1. Robustness and Regularization

1.1 Adversarial Examples.

(1.1.2) Prediction under attack. $w \in R^n$, $x \in R^n$

$$\nabla_x f(x, w) = \frac{\partial w^T x}{\partial x} = w$$

$$\Rightarrow \quad x' \leftarrow x - \epsilon \cdot w = x - \epsilon w$$

$$\Rightarrow \quad f(x'; w) = w^T x' = \boxed{w^T (x - \epsilon w)}$$

1.2 Gradient Descent and Weight Decay

(1.2.2) Closed form Ridge Regression Solution (ignore $\frac{1}{2n}$ for convenience).

$$\min_w \|Xw - t\|_2^2 + \lambda \|w\|_2^2$$

Expand $\Rightarrow$ $(Xw - t)^T (Xw - t) + \lambda w^T w$

$$= (w^T X^T - t^T)(Xw - t) + \lambda w^T w$$

$$= (w^T X^T X w - w^T X^T t - t^T X w + t^T t) + \lambda w^T w$$

take derivative wrt. w $\Rightarrow$ $2X^T X w - 2X^T t + 2\lambda w = 0$.

$$(X^T X + \lambda I) w = X^T t$$

$$\Rightarrow \boxed{w^*_{ridge} = (X^T X + \lambda I)^{-1} X^T t}$$

(1.2.3) Adversarial Attack under weight decay 1D: $X \in R^n$, $x \in R$

Given: $w^*_{ridge} = (X^T X + \lambda)^{-1} X^T t$ $\quad f(x'; w) = w^T (x - \epsilon w)$

$$\Rightarrow f(x'; w^*_{ridge}) = \left[(X^T X + \lambda)^{-1} X^T t\right]^T \left[x - \epsilon (X^T X + \lambda)^{-1} X^T t\right] = 0.$$

$$x - \epsilon (X^T X + \lambda)^{-1} X^T t = 0. \Rightarrow x = \epsilon (X^T X + \lambda)^{-1} X^T t$$

$$\boxed{\epsilon = \frac{x (X^T X + \lambda)}{X^T t}}$$

$\Rightarrow$ As $\lambda > 0$, the weight decay will make ϵ larger than the normal $\epsilon = \frac{x X^T X}{X^T t}$ which means it will perturb the data more and more robust to fool the model.

2. Trade off resources in Neural Net training

2.1 Effect of batch size

2.1.1 Batch size vs. Learning rate

$$g_B(w) = \frac{1}{B}\sum_{i=1}^{B} g_i(w), \quad \text{where } g_i(w) = \nabla\mathcal{L}(w) + \epsilon_i, \quad \epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2).$$

From the minibatch gradient formula, it clearly shows that the noisy version of gradient is closer to the true gradient $\nabla L(w)$ as the batch size B increases. which indicates more 'accurate' gradient can be generated by increasing the batch size. Therefore, the optimal learning rate will also increase with batch sizes since it can take large steps without losing accuracy and also accelerate the training time.

2.1.2 Training steps vs. Batch size

(a) C is the most efficient batch size among other two points since A requires much more training time to finish the entire process since the batch size is too small and the results may contain a lot noises; B requires much more resources since larger batch size requires more computational power which will induce more cost and slower convergence.

(b) Point A: noise dominated ⇒ seek parallel compute.
Point B: Curvature dominated. ⇒ use higher order optimizers

2.2 Model size, data size and compute

(a) C would be the best option. By looking at the right figure in Figure 2

and Figure 3, it clearly shows that the optimal model size is increasing with more compute and increasing model size will significantly increase the model performance. "A" may induce overfitting since same model with more training steps may make the test loss getting higher; "B" does not make sense since the fine-tuned learning rate in smaller batch sizes does not fit for larger batch size due to its inefficiency.

