induced ℓ_2-regularizer

$$(\mathbf{w}_1, \mathbf{w}_2) \mapsto \frac{1}{2} \cdot \|\mathbf{w}_* - \mathbf{w}_2 \odot \mathbf{w}_1\|_2^2 + \frac{\eta^2}{2} \cdot \left(\|\mathbf{w}_1\|_2^2 + \|\mathbf{w}_2\|_2^2\right)$$

- Want to study the ℓ_2-penalized iterates

$$\begin{bmatrix} \mathbf{w}_1(k+1) \\ \mathbf{w}_2(k+1) \end{bmatrix} = (1 - \alpha_k \eta^2) \cdot \begin{bmatrix} \mathbf{w}_1(k) \\ \mathbf{w}_2(k) \end{bmatrix} + \alpha_k \cdot \left(\mathbf{w}_* - \mathbf{w}_2(k) \odot \mathbf{w}_1(k)\right) \cdot \begin{bmatrix} \mathbf{w}_2(k) \\ \mathbf{w}_1(k) \end{bmatrix}$$

- Gradient descent can be hard to study, due to lack of analytical techniques, so let's simplify!

- Gradient descent can be hard to study, due to lack of analytical techniques, so let's simplify!
- Consider the recursion

$$\boldsymbol{\vartheta}_{k+1} = \boldsymbol{\vartheta}_k - \alpha_k \cdot \nabla_{\boldsymbol{\vartheta}_k} f(\boldsymbol{\vartheta}_k)$$

- Gradient descent can be hard to study, due to lack of analytical techniques, so let's simplify!
- Consider the recursion

$$\boldsymbol{\vartheta}_{k+1} - \boldsymbol{\vartheta}_k = -\alpha_k \cdot \nabla_{\boldsymbol{\vartheta}_k} f(\boldsymbol{\vartheta}_k)$$

- Gradient descent can be hard to study, due to lack of analytical techniques, so let's simplify!
- Consider the sum

$$\vartheta_{k+1} - \vartheta_0 = -\sum_{\ell=0}^{k} \alpha_\ell \cdot \nabla_{\vartheta_\ell} f(\vartheta_\ell)$$

- Gradient descent can be hard to study, due to lack of analytical techniques, so let's simplify!
- Consider the sum

$$\vartheta_{k+1} = \vartheta_0 - \sum_{\ell=0}^{k} \alpha_\ell \cdot \nabla_{\vartheta_\ell} f(\vartheta_\ell)$$

- Consider the sum

$$\boldsymbol{\vartheta}_{k+1} = \boldsymbol{\vartheta}_0 - \sum_{\ell=0}^{k} \alpha_\ell \cdot \nabla_{\boldsymbol{\vartheta}_\ell} f(\boldsymbol{\vartheta}_\ell)$$

- If $\sup_\ell \alpha_\ell \to 0$, converges to continuous-time function ($t \in \mathbb{R}_{\geq 0}$)

$$\boldsymbol{\vartheta}_t = \boldsymbol{\vartheta}_0 - \int_0^t \nabla_{\boldsymbol{\vartheta}_s} f(\boldsymbol{\vartheta}_s)\, \mathrm{d}s$$

- Consider the sum

$$\boldsymbol{\vartheta}_{k+1} = \boldsymbol{\vartheta}_0 - \sum_{\ell=0}^{k} \alpha_\ell \cdot \nabla_{\boldsymbol{\vartheta}_\ell} f(\boldsymbol{\vartheta}_\ell)$$

- If $\sup_\ell \alpha_\ell \to 0$, converges to continuous-time function ($t \in \mathbb{R}_{\geq 0}$)

$$\boldsymbol{\vartheta}_t = \boldsymbol{\vartheta}_0 - \int_0^t \nabla_{\boldsymbol{\vartheta}_s} f(\boldsymbol{\vartheta}_s)\, \mathrm{d}s$$

- Trajectory $t \mapsto \vartheta_t$ solves the system of ODEs

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\vartheta}_t = -\nabla_{\boldsymbol{\vartheta}_t} f(\boldsymbol{\vartheta}_t)$$

with boundary condition $\boldsymbol{\vartheta}_0$

- Consider the sum

$$\boldsymbol{\vartheta}_{k+1} = \boldsymbol{\vartheta}_0 - \sum_{\ell=0}^{k} \alpha_\ell \cdot \nabla_{\boldsymbol{\vartheta}_\ell} f(\boldsymbol{\vartheta}_\ell)$$

- If $\sup_\ell \alpha_\ell \to 0$, converges to continuous-time function ($t \in \mathbb{R}_{\geq 0}$)

$$\boldsymbol{\vartheta}_t = \boldsymbol{\vartheta}_0 - \int_0^t \nabla_{\boldsymbol{\vartheta}_s} f(\boldsymbol{\vartheta}_s)\, \mathrm{d}s$$

