Neural Networks Tutorial

Stefan G. Stark

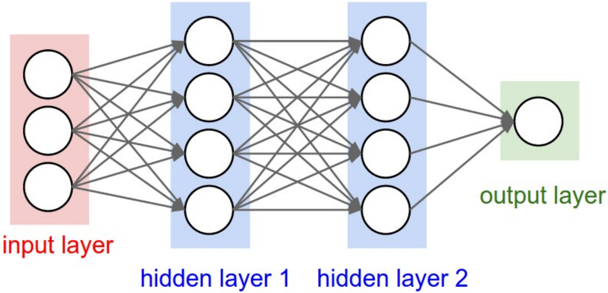
ETH Intro to Machine Learning Spring 2020

1 April 2020

Outline

- Neural Network Recap
  - Forward Pass
  - Backward Pass
  - Exam question
- Building large networks
  - Vanishing Gradients
  - Residual Neural Networks (ResNets)
  - Demo: Loading ResNet50 on your laptop
- HW2: Review Selected Problems
  - Problem 1
  - Problem 3
  - Problem 10

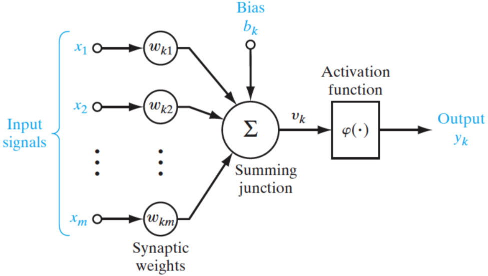
Neural Network recap



- Composed of modules called hidden layers
- Able to approximate non-linear functions

A single Hidden Layer

Linear transformation followed by a non-linear "activation"



Matrix Form

$\mathbf{y} = \phi(\mathbf{W}\mathbf{x} + \mathbf{b})$

Scalar form

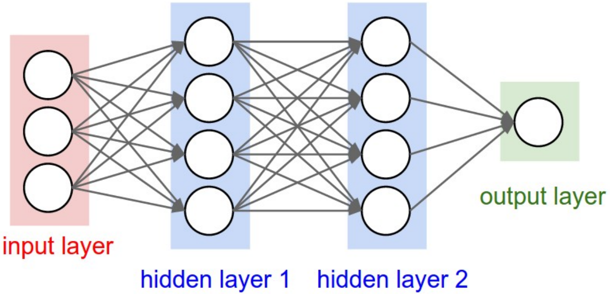
$y_k = \phi(\sum_i x_i w_{ki} + b_k)$

Haykin, Simon S., et al. Neural networks and learning machines. Vol. 3. Upper Saddle River: Pearson, 2009.

Forward Pass

Consider a deep neural net with L layers

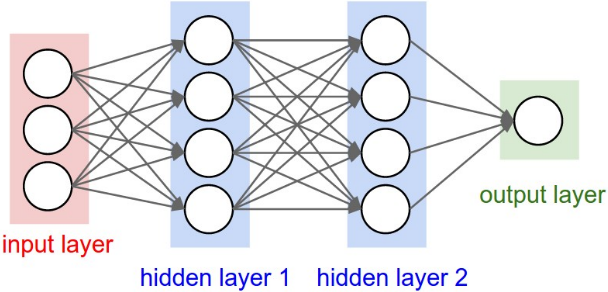
$$f(\mathbf{x}, \mathbf{W}) = \phi^{(L)}(W^{(L)}\phi^{(L-1)}(W^{(L-1)} \dots \phi^{(1)}(W^{(1)}\mathbf{x}) \dots)$$



Forward Pass

Consider a deep neural net with L layers

$$f(\mathbf{x}, \mathbf{W}) = \phi^{(L)}(W^{(L)}\phi^{(L-1)}(W^{(L-1)} \dots \phi^{(1)}(W^{(1)}\mathbf{x}) \dots)$$

