Training and Generalization

- Overfitting, Underfitting, and Goldilocks Fitting
- Training, Validation, and Testing Data Sets
- Model Order, Model Capacity, Generalization Loss
Training and Generalization

- **Goal:**
  - Learn the “true relationship” from training data pairs \((x_k, y_k)|_{k=0}^{K-1}\).
    \[ x = f_\theta(y) + \text{error} \]
  - What we learn needs to *generalize* beyond the training data.

- **Key parameters:**
  - \( P = \textbf{Model Order} = \text{number of parameters} = \text{Dimension of } \theta \in \mathbb{R}^P \)
  - \( N_x \times K = \# \text{ training points} = (\text{Dimension of } x) \times (\# \text{ of training pairs}) \)

- **Key issues**
  - If \( P \gg N_x \times K \): Model order is too high and there is a tendency to over fit.
  - If \( P \ll N_x \times K \): Model order is too low, and there is a tendency to under fit
Overfitting

- Training data

- Overfitting

- Model order too high
- Doesn’t generalize well
Underfitting

- Training data

- Underfitting
  - Model order too low
  - Doesn’t generalize well
Goldilocks Fitting

- Training data

- Best fitting

  - Model order “just right”
  - Best generalization
Partitioning of Labeled Data

- Let \((x_k, y_k)\) for \(k \in S = \{0, \cdots, K - 1\}\) be the full set data.
  - \(y_k\) is the input data.
  - \(x_k\) is the label or “ground truth” data.

- Typically, we randomly partition\(^*\) the data into three subsets:
  - \(S_T\) is the training data
  - \(S_V\) is the validation data
  - \(S_E\) is the testing (evaluation) data

\(^*\) Note that “partition” means \(S = S_T \cup S_V \cup S_E\) and \(\emptyset = S_T \cap S_V = S_T \cap S_E = S_V \cap S_E\)

- For each partition, we define a loss function:

\[
L_T(\theta) = \frac{1}{|S_T|} \sum_{k \in S_T} \|y_k - f_\theta(x_k)\|^2
\]

\[
L_V(\theta) = \frac{1}{|S_V|} \sum_{k \in S_V} \|y_k - f_\theta(x_k)\|^2
\]

\[
L_E(\theta) = \frac{1}{|S_E|} \sum_{k \in S_E} \|y_k - f_\theta(x_k)\|^2
\]
Roles of Data

- **Training data:**
  - Only data used to train model
  \[
  \theta^* = \arg\min_{\theta} \{L_T(\theta)\} = \arg\min_{\theta} \left\{ \frac{1}{K} \sum_{k \in S_T} \|y_k - f_\theta(x_k)\|_2^2 \right\}
  \]

- **Validation data:**
  - Used to compare models of different order.

- **Testing data**
  - Used for final evaluation of model performance.
Loss Function Convergence

- Loss vs. iterations of gradient-based optimization

Notice:
- As training continues, the model is overfit to the data
- Best to stop training when $L_V$ is at a minimum
- Model order is too high, but early termination of training can help fix problem
Loss vs. Model Order vs. # Training Pairs

- Loss vs. model order

\[ L_V \text{ - validation loss} \]

\[ L_T \text{ - training loss} \]

- Loss vs. # of training pairs

More training data is always better, but slower.
What are $L_T$ and $L_V$ telling you?

- Model order/model capacity may be too low…

- Model order/model capacity may be too high…
Never Test on Training Data!

- Never report training loss, $L_T$, as your ML system accuracy!
  - This is like doing a homework problem after you have seen the solution.
  - The network has “memorized” the answers.

- Don’t even report validation loss, $L_V$, as your ML system accuracy.
  - This is also biased by the fact that your tuned model order parameters.

- Only report testing loss, $L_E$, as your ML system accuracy.
  - This data is sequestered to ensure it is an unbiased estimate of loss.
1. Early termination

2. Regularization
   - $L_2$ and $L_1$ weight regularization
   - Loss function is modified to be
     \[ \tilde{L}(\theta) = L(\theta) + \beta S(\theta) \]
   - $\beta$ larger $\Rightarrow$ less overfitting

3. Dropout Method: Next slide
The Dropout Method*

- Drop nodes with probably $p \approx 0.2$
- Scale all node outputs by $p$:
  - To compute loss for validation and test
  - During inference

During Training: Done independently for each batch

During Validation, Testing, and Inference

Stochastic Gradient Descent

- Batches and Epochs
- Learning Rate and Momentum
- Weight initialization and regularization
- Dropout method
Batches, Epochs, and Stochastic Gradient Descent (SGD)

