S&DS 365 / 565

Intermediate Machine Learning

Kernels and Neural Networks

September 15

Yale

Reminders

- Assignment 1 out; due September 25 (week from this Thu)
- Quiz 2 posted Wednesday, material up to today
- Check Canvas/EdD for office hours—please join us!

Today: Kernels and Neural nets

1. Recap/discussion of RKHS concepts
2. Basic architecture of feedforward neural nets
3. Biological analogy and inspiration
4. Backpropagation
5. Examples: TensorFlow Playground
6. Next time: NTK and double descent

1: Mercer kernel recap

Summary from last time

- Smoothing methods compute local averages, weighting points by a kernel. The details of the kernel don't matter much
- Mercer kernels using penalization rather than smoothing
- Defining property: Matrix $\mathbb{K}$ is always positive semidefinite
- Equivalent to a type of ridge regression in function space
- The curse of dimensionality limits use of both approaches

Mercer Kernels: The big picture

Instead of using local smoothing, we can optimize the fit to the data subject to regularization (penalization). Choose $\widehat{m}$ to minimize

$$\sum_i (Y_i - \widehat{m}(X_i))^2 + \lambda \, \text{penalty}(\widehat{\mathrm{m}})$$

where penalty($\widehat{\mathrm{m}}$) is a *roughness penalty*.

λ is a parameter that controls the amount of smoothing.

How do we construct a penalty that measures roughness? One approach is: *Mercer Kernels* and *RKHS = Reproducing Kernel Hilbert Spaces.*

What is a Mercer Kernel?

A kernel is a bivariate function $K(x, x')$. We think of this as a measure of “similarity” between points x and x'.

What is a Mercer Kernel?

A kernel is a bivariate function $K(x, x')$. We think of this as a measure of "similarity" between points x and x'.

A Mercer kernel has a special property: For any set of points $x_1, \ldots, x_n$ the $n \times n$ matrix

$$\mathbb{K} = \left[K(x_i, x_j)\right]$$

is positive semidefinite (no negative eigenvalues)

What is a Mercer Kernel?

A kernel is a bivariate function $K(x, x')$. We think of this as a measure of "similarity" between points x and x'.

A Mercer kernel has a special property: For any set of points $x_1, \ldots, x_n$ the $n \times n$ matrix

$$\mathbb{K} = \left[K(x_i, x_j)\right]$$

is positive semidefinite (no negative eigenvalues)

This property has many important (and beautiful!) mathematical consequences. It is a characterization of Mercer kernels.

Mercer Kernels: Key example

A Gaussian gives us a Mercer kernel:

$$K(x, x') = e^{-\frac{\|x - x'\|^2}{2h^2}}$$

Note: Here we fix the bandwidth h.

Basis functions

We can create a set of *basis functions* based on K.

Fix z and think of $K(z, x)$ as a function of x. That is,

$$K(z, x) = K_z(x)$$

is a function of the second argument, with the first argument fixed.

Defining an inner product (geometry)

Because of the positive semidefinite property, we can create an *inner product* and *norm* over the span of these functions

If $f(x) = \sum_i \alpha_i K_{x_i}(x)$, $g(x) = \sum_i \beta_i K_{x_i}(x)$, the inner product is

$$\begin{aligned}\langle f, g\rangle_K &= \sum_i \sum_j \alpha_i \beta_j K(x_i, x_j) \\ &= \alpha^T \mathbb{K} \beta\end{aligned}$$

where $\mathbb{K} = [K(x_i, x_j)]$

Defining an inner product (geometry)

Because of the positive semidefinite property, we can create an *inner product* and *norm* over the span of these functions

The norm is

$$\begin{aligned} \|f\|_K^2 = \langle f, f \rangle_K &= \sum_i \sum_j \alpha_i \alpha_j K(x_i, x_j) \\ &= \alpha^T \mathbb{K} \alpha \geq 0 \end{aligned}$$

In fact $\|f\|_K = 0$ *if and only if* $f = 0$ (see notes)

Assignment 1 will solidify your understanding of Mercer kernels and kernel ridge regression!

Reducing to finite dimensions

Representer Theorem

Let $\widehat{m}$ minimize

$$J(m) = \sum_{i=1}^{n} (Y_i - m(X_i))^2 + \lambda \|m\|_K^2.$$

Then

$$\widehat{m}(x) = \sum_{i=1}^{n} \alpha_i K(X_i, x)$$

for some $\alpha_1, \ldots, \alpha_n$.

