EECS 182 Deep Neural Networks
Spring 2023 Anant Sahai

Discussion 5

1. Residual Connection

Neural networks learn by optimization with gradients. The deeper a neural network grows, the more likely it will suffer from either vanishing gradients or exploding gradients problem since downstream layers (ie. earlier layers) will be optimized with either exponentially small or large gradients due to chain rule. Exploding gradients can be resolved using **gradient clipping**, whereas tackling vanishing gradients is more tricky. Here we study how skip connections can help.

(a) Vanishing gradient results from taking the product of many gradients with norm less than 1. One reason is due to saturating activation functions such as sigmoid. Recall the form of sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$. **What's the maximum gradient that sigmoid can generate, ie. the maximum of $\frac{d}{dx}\sigma(x)$?** Based on what you computed, **why is ReLU more preferable than sigmoid in deep neural nets**?

Solution:

$$\begin{aligned}\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left(\frac{1}{1+e^{-x}}\right) = \frac{1}{(1+e^{-x})^2}\cdot e^{-x} \\ &= \frac{1}{1+e^{-x}}\cdot\frac{e^{-x}}{1+e^{-x}} \\ &= \sigma(x)\cdot(1-\sigma(x))\end{aligned} \tag{1}$$

We know that sigmoid function output falls in the range of (0, 1) and thus the maxium gradient of sigmoid is $\frac{1}{4}$ and the minimum is 0. The maximum can be obtained by solving $\frac{d}{d\sigma}\sigma\cdot(1-\sigma)=0$.

The maximum and minimum gradient value of ReLU, as seen in previous discussions, is 0 and 1 respectively, which makes it more desirable in deep neural nets than sigmoid because even in the non-saturating region of sigmoid (as covered in lectures, sigmoid saturates on both its tails quickly), the max value is still less than one, and multiplying many of these together inevitably leads to exponentially small gradient norms. For ReLU, vanishing gradients is empirically less problematic due to the gradient value of 1 on the positive end, while parameters can still become stuck at zero sometimes, often refered as the "dying ReLU" problem.

(b) One common way to tackle vanishing gradients is by adding residual connections. Fig 1 shows a simplified example of a residual block. Assuming that the weights are affine layer with zero biases, **what's the expression of X_1 in terms of W_1, W_2 and X_0?** $W_i \in \mathbb{R}^{n\times n},\ X_i \in \mathbb{R}^n$.

Solution:

$$X_1 = \text{ReLU}(W_2\text{ReLU}(W_1X_0) + X_0)$$

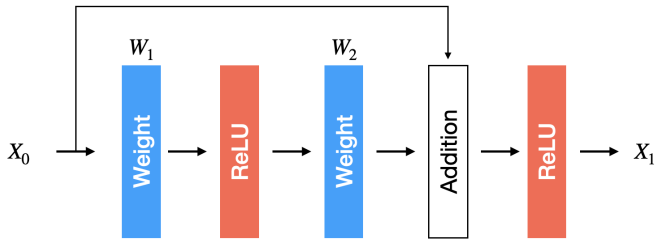

Figure 1: Diagram for Residual Block

(c) **Compute the gradient** $\frac{\partial X_1}{\partial X_0}$. Based on what you see numerically, **why does residual connection preserves gradient norms better**?

Solution:

Since there are two possiblities for ReLU gradient outputs, there are 3 possible answers:

- When the last ReLU layer outputs zero, $\frac{\partial X_1}{\partial X_0}$ is zero.
- When the both ReLU layers output 1, $\frac{\partial X_1}{\partial X_0} = W_2W_1 + I$.
- When the first ReLU layer outputs 0, and $X_0 > 0$, the gradient is just identity.

By enumerating the possiblities, we see that as long as ReLU does not "die", there's always an identity term to perserve the norm of the gradient, subsequently avoiding the vanishing gradient problem.

2. Regularization and Dropout

Recall that linear regression optimizes the following learning objective:

$$\mathcal{L}(\mathbf{w}) = ||\mathbf{y} - X\mathbf{w}||_2^2 \tag{2}$$

One way of using **dropout** during SGD on the d-dimensional input features $\mathbf{x}_i$ involves keeping each feature at random $\sim_{i.i.d}$ $Bernoulli(p)$ (and zeroing it out if not kept) and then performing a SGD step.

It turns out that such dropout makes our learning objective effectively become

$$\mathcal{L}(\check{\mathbf{w}}) = E_{R\sim Bernoulli(p)}\left[||\mathbf{y} - (R \odot X)\check{\mathbf{w}}||_2^2\right] \tag{3}$$

where $\odot$ is the element-wise product and the random binary matrix $R \in \{0,1\}^{n\times d}$ is such that $R_{i,j} \sim_{i.i.d}$ $Bernoulli(p)$. We use $\check{\mathbf{w}}$ to remind you that this is learned by dropout.