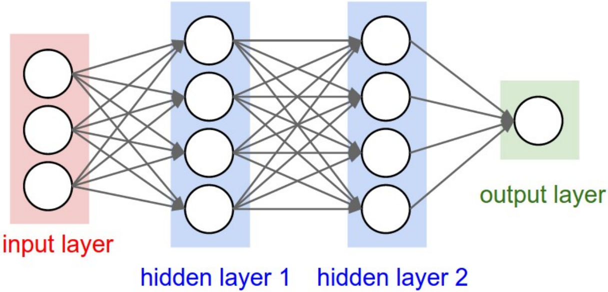


Why do we need non-linearities ϕ?

Forward Pass

Consider a deep neural net with L layers

$$f(\mathbf{x}, \mathbf{W}) = \phi^{(L)}(W^{(L)}\phi^{(L-1)}(W^{(L-1)} \ldots \phi^{(1)}(W^{(1)}\mathbf{x}) \ldots)$$



Why do we need non-linearities ϕ?

$$f_{linear}(\mathbf{x}, \mathbf{W}) = W^{(L)} W^{(L-1)} \ldots W^{(1)}\mathbf{x} = W^{*}\mathbf{x}$$

Training Neural nets

Given

- labels y^*, outputs $y = f(x)$
- loss function $\ell(y^*, y)$ on a single datapoint

Goal Minimize

$$L(\mathbf{W}) = \frac{1}{N} \sum_{i=1}^{N} \ell(y^*, y; \mathbf{W})$$

- Approximate $L(\mathbf{W})$ by subsampling dataset (batches)
- Use gradient based optimization methods, e.g. SGD, ADAM
- $W_{new} = W_{old} - \eta_t \frac{\partial}{\partial W} L(W_{old})$

Loss Functions: Regression

Labels: $y^* \in \mathbb{R}$ or $\mathbf{y}^* \in \mathbb{R}^d$
Output: Real-valued output (no activation)
Loss: e.g. L_2 loss

$$\ell(\mathbf{y}^*, \mathbf{y}) = ||\mathbf{y}^* - \mathbf{y}||_2^2$$

Loss Functions: Binary Classification

Labels: $y^* \in \{0, 1\}$
Output: single output neuron $y \in \mathbb{R}$. Probability of class 1:

$$\sigma = \frac{1}{1 + e^{-y}} \in (0, 1)$$

Loss: Binary Cross Entropy Loss

$$\ell(y^*, y) = -y^* \log(\sigma) - (1 - y^*) log(1 - \sigma)$$

Loss Functions: Multi-class Classification

Labels: $\mathbf{y}^*$ "one-hot" in $\mathbb{R}^C$
Output: $y \in \mathbb{R}^C$. Softmax: probability of class i:

$$\sigma_i = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

Loss: Cross Entropy Loss

$$\ell(\mathbf{y}^*, \mathbf{y}) = -\sum_i y_i^* \log(\sigma_i)$$

See MNIST, CIFAR, ImageNet

Training Neural Nets: Backpropagation

Use chain rule to compute gradients of $L(\mathbf{W})$

Training Neural Nets: Backpropagation

Use chain rule to compute gradients of $L(\mathbf{W})$
Define network recursively:

$$v^{(\ell)} = \phi(z^{(\ell)})$$
$$z^{(\ell)} = W^{(\ell)} v^{(\ell-1)}$$

Where $v^{(0)} = x$ and $v^{(L)} = f(x)$

Training Neural Nets: Backpropagation

Use chain rule to compute gradients of $L(\mathbf{W})$
Define network recursively:

$$v^{(\ell)} = \phi(z^{(\ell)})$$
$$z^{(\ell)} = W^{(\ell)} v^{(\ell-1)}$$

Where $v^{(0)} = x$ and $v^{(L)} = f(x)$

The gradient of the loss wrt an element of the k^{th} hidden layer is

$$\frac{\partial L(\mathbf{W})}{\partial w_{ij}^{(k)}} = \frac{\partial L}{\partial v^{(L)}} \frac{\partial v^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial v^{(L-1)}} \cdots \frac{\partial v^{(k)}}{\partial z^{(k)}} \frac{\partial z^{(k)}}{\partial w_{ij}^{(k)}}$$