- Partition training set into randomized batches

\[ S = \{1, \cdots, K\} = \bigcup_{b=1}^{B} S_b \]

\[ K_b = |S_b| = (\text{# of sampler per batch}) \]

- For each batch you compute a separate gradient

\[ \nabla L(\theta; S_b) \iff \text{Gradient for } b^{th} \text{ batch of training data} \]

\[
\begin{align*}
\text{Repeat until converged } \{ \\
\text{Repeat } b = 1 \text{ to } B \{ \\
\quad d &\leftarrow -\nabla L(\theta; S_b) \\
\quad \theta &\leftarrow \theta + \alpha d^t \\
\} \\
\} \\
\end{align*}
\]

One epoch

\[ \text{One batch} \]

\[ \text{Stochastic Gradient Descent (SGD)} \]
Theoretical Analysis of SGD

- Assume simple sampling (sampling with replacement)

\[ g_k = \nabla L_k(\theta) = \text{(gradient from } k^{th} \text{ training sample)} \]

- Each sample, \( g_{ki} \), is i.i.d. with distribution \( p(g) = \frac{\text{histogram}(g)}{K} \)

True gradient:

\[ g = \frac{1}{K} \sum_{k=0}^{K-1} g_k \]

Batch gradient:

\[ \hat{g} = \frac{1}{K_b} \sum_{i=0}^{K_b-1} g_{ki} \]

- Then

\[ \hat{g} = g + \frac{w}{\sqrt{K_b}} \]

where

\[ E[w] = 0 \]

\[ \text{Var}[w] = \frac{1}{K} \sum_{k=0}^{K-1} (g_k - g)(g_k - g)^t \]
Effect of Batch Size on SGD

Then we have that:

- As \( K_b \to \infty \) (batch size goes up) \( \Rightarrow \) noise decreases
- As \( K_b \to 0 \) (batch size goes down) \( \Rightarrow \) noise increases

\[
\hat{g} = g + \frac{W}{\sqrt{K_b}}
\]

True gradient:
\[
g = \frac{1}{K} \sum_{k=0}^{K-1} g_k
\]

Batch gradient:
\[
\hat{g} = \frac{1}{K} \sum_{i=0}^{K_{b-1}} g_{k_i}
\]
**Effect of Gradient Noise: Exploration**

\[ \hat{g} = g + \frac{W}{\sqrt{K_b}} \]

- **Batch Gradient**
- **True Gradient**
- **Noise**

### Smooth Function
- \( g(\theta) \) very good
- \( \hat{g}(\theta) \) good

### Bumpy Function
- \(-g(\theta)\) very bad
- \(-\hat{g}(\theta)\) better
Effect of Gradient Noise: Exploitation

\[ \hat{g} = g + \frac{w}{\sqrt{K_b}} \]

Perfect

\[ g(\theta) \]

True Gradient

Batch Gradient

Not so good

\[ \hat{g}(\theta) \]

Noise

Smooth Function
Why SGD works?
- The gradient for a small batch is much faster to compute and almost as good as the full gradient.
- If $K = 10,000$ and $K_b = |S_b| = 32$, then one iteration of SGD is approximately $\frac{10,000}{32} \approx 312$ times faster than GD.

Batch size
- Larger batches: less “noise” in gradient $\Rightarrow$
  - *Worse*: slower updates; less exploration.
  - *Better*: better local convergence.
- Smaller batches: more “noise” in gradient $\Rightarrow$
  - *Worse*: hunts around local minimum.
  - *Better*: faster updates; better exploration.

Patch size:
- Many algorithms train on image “patches”
- Apocryphal: Smaller patches speed training. Not true!!!!

Step size $\alpha$
- Too large $\Rightarrow$ hunts around local minimum
- Too small $\Rightarrow$ slow convergence
Momentum

- **SGD with momentum**
  - $\alpha$ is step size, and $\gamma$ is momentum typically with $\gamma = 0.9$

```plaintext
init v ← 0
Repeat until converged {
    Repeat $n = 0$ to $K_b - 1$
    {
        $d ← -\nabla L(\theta; S_b)$
        $v ← \gamma v + \alpha d$
        $\theta ← \theta + v^t$
    }
}
```

- **Interpretation**
  - $\theta$ is like position
  - $v$ is like velocity
  - Friction $= 1 - \gamma$
Momentum

