EECS 182 Deep Neural Networks
Spring 2023 Anant Sahai

Note 3: Regularization

1 Pragmatic view of regularization

- By adding the penalty term, we hope to shape the optimization to favor something that we like.
- Note that (explicit) regularization does not change the expressive power of the network. Therefore, you only change what the optimization is favoring.

2 Regularization in Least Squares

For a more intuitive understanding of regularization, we can look at least squares. Let's start by reviewing the problem: we have a matrix X and vector $\mathbf{y}$, and we want to calculate the weights $\mathbf{w}$ that best approximates $X\mathbf{w} = \mathbf{y}$.

Ordinary Least Squares (OLS)

Problem: $\text{argmin}_w \; ||\mathbf{y} - X\mathbf{w}||^2$

Solution: $\hat{\mathbf{w}} = \left(X^T X\right)^{-1} X^T \mathbf{y}$

We can add regularization to OLS. One version of regularized OLS is called ridge regression:

Ridge Regression

The standard form of regularization is

$$L_\theta = \frac{1}{n}\sum_{i=1}^{n} \ell_{\text{train}}\big(y_i, f_\theta(x_i)\big) + R(\theta).$$

As an example, observe the least squares with ridge regression. The task is finding w which gives $y \approx Xw$, where $X \in \mathbb{R}^{n\times d}$, $w \in \mathbb{R}^d$, and $y \in \mathbb{R}^n$:

$$\text{Optimization Problem: } \text{argmin}_w \; ||\mathbf{y} - X\mathbf{w}||^2 + \lambda||\mathbf{w}||^2$$

$$\hat{w} = \big(X^\mathsf{T}X + \lambda\mathbf{I}\big)^{-1}X^\mathsf{T}y. \tag{1}$$

λ is the **regularization parameter**. It penalizes high-magnitude weight vectors (can you see why?). We can select different λ values to control the behavior of Ridge regression. When λ is higher, the weight magnitude

penalty is more severe; when it's lower, the penalty is lighter, which allows Ridge regression to settle on weight vectors with higher magnitudes.

Note that when $\lambda = 0$, ridge regression becomes unregularized. This unregularized formulation simplifies to the OLS problem and solution. Intuitively, setting $\lambda = 0$ tells ridge regression to disregard the weight vector's magnitude altogether, allowing it to solve the problem as though it were simply OLS.

3 Perspective 1: Gradient Descent

We can also solve least squares with **gradient descent**. Gradient descent is a process where we aim to improve our model by repeatedly moving its parameters in the direction that minimizes the cost function.

The gradient for least squares without regularization is

OLS Gradient Descent

$$\text{Gradient (with step size): } \eta 2X^T\left(\mathbf{y} - X\hat{\mathbf{w}}\right)$$

$$\text{Update step: } \hat{\mathbf{w}}_{(t+1)} = \hat{\mathbf{w}}_{(t)} + \eta 2X^T\left(\mathbf{y} - X\hat{\mathbf{w}}_{(t)}\right)$$

$$\begin{aligned}\hat{\mathbf{w}}_{(t+1)} &= \text{Weights } \hat{\mathbf{w}} \text{ at timestep } t+1\\ \hat{\mathbf{w}}_{(t)} &= \text{Weights } \hat{\mathbf{w}} \text{ at timestep } t\\ \eta &= \text{Step size}\\ \mathbf{y} - X\hat{\mathbf{w}}_{(t)} &= \text{Residual at timestep } t\end{aligned}$$

We can create a similar update step for ridge regression:

Ridge Regression Gradient Descent

$$\text{Gradient (with step size): } = \eta\left(2X^T\left(\mathbf{y} - X\hat{\mathbf{w}}\right) - 2\lambda\hat{\mathbf{w}}\right)$$

$$\text{Update step: } \hat{\mathbf{w}}_{(t+1)} = \left(1 - 2\eta\lambda\right)\hat{\mathbf{w}}_{(t)} + \eta 2X^T\left(\mathbf{y} - X\hat{\mathbf{w}}_{(t)}\right)$$

$1 - 2\eta\lambda$, which is called the "weight decay", is between 0 and 1 with small η and λ. If the second term of is zero (or close to 0), the weight w decays to 0 exponentially fast.

Note 1. What if $\lambda \gg 1$? Then, the optimization problem above weights $||w||^2$ heavily and makes $||w||$ smaller (moving w closer to the origin) when minimizing the objective function.

Note 2. One of the common mistakes in practice is doing both (a) adding the explicit ridge type regularizer to the loss function and (b) enabling weight decay option. This is a redundancy, and you may get more of "weight decay" effect than you originally wanted.