Training Neural Nets: Backpropagation

Use chain rule to compute gradients of $L(\mathbf{W})$
Define network recursively:

$$v^{(\ell)} = \phi(z^{(\ell)})$$
$$z^{(\ell)} = W^{(\ell)} v^{(\ell-1)}$$

Where $v^{(0)} = x$ and $v^{(L)} = f(x)$

The gradient of the loss wrt an element of the k^{th} hidden layer is

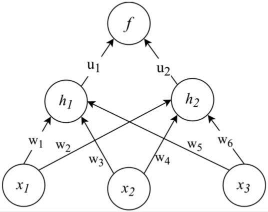
$$\frac{\partial L(\mathbf{W})}{\partial w_{ij}^{(k)}} = \frac{\partial L}{\partial v^{(L)}} \frac{\partial v^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial v^{(L-1)}} \cdots \frac{\partial v^{(k)}}{\partial z^{(k)}} \frac{\partial z^{(k)}}{\partial w_{ij}^{(k)}}$$

$$w_{ij}^{(k)} \leftarrow w_{ij}^{(k)} - \eta_t \frac{\partial L(\mathbf{W})}{\partial w_{ij}^{(k)}}$$

Exam question

Exam 2016 Question 5

Consider the following neural network with two logistic hidden units h_1, h_2, and three inputs x_1, x_2, x_3. The output neuron f is a linear unit, and we are using the squared error cost function $E = (y - f)^2$. The logistic function is defined as $\rho(x) = 1/\left(1 + e^{-x}\right)$.



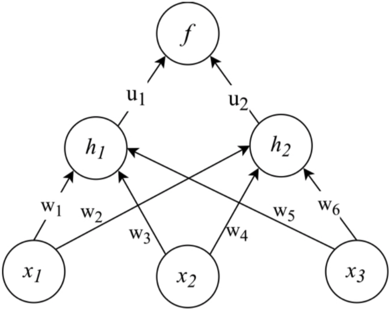
(i) Consider a single training example $\boldsymbol{x} = [x_1, x_2, x_3]$ with target output (label) y. Write down the sequence of calculations required to compute the squared error cost (called forward propagation).

(ii) A way to reduce the number of parameters to avoid overfitting is to tie certain weights together, so that they share a parameter. Suppose we decide to tie the weights w_1 and w_4, so that $w_1 = w_4 = w_{\text{tied}}$. What is the derivative of the error E with respect to w_{tied}, i.e. $\nabla_{w_{\text{tied}}} E$?

Exam question I: Forward Pass

Write down the sequence of calculations required to compute the squared error cost (called forward propagation).

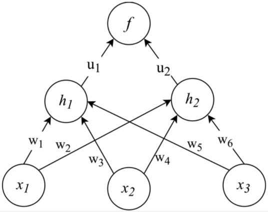
$$E = (y - f)^2$$
$$f = u_1 h_1 + u_2 h_2$$
$$h_1 = \rho(w_1 x_1 + w_3 x_2 + w_5 x_3)$$
$$h_2 = \rho(w_2 x_1 + w_4 x_2 + w_6 x_3)$$



Exam question

Exam 2016 Question 5

Consider the following neural network with two logistic hidden units h_1, h_2, and three inputs x_1, x_2, x_3. The output neuron f is a linear unit, and we are using the squared error cost function $E = (y - f)^2$. The logistic function is defined as $\rho(x) = 1/\left(1 + e^{-x}\right)$.



