CSC413 Homework 3 March 13, 2021

1. Robustness and Regularization

   1.1. Adversarial Examples

      1.1.1. Bounding FGSM (Optional)

      1.1.2. Prediction under Attack
      Let's find $\boldsymbol{x}^{'}$ first.

$$\begin{aligned}\boldsymbol{x}^{'} &= \boldsymbol{x} - \epsilon\nabla_{\boldsymbol{x}} f(\boldsymbol{x};\boldsymbol{w})\\ &= \boldsymbol{x} - \epsilon\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{w}^T\boldsymbol{x})\\ &= \boldsymbol{x} - \epsilon\boldsymbol{w}\end{aligned}$$

Thus, the model output under the adversarial attack should be

$$\begin{aligned}f(\boldsymbol{x}^{'};\boldsymbol{w}) &= \boldsymbol{w}^T\boldsymbol{x}^{'}\\ &= \boldsymbol{w}^T(\boldsymbol{x}-\epsilon\boldsymbol{w})\\ &= \boldsymbol{w}^T\boldsymbol{x}-\epsilon\boldsymbol{w}^T\boldsymbol{w}\\ &= \boldsymbol{w}^T\boldsymbol{x}-\epsilon||\boldsymbol{w}||_2^2\end{aligned}$$

   1.2. Gradient Descent and Weight Decay

      1.2.1. Toy Example (Optional)

      1.2.2. Closed Form Ridge Regression Solution
      Similar to the process of finding the solution of the plain regression, we firstly need to minimize the objective of linear regression with regularization by finding its derivative. Thus, we have

$$\begin{aligned}&\frac{\partial}{\partial \boldsymbol{w}}(\frac{1}{2n}||X\boldsymbol{w}-\boldsymbol{t}||_2^2+\lambda||\boldsymbol{w}||_2^2)\\ =&\frac{1}{n}X^T(X\boldsymbol{w}-\boldsymbol{t})+2\lambda\boldsymbol{w}\\ =&\frac{1}{n}X^TX\boldsymbol{w}-\frac{1}{n}X^T\boldsymbol{t}+2\lambda\boldsymbol{w}\\ =&(\frac{1}{n}X^TX+2\lambda\boldsymbol{I}_d)\boldsymbol{w}-\frac{1}{n}X^T\boldsymbol{t}\end{aligned}$$

By setting the previous equation to 0, we can get

$$\begin{aligned}(\frac{1}{n}X^TX+2\lambda\boldsymbol{I}_d)\boldsymbol{w}-\frac{1}{n}X^T\boldsymbol{t}=0\\ \implies (X^TX+2n\lambda\boldsymbol{I}_d)\boldsymbol{w}-X^T\boldsymbol{t}=0\\ \implies (X^TX+2n\lambda\boldsymbol{I}_d)\boldsymbol{w}=X^T\boldsymbol{t}\\ \implies \boldsymbol{w}^*_{ridge}=(X^TX+2n\lambda\boldsymbol{I}_d)^{-1}X^T\boldsymbol{t}\end{aligned}$$

Thus, the solution to ridge regression is $\boldsymbol{w}^*_{ridge}=(X^TX+2n\lambda\boldsymbol{I}_d)^{-1}X^T\boldsymbol{t}$.

      1.2.3. Adversarial Attack under Weight Decay
      According to 1.1.2 and 1.2.2, we can get

$$\begin{aligned}f(x^{'};w^*_{ridge}) &= w^*_{ridge}x-\epsilon {w^*_{ridge}}^2\\ &= (X^TX+2n\lambda)^{-1}X^T\boldsymbol{t}x-\epsilon((X^TX+2n\lambda)^{-1}X^T)^2\end{aligned}$$

By setting the previous equation to 0, we can get

$$\begin{aligned}
f(x^{'}; w^*_{ridge}) &= 0 \\
\implies (X^TX + 2n\lambda)^{-1}X^T\boldsymbol{t}x - \epsilon((X^TX + 2n\lambda)^{-1}X^T\boldsymbol{t})^2 &= 0 \\
\implies x - \epsilon(X^TX + 2n\lambda)^{-1}X^T\boldsymbol{t} &= 0 \\
\implies \epsilon(X^TX + 2n\lambda)^{-1}X^T\boldsymbol{t} &= x \\
\implies \epsilon &= x(X^T\boldsymbol{t})^{-1}(X^TX + 2n\lambda) \\
\implies \epsilon &= \frac{x(X^TX + 2n\lambda)}{X^T\boldsymbol{t}}
\end{aligned}$$

Since ϵ is affected by the weight decay coefficient λ, we can conclude that it makes the model more robust under the FGM adversarial attack. To be specific, if we do not have any weight decay, the ϵ would be $\frac{xX^TX}{X^T\boldsymbol{t}}$, which is under control. By adding the weight decay, we can set its coefficient to minimize the effect of the attack so that the model could still be stable.

1.2.4. The Adversary Strikes Back (Optional)

2. Trading off Resources in Neural Net Training

2.1. Effect of batch size

2.1.1. Batch size vs. learning rate
The optimal learning rate should also be increased as the batch size increases. As we increase the batch size, the minibatch gradient would become more stable, which results in the lower variance among all gradients. Thus, to minimize the number of steps, we need to increase the learning rate to speed up the process.