3. Dropout at Gaussian Noise

3.1 Linear regression with input dropout

$$\tilde{y}_m^{(i)} = \frac{1}{p}\sum_j m_j^{(i)} w_j x_j^{(i)}, \qquad \text{where } m_j^{(i)} \overset{\text{i.i.d.}}{\sim} \text{Ber}(p).$$

$$E_m[J] = \frac{1}{2N}\sum_{i=1}^{N} E_m[(\tilde{y}_m^i - t^i)^2]$$

$$= \frac{1}{2N}\sum_{i=1}^{N} E_m[\tilde{y}_m^{i\,2} - 2\tilde{y}_m^i t^i + t^{i\,2}]$$

$$= \frac{1}{2N}\sum_{i=1}^{N} E_m(\tilde{y}_m^{i\,2}) - \frac{1}{2N}\sum_{i=1}^{N} E_m[2\tilde{y}_m^i t^i] + \frac{1}{2N}\sum_{i=1}^{N} t^{i\,2}$$

$$= \frac{1}{2N}\sum_{i=1}^{N} \text{Var}[\tilde{y}_m^i] + \frac{1}{2N}\sum_{i=1}^{N} E[\tilde{y}_m^i]^2 - \frac{1}{2N}\sum_{i=1}^{N} E_m[2\tilde{y}_m^i t^i] + \frac{1}{2N}\sum_{i=1}^{N} t^{i\,2}$$

$$= \frac{1}{2N}\sum_{i=1}^{N} (E_m[\tilde{y}_m^i] - t^i)^2 + \frac{1}{2N}\sum_{i=1}^{N} \text{Var}[\tilde{y}_m^i] \qquad ①$$

$$\text{Var}[\tilde{y}_m^i] = \sum_j \text{Var}\left[\frac{1}{p} m_j^i w_j x_j^i\right]$$

$$\downarrow$$

$$p(1-p)$$

then, ① can be written as:

$$= \frac{1}{2N}\sum_{i=1}^{N} (E_m[\tilde{y}_m^i] - t^i)^2 + \frac{(1-p)}{2p}\sum_j \text{Var}[X_j] w_j^2 \qquad ②$$

3.2 Multiplicative Gaussian noise.

$$\tilde{y}_{\pi}^{(i)} = \sum_j (1 + \pi_j^{(i)}) w_j x_j^{(i)}, \qquad \text{where } \pi_j^{(i)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2).$$

Similar to ①: $\frac{1}{2N} \sum_{i=1}^{N} \left(E_\pi[\tilde{y}_\pi^{(i)}] - t^{(i)} \right)^2 + \frac{1}{2N} \sum_{i=1}^{N} \mathrm{Var}[\tilde{y}_\pi^{(i)}]$ ③

first term: ②: $E[\tilde{y}_m^{(i)}] = E\left[\frac{1}{p} \sum_j m_j^{(i)} w_j x_j^{(i)}\right] = \sum_j w_j x_j^{(i)}$

since $E_m[m_j^{(i)}] = p$

③: $E[\tilde{y}_\pi^{(i)}] = E\left[\sum_j (1 + \pi_j^{(i)}) w_j x_j^{(i)}\right] = \sum_j w_j x_j^{(i)}$

since $E_\pi[1 + \pi_j^{(i)}] = E_\pi[1 + 0] = 1$

second term: $\frac{1}{2N} \sum_{i=1}^{N} \sum_j \mathrm{Var}[(1 + \pi_j^{(i)}) w_j x_j^{(i)}]$

$\mathrm{Var}[1 + \pi_j^{(i)}] = \sigma^2$ $\Rightarrow = \frac{\sigma^2}{2} \sum_j \mathrm{Var}[x_j] w_j^2$

therefore, ③ can be written as.

$$\frac{1}{2N} \sum_{i=1}^{N} \left(E_\pi[\tilde{y}_\pi^{(i)}] - t^{(i)} \right)^2 + \frac{\sigma^2}{2} \sum_j \mathrm{Var}[x_j] w_j^2 \quad ④$$

Compare ③ and ④: when $\boxed{\sigma = \sqrt{\frac{1-p}{p}}}$

$\Rightarrow$ multiplicative gaussian noise is same as applying input dropout and they share the same structure of bia-variance decomposition.