(i) Consider a single training example $\boldsymbol{x} = [x_1, x_2, x_3]$ with target output (label) y. Write down the sequence of calculations required to compute the squared error cost (called forward propagation).

(ii) A way to reduce the number of parameters to avoid overfitting is to tie certain weights together, so that they share a parameter. Suppose we decide to tie the weights w_1 and w_4, so that $w_1 = w_4 = w_{\text{tied}}$. What is the derivative of the error E with respect to w_{tied}, i.e. $\nabla_{w_{\text{tied}}} E$?

Exam question II: Backward Pass

First, let's define the linear part of the first hidden layer:

$$v_1 = w_{tied} x_1 + w_3 x_2 + w_5 x_3$$
$$v_2 = w_2 x_1 + w_{tied} x_2 + w_6 x_3$$

From I: $E = (y - f)^2$, $f = u_1 h_1 + u_2 h_2$ and $h_1 = \rho(v_1)$, $h_2 = \rho(v_2)$

Exam question II: Backward Pass

First, let's define the linear part of the first hidden layer:

$$v_1 = w_{tied}x_1 + w_3x_2 + w_5x_3$$
$$v_2 = w_2x_1 + w_{tied}x_2 + w_6x_3$$

From I: $E = (y - f)^2$, $f = u_1h_1 + u_2h_2$ and $h_1 = \rho(v_1)$, $h_2 = \rho(v_2)$

$$\frac{\partial E}{\partial w_{tied}} = \frac{\partial E}{\partial f}\left(\frac{\partial f}{\partial h_1}\frac{\partial h_1}{\partial v_1}\frac{\partial v_1}{\partial w_{tied}} + \frac{\partial f}{\partial h_2}\frac{\partial h_2}{\partial v_2}\frac{\partial v_2}{\partial w_{tied}}\right)$$

Exam question II: Backward Pass

First, let's define the linear part of the first hidden layer:

$$v_1 = w_{tied}x_1 + w_3x_2 + w_5x_3$$
$$v_2 = w_2x_1 + w_{tied}x_2 + w_6x_3$$

From I: $E = (y - f)^2$, $f = u_1h_1 + u_2h_2$ and $h_1 = \rho(v_1)$, $h_2 = \rho(v_2)$

$$\frac{\partial E}{\partial w_{tied}} = \frac{\partial E}{\partial f}\left(\frac{\partial f}{\partial h_1}\frac{\partial h_1}{\partial v_1}\frac{\partial v_1}{\partial w_{tied}} + \frac{\partial f}{\partial h_2}\frac{\partial h_2}{\partial v_2}\frac{\partial v_2}{\partial w_{tied}}\right)$$

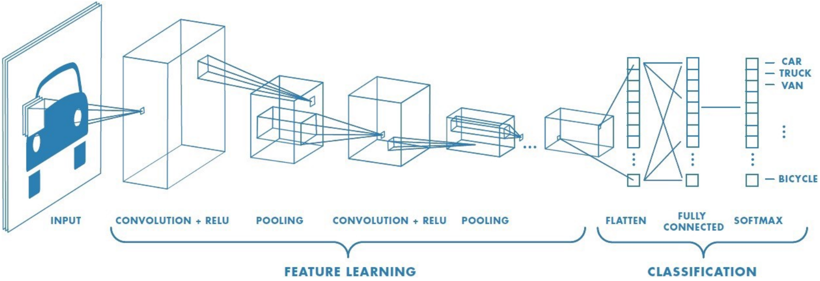
$\frac{\partial E}{\partial f} = 2(f - y)$, $\frac{\partial f}{\partial h_1} = u_1$ and $\frac{\partial v_1}{\partial w_{tied}} = x_1$. $\frac{\partial h_1}{\partial v_1}$ is harder ..

