S&DS 365 / 665

Intermediate Machine Learning

Random Features and Double Descent

(continued)

September 25

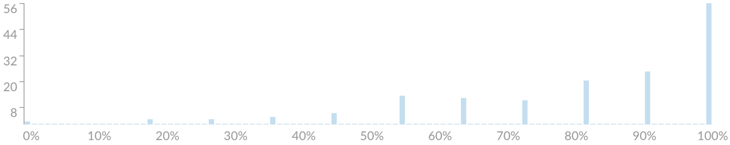
Reminders

- Assignment 1 is due Wednesday
- Assignment 2 posted Wednesday
- Quiz 2 grades posted

Quiz Summary

Section Filter ▾ | Student Analysis | Item Analysis

Average Score	High Score	Low Score	Standard Deviation	Average Time
81%	100%	0%	2.36	18:24



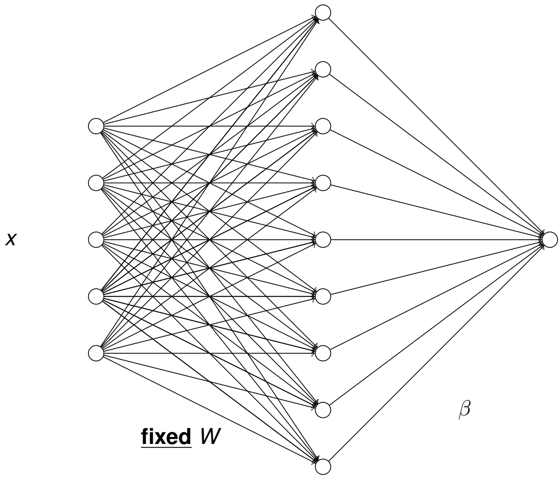
Random features

Fix the weights at their random initializations, for all but the last layer

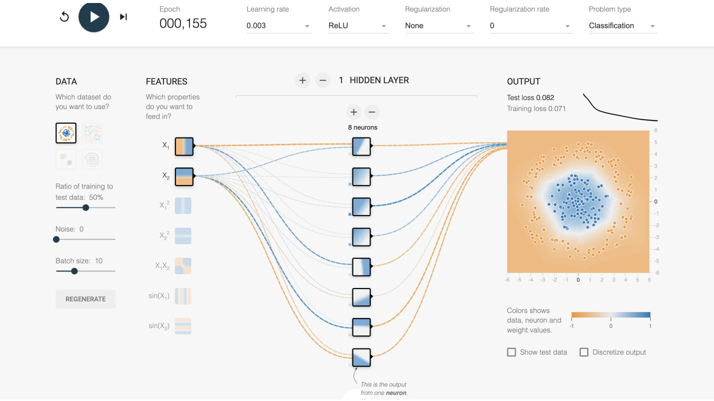
Just train the parameters β

This is called the *random features model*. It's a linear model with random covariates obtained from the hidden neurons.

Random features model



Demo



https://playground.tensorflow.org/

What's going on?

- These models are curiously robust to overfitting
- Why is this?
- Some insight: Kernels and double descent

Double descent

We went over notes on minimum norm and double descent the board.

A problem on Assignment 2 will help solidify your understanding of this.

OLS and minimal norm solution

OLS: $p < n$

$$\widehat{\beta} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T Y$$

Minimal norm solution: $p > n$:

$$\widehat{\beta}_{\text{mn}} = \mathbb{X}^T(\mathbb{X}\mathbb{X}^T)^{-1} Y$$

"Ridgeless regression"

As λ decreases to zero, the ridge regression estimate:

- Converges to OLS in the "classical regime" $\gamma < 1$
- Converges to $\widehat{\beta}_{\text{mn}}$ in "overparameterized regime" $\gamma > 1$

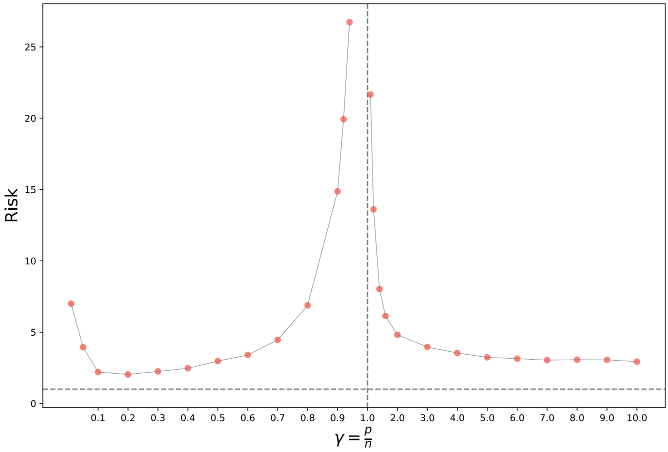
“Ridgeless regression”

