S&DS 365 / 665

Intermediate Machine Learning

Random Features and Double Descent

September 17

Yale

Version of 2025/09/17 at 12:45

Reminders

- Assignment 1 is out, due a week from tomorrow
- Quiz 2 available at 2:30 pm today on Canvas; 48 hours / 20 mins
- Any material covered in class (up to and including Monday)
- OH schedule posted to Canvas

Last time: Basics of neural nets

1. Basic architecture of feedforward neural nets
2. Backpropagation
3. Examples: TensorFlow

Today:

- NTK and double descent

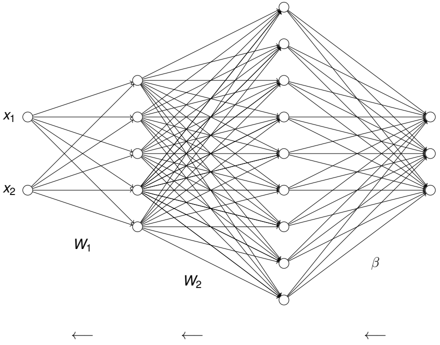
Next up: Convolutional neural nets

- Mechanics of convolutional networks
- Filters and pooling and flattening (oh my!)
- Example: Classifying Ca2+ brain scans
- Other examples

Training neural networks

- The parameters are trained by stochastic gradient descent.
- To calculate derivatives we just use the chain rule, working our way backwards from the last layer to the first.

High level idea



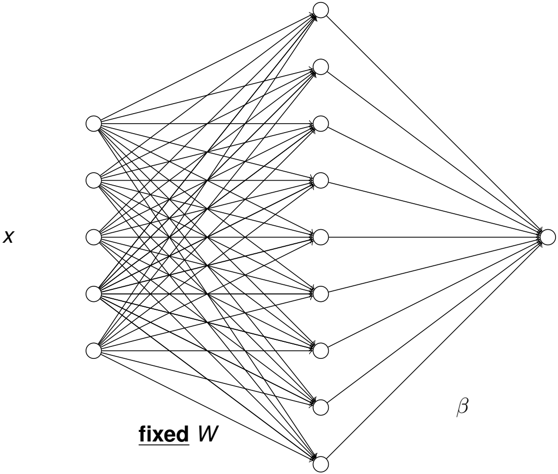
Start at last layer, send error information back to previous layers

Random features

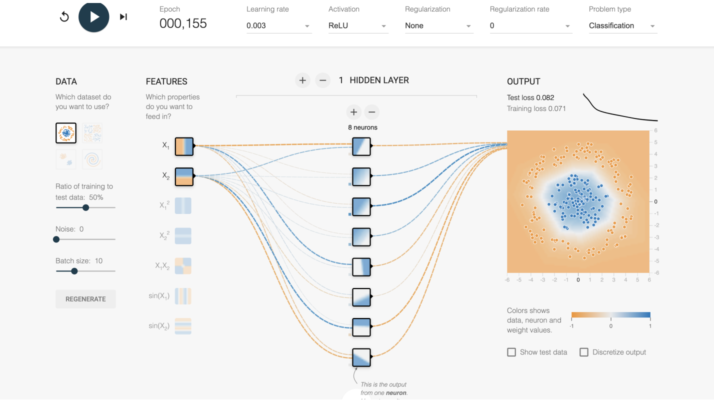
Today, we'll consider fixing the weights W at their random initializations, and just train the parameters β

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

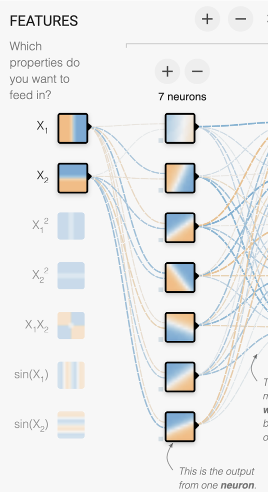
Fruit flies

Drosophila melanogaster

- Model scientific organism
- Eight Nobel prizes for research using *Drosophila*

The statistical fruit fly

A fruit fly for deep learning: Random features



http://www.argmin.net/2017/12/05/kitchen-sinks/

A fruit fly for deep learning: Random features

Random Features for Large-Scale Kernel Machines

Ali Rahimi
Intel Research Seattle
Seattle, WA 98105
ali.rahimi@intel.com

Benjamin Recht
Caltech IST
Pasadena, CA 91125
brecht@ist.caltech.edu

Abstract

To accelerate the training of kernel machines, we propose to map the input data to a randomized low-dimensional feature space and then apply existing fast linear methods. The features are designed so that the inner products of the transformed data are approximately equal to those in the feature space of a user specified shift-invariant kernel. We explore two sets of random features, provide convergence

http://www.argmin.net/2017/12/05/kitchen-sinks/

A fruit fly for deep learning: Random features

http://www.argmin.net/2017/12/05/kitchen-sinks/

A fruit fly for deep learning: Random features

arg min$_{\text{blog}}$ About

Reflections on Random Kitchen Sinks

Ali Rahimi and Ben Recht • Dec 5, 2017

Ed. Note: Ali Rahimi and I won the test of time award at NIPS 2017 for our paper "Random Features for Large-scale Kernel Machines". This post is the text of the acceptance speech we wrote. An addendum with some reflections on this talk appears in the following post.