4 Perspective 2: Leveraging SVD for better understanding

Regarding linear algebra and Euclidean norm, a very powerful tool is observing in some different coordinate systems to see more clearly what is going on. In particular, in $\|\cdot\|_2$, the SVD is very useful.

The full SVD $X \in \mathbb{R}^{n\times d}$ can be written as

$$X = U\Sigma V^\mathsf{T}$$

where $U \in \mathbb{R}^{n\times n}$, $\Sigma \in \mathbb{R}^{n\times d}$, $V \in \mathbb{R}^{d\times d}$, U and V are orthonormal matrices, and Σ is "diagonal".

4.1 Ordinary Least Squares

$$\begin{aligned}
Xw \approx y \quad &\Leftrightarrow \quad U\Sigma V^\mathsf{T} w \approx y \\
&\Leftrightarrow \quad \Sigma V^\mathsf{T} w \approx U^\mathsf{T} y \\
&\Leftrightarrow \quad \Sigma\tilde{w} \approx \tilde{y} \quad (\text{Changing coordinate : } \tilde{w} = V^\mathsf{T} w,\ \tilde{y} = U^\mathsf{T} y) \\
&\Leftrightarrow \quad \begin{bmatrix} \sigma_1 & & \mathbf{0} \\ & \ddots & & 0_{(d-n)\times n} \\ \mathbf{0} & & \sigma_n \end{bmatrix} \tilde{w} \approx \tilde{y} \quad (\text{Assume } n < d) \\
&\Leftrightarrow \quad \tilde{w}_i = \begin{cases} \frac{1}{\sigma_i}\tilde{y}_i & : i \le n \\ 0 & : \text{o.w.} \end{cases}.
\end{aligned}$$

The last relation comes from approximating $\Sigma\tilde{w} \approx \tilde{y}$ using the min-norm

$$\tilde{w} = \Sigma^T(\Sigma\Sigma^T)^{-1}\tilde{y} = \begin{bmatrix} \sigma_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \sigma_d \\ & 0_{(n-d)\times d} & \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_1^2} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \frac{1}{\sigma_d^2} \end{bmatrix} \tilde{y} = \begin{bmatrix} \frac{1}{\sigma_1} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \frac{1}{\sigma_d} \\ & 0_{(n-d)\times d} & \end{bmatrix} \tilde{y}$$

If instead $n \ge d$, then Σ is a tall matrix instead, which gives, using the least squares solution of $\Sigma\tilde{w} \approx \tilde{y}$ approximated by $\tilde{w} = (\Sigma^T\Sigma)^{-1}\Sigma^T\tilde{y}$,

$$\begin{bmatrix} \sigma_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \sigma_d \\ & 0_{d\times(n-d)} & \end{bmatrix} \tilde{w} \approx \tilde{y} \quad (\text{Assume } n \ge d) \Leftrightarrow \tilde{w}_i = \frac{1}{\sigma_i}\tilde{y}_i, i \le d$$

Both cases above can be combined for brevity as

$$\tilde{w}_i = \begin{cases} \frac{1}{\sigma_i}\tilde{y}_i & : i \leq \min(n,d) \\ 0 & : \text{o.w.} \end{cases}.$$

4.2 Ridge regression

With similar observation as the OLS case, since U and V are orthonormal matrices, the ridge regression of the form

$$||\mathbf{y} - X\mathbf{w}||^2 + \lambda||\mathbf{w}||^2$$

is equivalent to

$$\|\tilde{y} - \Sigma\tilde{w}\|_2^2 + \lambda\|w\|_2^2.$$

This optimization problem can be decoupled into n scalar ridge problems for each w_i, since Σ is "diagonal". Substituting the SVD simplification into the Ridge solution,

$$\begin{aligned} \hat{w} = \left(\Sigma^\mathsf{T}\Sigma + \lambda\mathbf{I}\right)^{-1}\Sigma^\mathsf{T}\tilde{y} &= \begin{bmatrix} \frac{1}{\sigma_1^2+\lambda} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \frac{1}{\sigma_d^2+\lambda} \end{bmatrix} \begin{bmatrix} \sigma_1 & & \mathbf{0} & \\ & \ddots & & 0_{d\times(n-d)} \\ \mathbf{0} & & \sigma_d & \end{bmatrix} \tilde{y} \\ &= \begin{bmatrix} \frac{\sigma_1}{\sigma_1^2+\lambda} & & \mathbf{0} & \\ & \ddots & & 0_{d\times(n-d)} \\ \mathbf{0} & & \frac{\sigma_d}{\sigma_d^2+\lambda} & \end{bmatrix} \tilde{y} \end{aligned}$$

Then, by solving each of n problems, the solution is (with the same reasoning as above using the least squares solutions):

$$\hat{w}_i = \begin{cases} \frac{\sigma_i}{\sigma_i^2+\lambda}\tilde{y}_i & : i \leq \min(n,d) \\ 0 & : \text{o.w.} \end{cases}.$$

From here, let's observe the role of λ.