This is a consequence of

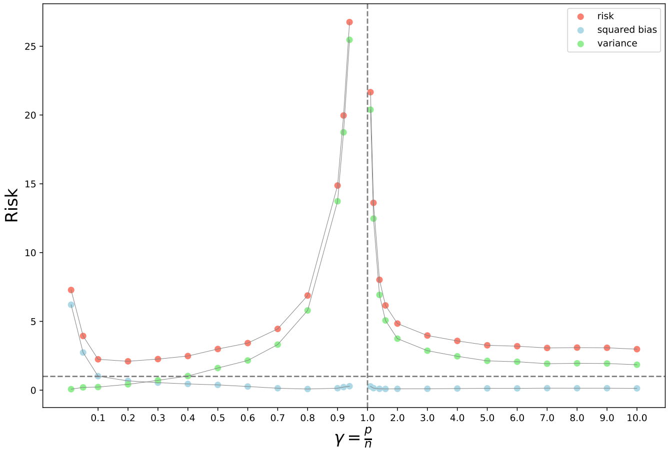
$$(X^T X + \lambda I_p)^{-1} X^T = X^T (X X^T + \lambda I_n)^{-1}$$

which follows from the Woodbury formula (Assn 2).

Double descent



Double descent



Double descent

- As $\gamma \to \infty$, the bias stays small
- Each entry of $\frac{1}{p}\mathbb{X}\mathbb{X}^T$ is the average over an increasing number of identically distributed random vectors
- As a result, the variance decreases

The theory underlying this is out of scope for our class; please see the references in the notes if interested.

Neural tangent kernel

The *neural tangent kernel (NTK)* has been useful in understanding the performance of large neural networks, and the dynamics of stochastic gradient descent training.

Parameterized functions

Suppose we have a parameterized function $f_\theta(x) \equiv f(x; \theta)$

Almost all machine learning takes this form — for classification and regression, these give us estimates of the regression function

For neural nets, the parameters θ are all of the weight matrices and bias (intercept) vectors across the layers.

Feature maps

Suppose we have a parameterized function $f_\theta(x) \equiv f(x;\theta)$

We then define a *feature map*

$$x \mapsto \varphi(x) = \nabla_\theta f(x;\theta_0) = \begin{pmatrix} \frac{\partial f(x;\theta_0)}{\partial \theta_1} \\ \frac{\partial f(x;\theta_0)}{\partial \theta_2} \\ \vdots \\ \frac{\partial f(x;\theta_0)}{\partial \theta_p} \end{pmatrix}$$

This defines a Mercer kernel

$$K(x,x') = \varphi(x)^T \varphi(x') = \nabla_\theta f(x;\theta_0)^T \nabla_\theta f(x';\theta_0)$$

Feature maps

This defines a Mercer kernel

$$K(x, x') = \varphi(x)^T \varphi(x') = \nabla_\theta f(x; \theta_0)^T \nabla_\theta f(x'; \theta_0)$$

What is the NTK for the random features model?

Feature maps

The NTK for the random features model is

$$K(x, x') = h(x)^T h(x')$$

Feature maps

Conversely, a deep neural network with a large number of neurons is approximately equivalent to a random features model!

Why?

NTK and SGD

- The dynamics of stochastic gradient descent for deep networks has been studied
- Upshot: As the number of neurons in the layers grows, the parameters in the network barely change during training, even though the training error quickly decreases to zero

NTK and random features

And, if the parameters only change by a small amount, a linear approximation can be used:

Let $\theta = \theta_0 + \beta$. Then

$$\begin{aligned} f(x, \theta) &\approx f(x, \theta_0) + \nabla_\theta f(x, \theta_0)^T \beta \\ &= \nabla_\theta f(x, \theta_0)^T \beta \end{aligned}$$

assuming that $f(x, \theta_0) = 0$ (not a problem to assume this)

NTK and random features

This tells us that the neural network is (approximately) equivalent to a random features model!

The random features are $h(x) \equiv \nabla_\theta f(x, \theta_0)$

Summary

- Neural nets are layered linear models with nonlinearities added
- Trained using stochastic gradient descent with backprop
- A surprise in the risk properties: Double descent
- Kernel connection: NTK