So, we only need to find the coefficients

$$\alpha = (\alpha_1, \ldots, \alpha_n).$$

Gradient descent

The gradient descent updates to α are

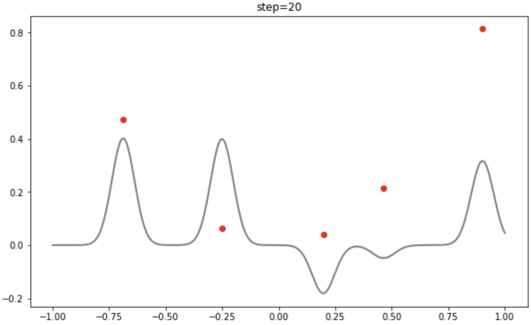
$$\alpha \longleftarrow \alpha + \eta\,(\mathbb{K}(y - \mathbb{K}\alpha) - \lambda\mathbb{K}\alpha)$$

where $\mathbb{K}$ is the $n \times n$ Gram matrix and $\eta > 0$ is a step size.

Demo

```
    if step % 10 == 0:
        plot_function_and_data(x, f, X, y, t=step, sleeptime=.5)
    alpha = alpha + stepsize * (K.T @ (y - K @ alpha) - lam * K
```

[6] 6.4s



Kernels from features—and vice-versa

If $x \to \varphi(x) \in \mathbb{R}^d$ is a feature mapping, we can define a Mercer kernel by

$$K(x, x') = \varphi(x)^T \varphi(x')$$

Conversely, for any Mercer kernel we can derive the corresponding feature map (from the spectral theorem)

See the slides at the end

The importance of being Kernelist

- Mercer kernels play a central role in machine learning
  - Can define similarity functions that are kernels for all kinds of data — graphs, molecules, text documents
  - Gaussian processes
  - Modern understanding of deep neural networks

Summary for kernels

- Smoothing methods compute local averages, weighting points by a kernel. The details of the kernel don't matter much
- Mercer kernels using penalization rather than smoothing
- Defining property: Matrix $\mathbb{K}$ is always positive semidefinite
- Equivalent to a type of ridge regression in function space
- The curse of dimensionality limits use of both approaches

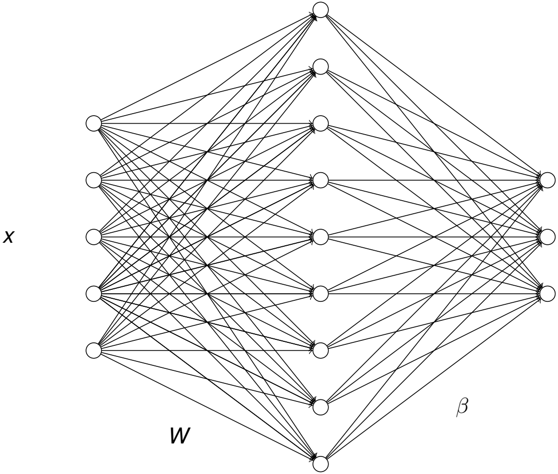
2: Neural net basics

Recall :-)

What does "Intermediate" imply?

- A second course in machine learning
- Assume familiar with things like PCA, bias/variance, maximum likelihood, basics of neural nets
- Have experimented with basic ML methods on some data sets
- Previous exposure to Python
- More on this later...

4


x
W
β

Interpretation

The parameter matrix W and vector β defines a (parametric) classification model, equivalent to a logistic regression:

$$\mathbb{P}(y \mid x) = \mathrm{Softmax}(\beta^T \varphi(Wx))$$

Interpretation

The parameter matrix W and vector β defines a (parametric) classification model, equivalent to a logistic regression:

$$\mathbb{P}(y \mid x) = \text{Softmax}(\beta^T \varphi(Wx))$$

More generally, with intercept parameters:

$$\mathbb{P}(y \mid x) = \text{Softmax}(\beta^T \varphi(Wx + b) + \beta_0)$$

Nonlinearities

Add nonlinearity

$$h(x) = \varphi(Wx + b)$$

applied component-wise.

For regression, the last layer is just linear:

$$f(x) = \beta^T h(x) + \beta_0$$

For classification, we use Softmax.