2.1.2. Training steps vs. batch size

(a) The point C would be the best choice. Although the point B requires relatively less training steps, the batch size is almost 2^n, which is not efficient in calculations. For the point A, although the batch size is small, it needs much more training steps, which takes more times.

(b) **Point A:** Regime: noise dominated. Potential way to accelerate training: seek parallel compute.
Point B: Regime: curvature dominated. Potential way to accelerate training: higher order optimizers.

2.2. Model size, dataset size and compute
C is the best option. Although larger dataset size or more compute resources could rapidly improve the performance in some points according to Figure 3, it would always be converged when it's large enough. However, according to Figure 4, we can see adding the number of parameters would always get better performance regardless of its batch size as well as the number of training steps.

3. Dropout as Gaussian noise

3.1. Warm-up: linear regression with input dropout
According to lecture 6, we have

$$\begin{aligned}
\mathbb{E}_m[\mathcal{J}] &= \frac{1}{2N}\sum_{i=1}^{N}\mathbb{E}_m[(\tilde{y}^{(i)} - t^{(i)})^2] \\
&= \frac{1}{2N}\sum_{i=1}^{N}(\mathbb{E}_m[\tilde{y}^{(i)}] - t^{(i)})^2 + \frac{1}{2N}\sum_{i=1}^{N}Var_m[\tilde{y}^{(i)}]
\end{aligned}$$

For the mean of $\tilde{y}^{(i)}$, we can get

$$\begin{aligned}
\mathbb{E}_m[\tilde{y}^{(i)}] &= \mathbb{E}_m[\frac{1}{p}\sum_j m_j^{(i)} w_j x_j^{(i)}] \\
&= \frac{1}{p}\sum_j \mathbb{E}_m[m_j^{(i)}]\mathbb{E}_m[w_j x_j^{(i)}] && \text{since } m \text{ and } x \text{ are independent} \\
&= \frac{1}{p}\sum_j p\mathbb{E}_m[y_j^{(i)}] \\
&= \sum_j y_j^{(i)} \\
&= y^{(i)}
\end{aligned}$$

The variance of $\tilde{y}^{(i)}$ should be

$$\begin{aligned}
Var_m[\tilde{y}^{(i)}] &= Var_m[\frac{1}{p}\sum_j m_j^{(i)} w_j x_j^{(i)}] \\
&= \frac{1}{p^2}\sum_j Var_m[m_j^{(i)} w_j x_j^{(i)}] && \text{since inputs are independent} \\
&= \frac{1}{p^2}\sum_j Var_m[m_j^{(i)}](w_j x_j^{(i)})^2 \\
&= \frac{p(1-p)}{p^2}\sum_j (w_j x_j^{(i)})^2 \\
&= \frac{1-p}{p}\sum_j (w_j x_j^{(i)})^2
\end{aligned}$$

3.2. Multiplicative Gaussian noise
By the bias-variance decomposition, we have

$$\begin{aligned}
\mathbb{E}_\pi[\mathcal{J}] &= \frac{1}{2N}\sum_{i=1}^{N} \mathbb{E}_\pi[(\tilde{y}^{(i)} - t^{(i)})^2] \\
&= \frac{1}{2N}\sum_{i=1}^{N} (\mathbb{E}_\pi[\tilde{y}^{(i)}] - t^{(i)})^2 + \frac{1}{2N}\sum_{i=1}^{N} Var_\pi[\tilde{y}^{(i)}]
\end{aligned}$$

For the mean of $\tilde{y}^{(i)}$, we can get

$$\begin{aligned}
\mathbb{E}_\pi[\tilde{y}^{(i)}] &= \mathbb{E}_\pi[\sum_j (1+\pi^{(i)}) w_j x_j^{(i)}] \\
&= \sum_j (1+\mathbb{E}_\pi[\pi^{(i)}])\mathbb{E}_\pi[w_j x_j^{(i)}] && \text{since } m \text{ and } x \text{ are independent} \\
&= \sum_j \mathbb{E}_\pi[y_j^{(i)}] \\
&= \sum_j y_j^{(i)} \\
&= y^{(i)}
\end{aligned}$$

The variance of $\tilde{y}^{(i)}$ should be

$$\begin{aligned}
Var_\pi[\tilde{y}^{(i)}] &= Var_\pi[\sum_j (1+\pi^{(i)}) w_j x_j^{(i)}] \\
&= \sum_j Var_\pi[(1+\pi^{(i)}) w_j x_j^{(i)}] \qquad \text{since inputs are independent} \\
&= \sum_j Var_\pi[\pi^{(i)}](w_j x_j^{(i)})^2 \\
&= \sigma^2 \sum_j (w_j x_j^{(i)})^2
\end{aligned}$$

We can see that the mean is same as the one for 3.1.. From 3.1, we know $Var_m[\tilde{y}^{(i)}] = \frac{1-p}{p} \sum_j (w_j x_j^{(i)})^2$. Thus, if we set $\sigma = \sqrt{\frac{1-p}{p}}$, the multiplicative Gaussian noise with zero mean would be equivalent to applying input dropout with probability $1-p$.