- **SGD with momentum**
  - $\alpha$ is step size, and $\gamma$ is momentum typically with $\gamma = 0.9$


\[
\text{init } v \leftarrow 0 \\
\text{Repeat until converged} \{ \\
\quad \text{Repeat } n = 0 \text{ to } K_b - 1 \{ \\
\quad\quad d \leftarrow -\nabla L(\theta; S_b) \\
\quad\quad v \leftarrow \gamma v + \alpha d \\
\quad\quad \theta \leftarrow \theta + v^t \\
\quad \}
\}\n\]

*Stochastic Gradient Descent (SGD) with momentum*

- **Interpretation**
  - $\theta$ is like position
  - $v$ is like velocity
  - Friction $= 1 - \gamma$
**Interpretation of Momentum**

- Special case of impulsive input: If $d_n = \delta_n$

  ```plaintext
  init $\nu \leftarrow 0$
  Repeat $n = 0$ to $K_b - 1$
  
  $\nu \leftarrow \gamma \nu + \alpha \delta_n$
  $\theta \leftarrow \theta + \nu^t$
  ```

  Momentum

- Then

  \[
  \theta_n = \theta_0 + \left( \frac{\alpha}{1 - \gamma} \right) (1 - \gamma^n) 
  \]

  ![Graph showing $d_n$, $\theta_n$, and the asymptotic value](image)

  - Asymptotic value $= \frac{\alpha}{1 - \gamma}$
  - Time constant $= -1/\log \gamma$
Intuition

Friction = 1 − γ

Time constant = −1/ log γ
Nesterov Momentum*

- SGD with momentum
  - $\alpha$ is step size, and $\gamma$ is momentum typically with $\gamma \approx 0.9$

\[
\text{init } v \leftarrow 0 \\
\text{Repeat until converged } \{ \\
\quad \text{Repeat } n = 0 \text{ to } K_b - 1 \{ \\
\quad\quad d \leftarrow -\nabla L(\theta + \gamma v^t; S_b) \\
\quad\quad v \leftarrow \gamma v + \alpha d \\
\quad\quad \theta \leftarrow \theta + v^t \\
\quad \} \\
\}\]

\text{Stochastic Gradient Descent (SGD) with Nesterov momentum}

- Intuition:
  - Even if $d_n = 0$, we have that $\theta_{n+1} = \theta_n + \gamma v^t$ because of momentum
  - So compute the gradient at $\theta_n + \gamma v^t$

ADAM (Adaptive Moment Estimation)*

- SGD with ADAM optimization = Momentum + Preconditioning

\[
\text{init } v \leftarrow 0; \ r \leftarrow 0; \\
\text{init } t \leftarrow 0 \\
\text{Repeat until converged } \{ \\
\text{Repeat } n = 0 \text{ to } K_b \{ \\
\quad t \leftarrow t + 1 \\
\quad d \leftarrow -\nabla L(\theta; S_b) \\
\quad v \leftarrow \beta_1 v + (1 - \beta_1)d \\
\quad r \leftarrow \beta_2 r + (1 - \beta_2)d^2 \\
\quad \hat{v} \leftarrow v / (1 - \beta_1^t) \\
\quad \hat{r} \leftarrow r / (1 - \beta_2^t) \\
\quad \theta \leftarrow \theta + \alpha (\sqrt{\hat{r}} + \epsilon)^{-1} \hat{v} \\
\}\} \\
\}
\]

- Typical parameters: \(\alpha = 0.001; \beta_1 = 0.9; \beta_2 = 0.999; \epsilon = 10^{-8}\)

ADAM (Adaptive Moment Estimation)*

- SGD with ADAM optimization = Momentum + Preconditioning

\[
\text{init } v \leftarrow 0; \quad r \leftarrow 0; \\
\text{init } t \leftarrow 0 \\
\text{Repeat until converged} \\
\quad \text{Repeat } n = 0 \text{ to } K_b \\
\quad \quad t \leftarrow t + 1 \\
\quad \quad d \leftarrow -\nabla L(\theta; S_b) \\
\quad \quad v \leftarrow \beta_1 v + (1 - \beta_1)d \\
\quad \quad r \leftarrow \beta_2 r + (1 - \beta_2)d^2 \\
\quad \quad \hat{v} \leftarrow v / (1 - \beta_1^t) \\
\quad \quad \hat{r} \leftarrow r / (1 - \beta_2^t) \\
\quad \quad \theta \leftarrow \theta + \alpha \left( \sqrt{\hat{r}} + \epsilon \right)^{-1} \hat{v} \\
\] \\
\text{ADAM Optimization}

- Typical parameters: \( \alpha = 0.001; \beta_1 = 0.9; \beta_2 = 0.999; \epsilon = 10^{-8} \)