Exam question II: Backward Pass cont.

$$
\begin{aligned}
\frac{\partial h}{\partial v} = \frac{\partial \rho(v)}{\partial v} &= \frac{\partial}{\partial v}\frac{1}{1+e^{-v}} \\
&= -(1+e^{-v})^{-2}\frac{\partial}{\partial v}(1+e^{-v}) \\
&= -(1+e^{-v})^{-2}(-e^{-v}) \\
&= \frac{1}{(1+e^{-v})}\frac{e^{-v}}{1+e^{-v}} \\
&= \rho(v)(1-\rho(v)) \\
&= h(1-h)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial E}{\partial w_{tied}} &= \frac{\partial E}{\partial f}\left(\frac{\partial f}{\partial h_1}\frac{\partial h_1}{\partial v_1}\frac{\partial v_1}{\partial w_{tied}} + \frac{\partial f}{\partial h_2}\frac{\partial h_2}{\partial v_2}\frac{\partial v_2}{\partial w_{tied}}\right) \\
&= 2(f-y)\left(u_1 h_1(1-h_1)x_1 + u_2 h_2(1-h_2)x_2\right)
\end{aligned}
$$

CNNs & Representation Learning



- More layers $\rightarrow$ better representation
- Better representation $\rightarrow$ better accuracy

(Assuming you can optimize)

ResNets: Problem setting

Is learning better networks as easy as stacking more layers?

ResNets: Problem setting

Is learning better networks as easy as stacking more layers?

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

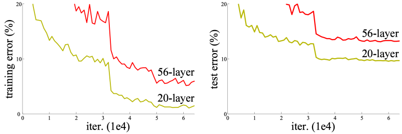


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

ResNets: Problem setting

Is learning better networks as easy as stacking more layers?

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

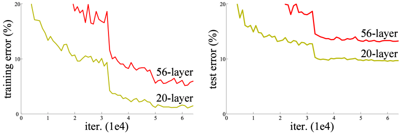


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

No! Adding more layers decreases accuracy for both test & train.
Why?

Vanishing Gradients

What happens to the gradients if you build a very deep network?

$$\frac{\partial L(\mathbf{W})}{\partial w_{ij}^{(k)}} = \frac{\partial L}{\partial v^{(L)}} \frac{\partial v^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial v^{(L-1)}} \cdots \frac{\partial v^{(k)}}{\partial z^{(k)}} \frac{\partial z^{(k)}}{\partial w_{ij}^{(k)}}$$

Vanishing Gradients

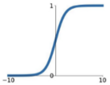
What happens to the gradients if you build a very deep network?

$$\frac{\partial L(\mathbf{W})}{\partial w_{ij}^{(k)}} = \frac{\partial L}{\partial v^{(L)}} \frac{\partial v^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial v^{(L-1)}} \cdots \frac{\partial v^{(k)}}{\partial z^{(k)}} \frac{\partial z^{(k)}}{\partial w_{ij}^{(k)}}$$

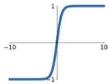
Causes of vanishing gradients

- Deep nets e.g. $k << L$
- "Saturated" activations
- poor initialization, etc

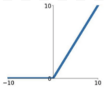
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



tanh
$\tanh(x)$



ReLU
$\max(0, x)$



ResNets: Framework to train super deep networks

- Add skip connections
- More stable gradients through connections
- Only changes the forward pass

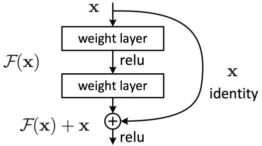


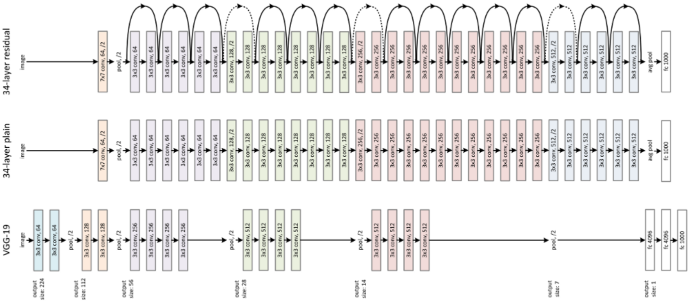
Figure 2. Residual learning: a building block.