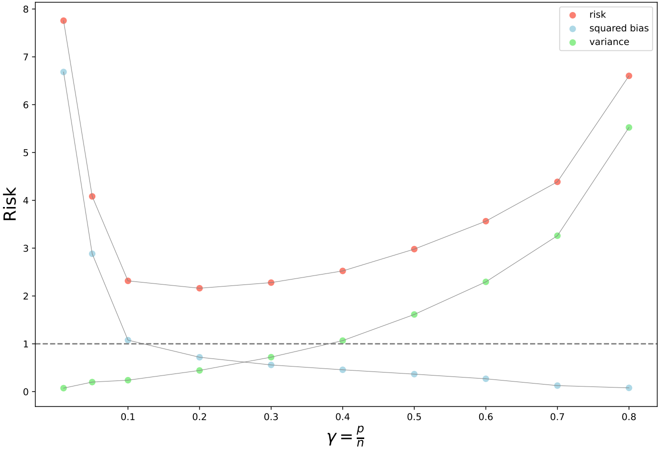
Video of the talk can be found here.

It feels great to get an award. Thank you. But I have to say, nothing makes you feel old like an award called a "test of time". It's forcing me to accept my age. Ben and I are both old now, and we've decided to name this talk accordingly.

Back When We Were Kids

We're getting this award for this paper. But this paper was the beginning of a trilogy of sorts. And like all stories worth telling, the good stuff happens in the middle, not at the beginning. If you'll put up with my old man ways, I'd like to tell you the story of these papers, and take you way back to NIPS 2006, when Ben and I were young spry men and dinosaurs roamed the earth.

Classical risk: Single descent for OLS



Double descent

We'll go over notes on the double descent phenomenon on the board, which will help you to complete a problem on the next assignment.

`https://github.com/YData123/sds365-fa25/raw/main/notes/double-descent.pdf`

OLS and minimal norm solution

OLS: $p < n$

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

Minimal norm solution: $p > n$:

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

"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" $\gamma > 1$

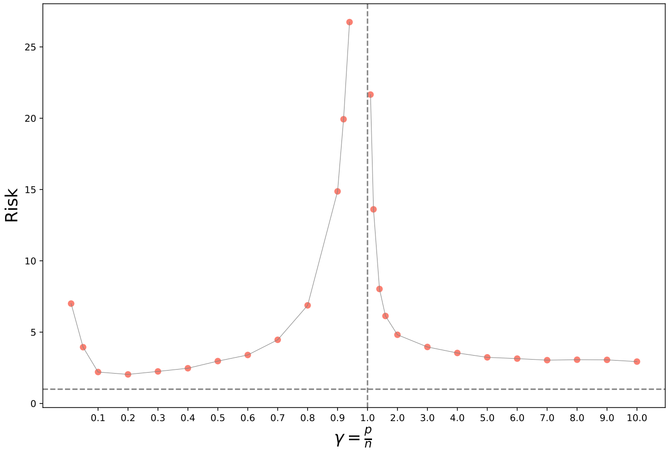
"Ridgeless regression"

This is a consequence of

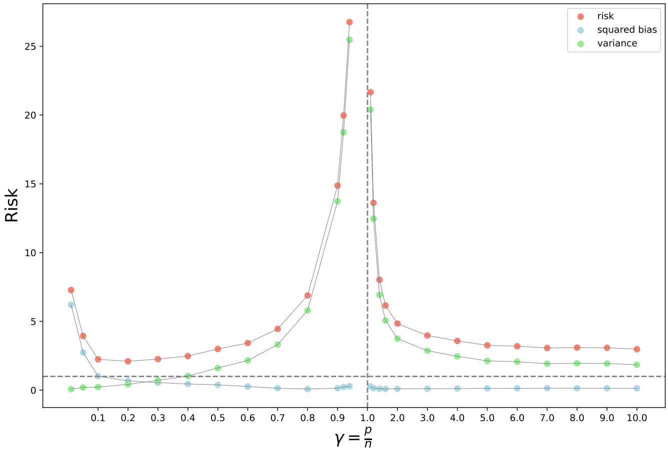
$$(X^T X + \lambda I_p)^{-1} X^T = X^T (X X^T + \lambda I_n)^{-1}$$

which follows from the Woodbury formula (Assn 2).

Double descent



Double descent



Double descent

- As $\gamma \to \infty$, the bias stays small – data are always interpolated
- 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*

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

This defines a Mercer kernel

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

This is sometimes called the Fisher Information Kernel.

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?

This is sometimes called the Fisher Information Kernel.

Feature maps

The NTK for the random features model is

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

This is sometimes called the Fisher Information Kernel.

Feature maps

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

Why? The next three slides sketch the argument.

This is sometimes called the Fisher Information Kernel.

NTK and SGD

- The dynamics of stochastic gradient descent for deep networks has been studied
- Mathematical result: 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

Consequence: 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)

Note: β here is not the vector of weights in the last layer. It's a vector of new parameters that combine the "features" that are the derivatives of the neural net output as a function, with respect to the weights, evaluated at a random initialization.

NTK and random features

Putting these two results together, tells us that a 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
- Insight into risk properties: Overparameterization and double descent
- Kernel connection: Neural Tangent Kernel (NTK)