Nonlinearities

Commonly used nonlinearities:

$$\varphi(u) = \tanh(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$$

$$\varphi(u) = \text{sigmoid}(u) = \frac{e^u}{1 + e^u}$$

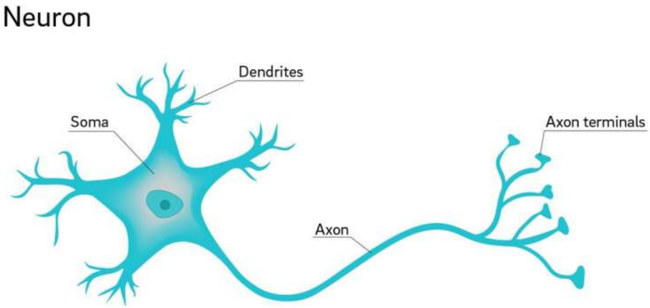
$$\varphi(u) = \text{relu}(u) = \max(u, 0)$$

Without the activation function φ, we just have a regular linear model.

Nonlinearities

So, a neural network is nothing more than a parametric regression model with a restricted type of nonlinearity

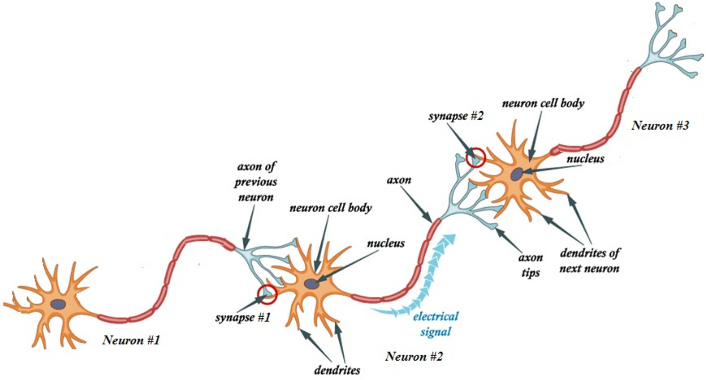
Biological Analogy



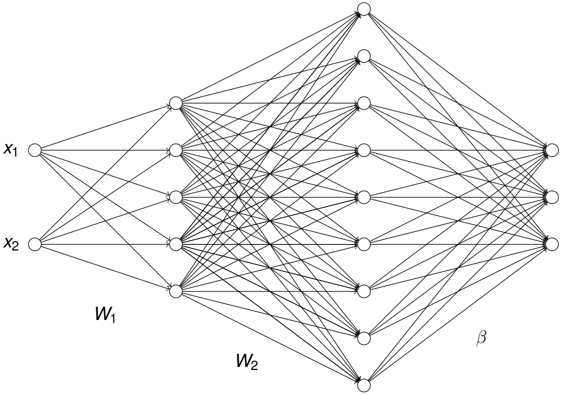
Biological Analogy

- The dendrites play the role of inputs, collecting signals from other neurons and transmitting them to the soma, which is the “central processing unit.”
- If the total input arriving at the soma reaches a threshold, an output is generated.
- The axon is the output device, which transmits the output signal to the dendrites of other neurons.

Biological Analogy



Two-layer perceptron



Corresponding model

This represents the family of models

$$\mathbb{P}(y \mid x) = \text{Softmax}(\beta^T \varphi(W_2\, \varphi(W_1 x)))$$

Corresponding model

This represents the family of models

$$\mathbb{P}(y \mid x) = \text{Softmax}(\beta^T \varphi(W_2\, \varphi(W_1 x)))$$

More generally, with intercept parameters:

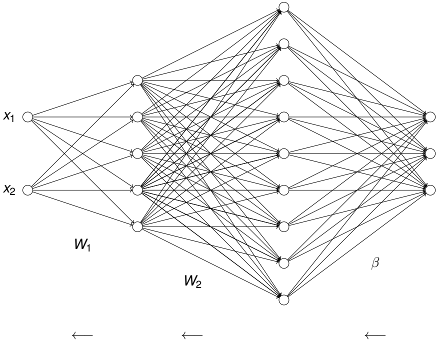
$$\mathbb{P}(y \mid x) = \text{Softmax}(\beta^T \varphi(W_2\, \varphi(W_1 x + b_1) + b_2) + \beta_0)$$

3: Backprop

Training

- 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

4: Demos

Interactive examples

https://playground.tensorflow.org/

What's going on?

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

Next time!

Summary

- Neural nets are layered linear models with nonlinearities added
- Trained using stochastic gradient descent with backprop
- Can be automated to train complex networks (with no math!)