Let $H(x) = F(x) + x$

$$\frac{\partial}{\partial x} H(x) = \frac{\partial}{\partial x} F(x) + 1$$

A ResNet module $F(x)$ need only model the **residual** $H(x) - x$

ResNets: Architecture



ResNets: Performance

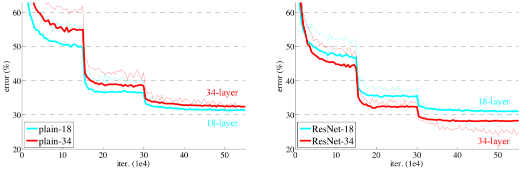


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

model	top-1 err.	top-5 err.
VGG-16 [41]	28.07	9.33
GoogLeNet [44]	-	9.15
PReLU-net [13]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

method			error (%)
Maxout [10]			9.38
NIN [25]			8.81
DSN [24]			8.22
	# layers	# params	
FitNet [35]	19	2.5M	8.39
Highway [42, 43]	19	2.3M	7.54 (7.72±0.16)
Highway [42, 43]	32	1.25M	8.80
ResNet	20	0.27M	8.75
ResNet	32	0.46M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97
ResNet	110	1.7M	**6.43** (6.61±0.16)
ResNet	1202	19.4M	7.93

Table 6. Classification error on the **CIFAR-10** test set. All methods are with data augmentation. For ResNet-110, we run it 5 times and show "best (mean±std)" as in [43].

Demo

- Training a ResNet requires a **lot** of resources
- But the model itself is small and can be loaded onto a laptops

HW2: Problem 1

Solving for $\mathbf{w}_{ols}$:

$$\begin{aligned}
\hat{R}(\mathbf{w}) &= \sum_{i=1}^{n} \left(y_i - \mathbf{w}^T \mathbf{x}_i\right)^2 && (1) \\
&= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) && (2) \\
&= \mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} - 2\mathbf{y}^T \mathbf{X}\mathbf{w} + \mathbf{y}^T \mathbf{y} && (3)
\end{aligned}$$

Compute gradient:

$$\frac{\partial}{\partial \mathbf{w}} \hat{R}(\mathbf{w}) = 2\mathbf{X}^T \mathbf{X}\mathbf{w} - 2\mathbf{X}^T \mathbf{y}$$

Set to 0:

$$\mathbf{w}_{ols} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y}$$

HW2: Problem 1 cont.

Let $U\Sigma V^T$ be the SVD of X. We need:

- U, V orthonormal: $U^T = U^{-1}$, $U^T U = \mathbf{I}$
- $(AB)^{-1} = B^{-1}A^{-1}$

What is $\mathbf{w}_{ols}$?

$$\begin{aligned}
\mathbf{w}_{ols} &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} && (1)\\
&= \left(\mathbf{V}\Sigma\mathbf{U}^T\mathbf{U}\Sigma\mathbf{V}^T\right)^{-1}\mathbf{V}\Sigma\mathbf{U}^T\mathbf{y} && (2)\\
&= \left(\mathbf{V}\Sigma^2\mathbf{V}^T\right)^{-1}\mathbf{V}\Sigma\mathbf{U}^T\mathbf{y} && (3)\\
&= \mathbf{V}\Sigma^{-2}\mathbf{V}^T\mathbf{V}\Sigma\mathbf{U}^T\mathbf{y} && (4)\\
&= \mathbf{V}\Sigma^{-2}\Sigma\mathbf{U}^T\mathbf{y} && (5)\\
&= \mathbf{V}\Sigma^{-1}\mathbf{U}^T\mathbf{y} && (6)
\end{aligned}$$

HW2: Problem 3

The ridge penalty term, $\lambda w^T w$

(a) shrinks the low variance components.