- High level perspective on closed form solution (1): even though $X^\mathsf{T}X$ is not invertible or barely invertible, adding $\lambda\mathbf{I}$ makes $X^\mathsf{T}X + \lambda\mathbf{I}$ invertible.
- If $\lambda \ll \sigma_i$, then we have

  $$\hat{w}_i \sim \frac{1}{\sigma_i}y_i$$

  which gives the usual solution from ordinary least squares. When $\lambda \gg \sigma_i$ instead, we obtain

  $$\hat{w}_i \sim \frac{\sigma_i}{\lambda}y_i$$

  and we prevent the case of blowup for small eigenvalues, creating overfitting and/or solutions that are difficult to believe. Ridge essentially makes weights associated with smaller singular values close to 0, while leaving larger singular values which give the system the greatest variance, unchanged. In many

such systems, in practice, we have a wide, dynamic range of singular values, so this is a common technique.

- Note: Ridge in this sense is similar to projecting the data onto a low dimensional space of the higher singular values, but has the benefit of not requiring the computational expense of finding the SVD explicitly.
- Graphically, on a log scale, λ changes the singular values as such:

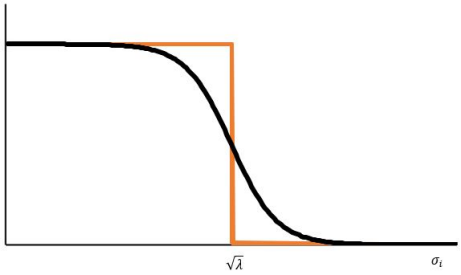

Figure 1: Image that depicts the σ_i's coefficient in Ridge regression solution. The goal is to have a sharp decrease to higher sigma values so Ridge regression (black) well approximates a step function (orange).

4.3 Gradient Descent

Looking at GD in SVD coordinates, we have the update in between w_{t-1} and w_t in the form

$$w_t = w_{t-1} + 2\eta\Sigma^T(\tilde{y} - \Sigma\tilde{w}_t)$$

looking at each coordinate, this update becomes

$$w_t[i] = w_{t-1}[i] + 2\eta\sigma_i(\tilde{y}[i] - \sigma_i\tilde{w}_t[i])$$

This gives the interpretation that for large singular values, gradient descent moves much more and faster than smaller singular values, which can take longer to converge. Thus, we first fit in the largest σ_i direction, then the next and so on. When working with very small singular values that have the propensity to give large solutions, we may converge to something very large, but slowly, especially without regularization.

- Solution: Use gradient descent, but limit the number of steps to a finite number to avoid overtraining. This is sometimes called early stopping.

5 Data Regularization

There are a few different tricks that can be used to avoid explictly including Ridge regularization while still accomplishing the same result, either by adding extra entries or extra features to the data matrix.

5.1 Adding Extra Data

We can augment the data matrix X and the vector y as follows:

$$\hat{X} = \begin{bmatrix} X \\ \sqrt{\lambda} I_d \end{bmatrix}, \hat{y} = \begin{bmatrix} y \\ 0_d \end{bmatrix}$$

Using this new data in the ordinary least squares solution, we obtain

$$\hat{w} = (\hat{X}^T \hat{X})^{-1} \hat{X}^T \hat{y} = (X^T X + \lambda I_d)^{-1} X^T y$$

which gives the exact same solution as Ridge regression.

5.2 Adding Extra Features

Instead of adding extra data, we can instead add additional features to make our matrix wide and have infinitely many solutions.

$$\hat{X} = \begin{bmatrix} X & \sqrt{\lambda} I_n \end{bmatrix}$$

Here, if we solve the min-norm problem

$$\min_{\hat{w}: \hat{X}\hat{w} = y} ||\hat{w}||^2$$

Then, we have the closed-form solution of the min-norm problem given by

$$\begin{bmatrix} \hat{w} \\ f \end{bmatrix} = \hat{X}^T (\hat{X}\hat{X}^T)^{-1} y = \begin{bmatrix} X^T \\ \sqrt{\lambda} I_n \end{bmatrix} (XX^T + \lambda I_n)^{-1} y = (X^T X + \lambda I_d)^{-1} \begin{bmatrix} X^T \\ \sqrt{\lambda} I_n \end{bmatrix} y$$

where the first n entries give the same solution as Ridge regression once again.