- Trajectory $t \mapsto \vartheta_t$ solves the system of ODEs

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\vartheta}_t = -\nabla_{\boldsymbol{\vartheta}_t} f(\boldsymbol{\vartheta}_t)$$

with boundary condition $\boldsymbol{\vartheta}_0$ (gradient flow of f)

- In our model, the gradient flow with ℓ_2-penalty takes form

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{w}_1(t)\\ \mathbf{w}_2(t)\end{bmatrix} = -\frac{1}{2}\cdot\begin{bmatrix}\nabla_{\mathbf{w}_1(t)}\|\mathbf{w}_* - \mathbf{w}_2(t)\odot\mathbf{w}_1(t)\|_2^2\\ \nabla_{\mathbf{w}_2(t)}\|\mathbf{w}_* - \mathbf{w}_2(t)\odot\mathbf{w}_1(t)\|_2^2\end{bmatrix} - \frac{\eta^2}{2}\cdot\begin{bmatrix}\nabla_{\mathbf{w}_1(t)}\|\mathbf{w}_1(t)\|_2^2\\ \nabla_{\mathbf{w}_1(t)}\|\mathbf{w}_2(t)\|_2^2\end{bmatrix}$$

The ℓ_2-Penalized Flow

- In our model, the gradient flow with ℓ_2-penalty takes form

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{w}_1(t)\\ \mathbf{w}_2(t)\end{bmatrix} = \left(\mathbf{w}_* - \mathbf{w}_2(t) \odot \mathbf{w}_1(t)\right) \cdot \begin{bmatrix}\mathbf{w}_2(t)\\ \mathbf{w}_1(t)\end{bmatrix} - \eta^2 \cdot \begin{bmatrix}\mathbf{w}_1(t)\\ \mathbf{w}_2(t)\end{bmatrix}$$

- In our model, the gradient flow with ℓ_2-penalty takes form

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{w}_1(t)\\ \mathbf{w}_2(t)\end{bmatrix} = \Big(\mathbf{w}_* - \mathbf{w}_2(t) \odot \mathbf{w}_1(t)\Big) \cdot \begin{bmatrix}\mathbf{w}_2(t)\\ \mathbf{w}_1(t)\end{bmatrix} - \eta^2 \cdot \begin{bmatrix}\mathbf{w}_1(t)\\ \mathbf{w}_2(t)\end{bmatrix}$$

Exercise

As $t \to \infty$, the gradient flow converges to a critical point of the ℓ_2-penalized loss. We know that all critical points satisfy $\mathbf{w}_1 \odot \mathbf{w}_1 = \mathbf{w}_2 \odot \mathbf{w}_2$, so

$$\lim_{t\to\infty}\Big(\mathbf{w}_1(t) \odot \mathbf{w}_1(t) - \mathbf{w}_2(t) \odot \mathbf{w}_2(t)\Big) = \mathbf{0},$$

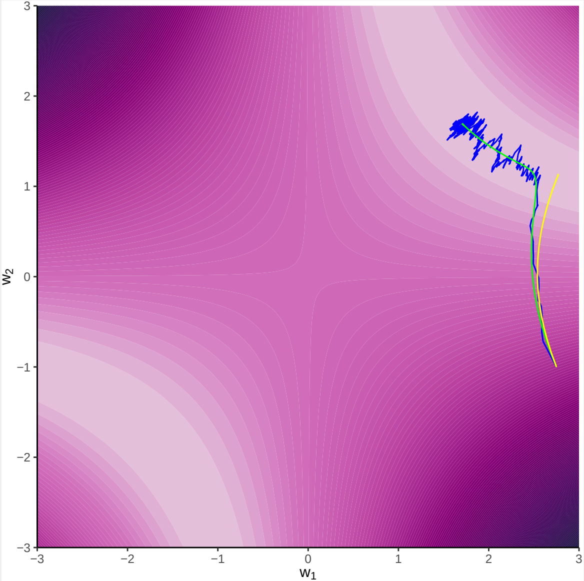
but how can you characterize this convergence?

Theorem

For every $t \geq 0$,

$$\begin{aligned} &\mathbf{w}_1(t) \odot \mathbf{w}_1(t) - \mathbf{w}_2(t) \odot \mathbf{w}_2(t) \\ &= e^{-2\eta^2 t} \cdot \Big(\mathbf{w}_1(0) \odot \mathbf{w}_1(0) - \mathbf{w}_2(0) \odot \mathbf{w}_2(0)\Big). \end{aligned}$$

The ℓ_2-Penalized Flow



The ℓ_2-Penalized Flow