Recalling how Tikhonov-regularized (generalized ridge-regression) least-squares problems involve solving:

$$\mathcal{L}(\mathbf{w}) = ||\mathbf{y} - X\mathbf{w}||_2^2 + ||\Gamma\mathbf{w}||_2^2 \tag{4}$$

for some suitable matrix Γ, it turns out we can manipulate (3) to eliminate the expectations and get:

$$\mathcal{L}(\check{\mathbf{w}}) = ||\mathbf{y} - pX\check{\mathbf{w}}||_2^2 + p(1-p)||\check{\Gamma}\check{\mathbf{w}}||_2^2 \tag{5}$$

with $\check{\Gamma}$ being a diagonal matrix whose j-th diagonal entry is the norm of the j-th column of the training matrix X.

(a) **How should we transform the $\check{\mathbf{w}}$ we learn using (5) (i.e. with dropout) to get something that looks a solution to the traditionally regularized problem (4)?**

(Hint: This is related to how we adjust weights learned using dropout training for using them at inference time. PyTorch by default does this adjustment during training itself, but here, we are doing dropout slightly differently with no adjustments during training.)

Solution: Choose $\mathbf{w} = p\check{\mathbf{w}}$. I.e., we need to multiply $\check{\mathbf{w}}$ by p.

The idea here is to absorb the p term into $\mathbf{w}$, and then choose Γ accordingly.

By choosing $\mathbf{w} = p\check{\mathbf{w}}$ (and thus $\check{\mathbf{w}} = \frac{1}{p}\mathbf{w}$), we would have

$$\begin{aligned}
\mathcal{L}(w) &= ||y - pX\check{\mathbf{w}}||_2^2 + p(1-p)||\check{\Gamma}\check{\mathbf{w}}||_2^2 && (6)\\
&= ||y - X\mathbf{w}||_2^2 + p(1-p)||\check{\Gamma}\frac{\mathbf{w}}{p}||_2^2 && (7)\\
&= ||y - X\mathbf{w}||_2^2 + ||\sqrt{\frac{(1-p)}{p}}\check{\Gamma}\mathbf{w}||_2^2 && (8)\\
&= ||y - X\mathbf{w}||_2^2 + ||\Gamma\mathbf{w}||_2^2 && (9)
\end{aligned}$$

Where we chose $\Gamma = \sqrt{\frac{1-p}{p}}\check{\Gamma}$

(b) With the understanding that the Γ in (4) is an invertible matrix, **change variables in (4) to make the**

problem look like classical ridge regression:

$$\mathcal{L}(\widetilde{\mathbf{w}}) = ||\mathbf{y} - \widetilde{X}\widetilde{\mathbf{w}}||_2^2 + \lambda||\widetilde{\mathbf{w}}||_2^2 \tag{10}$$

Explicitly, what is the changed data matrix $\widetilde{X}$ in terms of the original data matrix X and Γ?

Solution: To make the regularization term in (4) look like ridge regression, we'll let $\widetilde{\mathbf{w}} = \Gamma\mathbf{w}$, so $\mathbf{w} = \Gamma^{-1}\widetilde{\mathbf{w}}$. Plugging this into (4) gives us

$$||\mathbf{y} - X\Gamma^{-1}\tilde{\mathbf{w}}||_2^2 + ||\tilde{\mathbf{w}}||_2^2 \tag{11}$$

From this, we see our modified data matrix should be

$$\tilde{X} = X\Gamma^{-1}.$$

(c) Continuing the previous part, with the further understanding that Γ is a *diagonal* invertible matrix with the j-th diagonal entry proportional to the norm of the j-th column in X, **what can you say about the norms of the columns of the effective training matrix $\widetilde{X}$** and **speculate briefly on the relationship between dropout and batch-normalization.**

Solution:

Let's say each x_j, $\tilde{x}_j$ are the j-th column vector of X and $\tilde{X}$, respectively. Recall that Γ is defined to be a diagonal matrix whose j-th diagonal entry is the norm of the j-th column of X and $\tilde{X} = cX\check{\Gamma}^{-1}$, where c is a rescaling positive constant of that you found in the previous part.

$$\tilde{X} = cX\check{\Gamma}^{-1} = \begin{bmatrix}\tilde{x}_1 & \tilde{x}_2 & \dots & \tilde{x}_d\end{bmatrix} = c\begin{bmatrix}x_1 & x_2 & \dots & x_d\end{bmatrix}\begin{bmatrix}\frac{1}{\|x_1\|_2} & 0 & \dots & 0 \\ 0 & \frac{1}{\|x_2\|_2} & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & \frac{1}{\|x_d\|_2}\end{bmatrix}$$

$$= \begin{bmatrix}c\frac{x_1}{\|x_1\|_2} & c\frac{x_2}{\|x_2\|_2} & \dots & c\frac{x_d}{\|x_d\|_2}\end{bmatrix}$$

Therefore, $\tilde{x}_j = c\frac{x_j}{\|x_j\|_2}$ and $\left\|\tilde{x}_j\right\|_2 = c$. In other words, dropout makes each column vector of the matrix X constant-norm of c in effect, where c is appropriate rescaling constant from part $2a$.

Note that this is similar to batch-normalization's standardization and scaling operations, which makes the column vectors have the same variance.

Contributors:

- Kevin Li .
- Saagar Sanghavi.
- Jerome Quenum.
- Anant Sahai.