Σ is a diagonal matrix that contains the singular values of X

- $d_j = \Sigma_{jj}$ correspond to the stddev of feature j

$$\mathbf{w}_{ridge} = \mathbf{V}\left(\Sigma^2 + \lambda \mathbf{I}\right)^{-1}\Sigma \mathbf{U}^T \mathbf{y}$$
$$\mathbf{w}_{ols} = \mathbf{V}\Sigma^{-1}\mathbf{U}^T \mathbf{y}$$

Since Σ is diagonal we can write:

$$\mathbf{X}\mathbf{w}_{ols} = \mathbf{U}\Sigma\mathbf{V}^T\mathbf{V}\Sigma^{-1}\mathbf{U}^T\mathbf{y} = \mathbf{U}\mathbf{U}^T\mathbf{y} = \sum_j \mathbf{u}_j \mathbf{u}_j^T \mathbf{y}$$

$$\mathbf{X}\mathbf{w}_{ridge} = \mathbf{U}\Sigma\left(\Sigma^2 + \lambda \mathbf{I}\right)^{-1}\Sigma\mathbf{U}^T\mathbf{y} = \sum_j \mathbf{u}_j \frac{d_j^2}{d_j^2 + \lambda}\mathbf{u}_j^T \mathbf{y}$$

See Elements of statistical learning p. 66 for more details.

HW2: Problem 10 Is the variance of $\mathbf{w}$ less than $\mathbf{w}_{ridge}$?

Define $\Sigma_\lambda = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)$. Σ_λ is symmetric: $\left[\Sigma_\lambda^{-1}\right]^T = \Sigma_\lambda^{-1}$
From 8: $Var\left[\mathbf{w}\right] = \sigma^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$
From 9: $Var\left[\mathbf{w}_{ridge}\right] = \sigma^2\Sigma_\lambda^{-1}\left(\mathbf{X}^T\mathbf{X}\right)\Sigma_\lambda^{-1}$

$$\Delta Var = Var\left[\mathbf{w}\right] - Var\left[\mathbf{w}_{ridge}\right]$$

$Var\left[\mathbf{w}\right] \succeq Var\left[\mathbf{w}_{ridge}\right] \implies \Delta Var \succeq 0$
$A \succeq 0$ iff A is non-negative definite.

HW2: Problem 10 cont.

$$\Sigma_\lambda = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)$$

$$\begin{aligned}
\Delta Var &= Var\left[\mathbf{w}\right] - Var\left[\mathbf{w}_{ridge}\right] && (1)\\
&= \sigma^2\left[\left(\mathbf{X}^T\mathbf{X}\right)^{-1} - \Sigma_\lambda^{-1}\left(\mathbf{X}^T\mathbf{X}\right)\Sigma_\lambda^{-1}\right] && (2)\\
&= \sigma^2\left[\Sigma_\lambda^{-1}\Sigma_\lambda\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\Sigma_\lambda\Sigma_\lambda^{-1} - \Sigma_\lambda^{-1}\left(\mathbf{X}^T\mathbf{X}\right)\Sigma_\lambda^{-1}\right] && (3)\\
&= \sigma^2\Sigma_\lambda^{-1}\left[\Sigma_\lambda\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\Sigma_\lambda - \mathbf{X}^T\mathbf{X}\right]\Sigma_\lambda^{-1} && (4)\\
&= \sigma^2\Sigma_\lambda^{-1}\left[\mathbf{X}^T\mathbf{X} + 2\lambda\mathbf{I} + \lambda^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1} - \mathbf{X}^T\mathbf{X}\right]\Sigma_\lambda^{-1} && (5)\\
&= \sigma^2\Sigma_\lambda^{-1}\left[2\lambda\mathbf{I} + \lambda^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\right]\Sigma_\lambda^{-1} && (6)\\
&\succeq 0 && (7)
\end{aligned}$$