6 Implicit Regularization

Implicit regularization is the regularization that occurs when we aren't consciously doing any regularization. When performing explicit regularization as above, we must specify a specific regularization hyperparameter and modify the training data, model architecture, or loss function. In contrast, implicit regularization is an unexpected benefit stemming from our choice of optimizer. We will see that choosing gradient descent as our optimization algorithm, combined with the large size of DNNs, provides enough regularization for DNNs to generalize well without us intentionally restricting the parameters.

To gain intuition, let's look at gradient descent (GD) updates for OLS in SVD coordinates:

$$\widetilde{\boldsymbol{w}}_{t+1} = \widetilde{\boldsymbol{w}}_t + 2\eta \Sigma^{(\widetilde{\boldsymbol{y}} - \Sigma \widetilde{\boldsymbol{w}}_t)},$$

where $\widetilde{\boldsymbol{w}}_t$ represents the parameter vector at the current step, $\widetilde{\boldsymbol{w}}_{t+1}$ represents the updated parameter vector, Σ represents the diagonal matrix of singular values from the above SVD, $\widetilde{\boldsymbol{y}} :=^{\boldsymbol{y}}$ as defined above, and η is our learning rate hyperparameter.

Note that because we are dealing with a diagonal matrix, this works out to updating each component of the weight vector individually. They don't interact with each other at all during GD, so each is being modified as follows:

$$\widetilde{\boldsymbol{w}}_{t+1}[i] = \widetilde{\boldsymbol{w}}_t[i] + 2\eta\sigma_i \left(\widetilde{\boldsymbol{y}}[i] - \sigma_i \widetilde{\boldsymbol{w}}_t[i]\right).$$

This is potentially unstable because we are not reducing $\widetilde{\boldsymbol{w}}_t[i]$ at each step as we did in the last lecture. This means that, subject to a bounded input, we might get an unbounded output if we allow the algorithm to run forever. Observe that the stationary point is the solution we discussed earlier, $\widetilde{\boldsymbol{w}}[i] = \frac{1}{\sigma_i}\widetilde{\boldsymbol{y}}[i]$. If σ_i is tiny, then we find ourselves in a bad situation.

Let's carefully calculate the first few steps of GD to see what's going on:

$$\begin{aligned}
\widetilde{\boldsymbol{w}}_0[i] &= 0 \\
\widetilde{\boldsymbol{w}}_1[i] &= 2\eta\sigma_i\widetilde{\boldsymbol{y}}[i] \\
\widetilde{\boldsymbol{w}}_2[i] &= 2\eta\sigma_i\widetilde{\boldsymbol{y}}[i] + 2\eta\sigma_i \left(\widetilde{\boldsymbol{y}}[i] - \sigma_i 2\eta\sigma_i\widetilde{\boldsymbol{y}}[i]\right) \approx 4\eta\sigma_i\widetilde{\boldsymbol{y}}[i].
\end{aligned}$$

Observe that this is roughly a linear function with an extremely small slope if σ_i is tiny. In this situation, GD barely moves in the early stages even though it will eventually converge to a very large value, as previously discussed. Together with early stopping, this means that GD is trying to do something like ridge regularization for us because it will resist enlarging the directions corresponding to small singular values. Early stopping is when we stop the training process because validation performance has gotten worse or has not improved for a long time. It is important to note that GD, when initialized at zero, will converge to the minimum-norm solution. This is a good exercise for the reader to verify.

To summarize, there are three kinds of regularization: explicit regularization, data augmentation (adding fake observations or features), and implicit regularization (optimizer has an implicit regularizing effect). Regarding DNNs, the combination of the min-norm seeking behavior of gradient descent and the feature augmentation that is implicit when using large networks gives a lot of regularization even if we weren't thinking about it.

7 Taking Back to Neural Networks

We are often interested in minimizing the loss function of a neural network $f_\theta(x)$, but we need to ensure that the optimization algorithms can produce the minimization we want. With nonlinear problems, we can often linearize around the operating point by taking

$$f_\theta(x) = f_{\theta_0}(x) + \frac{\partial f}{\partial x}\Big|_{\theta_0} \cdot \Delta\theta$$

and linearize around every point as part of a Generalized Linear Model (GLM). In minimizing the loss function, we have the following characteristics:

- Gradient Descent is the tool that we understand the most and is often our first choice.
- The loss function will typically contain training data with labels that we would like to predict with good accuracy.

- The singular values of the data matrices involved greatly matter and can lead to overfitting, slow convergence, and unreasonably large parameters.
- In solving this problem, we usually add regularization, which we can understand through SVD as bounding the solution to the smaller singular values while leaving the solution of the larger singular values unchanged.

