Data C182 Designing, Visualizing & Understanding DNN
Fall 2024 Eric Kim, Naveen Ashish
Discussion 01

This discussion worksheet/note contains information from future lectures.

Welcome to Data C182 - we're excited to have you here and have some *deep* conversations with you! This discussion will cover some statistics review.

1. Class Logistics

Welcome to the first discussion!

- The goal of the sections/discussions is to provide useful supplemental information to the main lecture
- There will be a mix of practical skills discussions and theoretical discussion

List of Discussion Schedules is available on the course website. If you have requests on topics we should include in any future discussions, please let us know.

More importantly, please familiarize yourselves with class logistics available on our class website.

Read through the syllabus on the class website, and answer the following questions:

(a) What times and where will lectures happen?

(b) When and where is the midterm?

(c) How much slip days will you be given?

(d) Can you use slip days for the final project?
TODO: TODO: Add solution

2. Extra Derivations

1 Regularizations

1.1 Notation

- Model parameters: θ
- Loss function: ℓ, e.g., ℓ_2 loss: $\ell = \|f(x) - y\|_2^2$
- Supervised dataset: X, Y
- Risk: $R(\theta) = \mathbb{E}[\ell(\theta; X, Y)]$
- Empirical risk: $\hat{R}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(\theta; x_i, y_i)$

1.2 Maximum Likelihood Estimation (MLE)

The goal of MLE is to find the model parameters that maximize the likelihood of observing the data. This can be written as:

$$\theta_{\text{MLE}} = \arg\max_{\theta} \prod_{i=1}^{N} P_\theta(y_i|x_i).$$

Since the product of probabilities can be very small, it is common to maximize the log-likelihood instead:

$$\theta_{\text{MLE}} = \arg\max_{\theta} \sum_{i=1}^{N} \log P_\theta(y_i|x_i).$$

Maximizing the log-likelihood is equivalent to minimizing the negative log-likelihood (NLL).

1.3 Maximum A Posteriori (MAP) Estimation

MAP estimation incorporates a prior distribution over the model parameters. The goal is to find the model parameters that maximize the posterior probability of the parameters given the data. This can be written as:

$$\begin{aligned}
\theta_{\text{MAP}} &= \arg\max_{\theta} P(\theta|Y, X) \\
&= \arg\max_{\theta} P(Y|\theta, X)P(\theta) \\
&= \arg\max_{\theta} \log P(Y|\theta, X) + \log P(\theta) \\
&= \arg\max_{\theta} \sum_{i=1}^{N} \log P_\theta(y_i|x_i) + \log P(\theta).
\end{aligned}$$

The term $\log P(\theta)$ is the log-prior. Maximizing the posterior is equivalent to minimizing the NLL plus a regularizer, which is derived from the log-prior.

1.4 Gaussian Prior and ℓ_2 Regularization

A common choice for the prior is a Gaussian distribution:

$$P(\theta) = \mathcal{N}(\theta; 0, \sigma^2 I),$$

where σ^2 is the variance and I is the identity matrix. The log-prior is then:

$$\begin{aligned}
\log P(\theta) &= \log \left[\frac{1}{(2\pi)^{d/2}|\sigma^2 I|^{1/2}} \exp\left(-\frac{1}{2}\theta^T (\sigma^2 I)^{-1} \theta \right) \right] \\
&= -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log\Big(|\sigma^2 I|\Big) - \frac{1}{2\sigma^2}\theta^T\theta
\end{aligned}$$

$$= -\frac{1}{2\sigma^2}\|\theta\|_2^2 + \text{const.},$$

where d is the dimensionality of θ. Therefore, the regularizer is:

$$-\log P(\theta) = \frac{1}{2\sigma^2}\|\theta\|_2^2 + \text{const.} = \lambda\|\theta\|_2^2,$$

where $\lambda = \frac{1}{2\sigma^2}$. This is the ℓ_2 regularization term.

2 Bias-Variance Trade-off

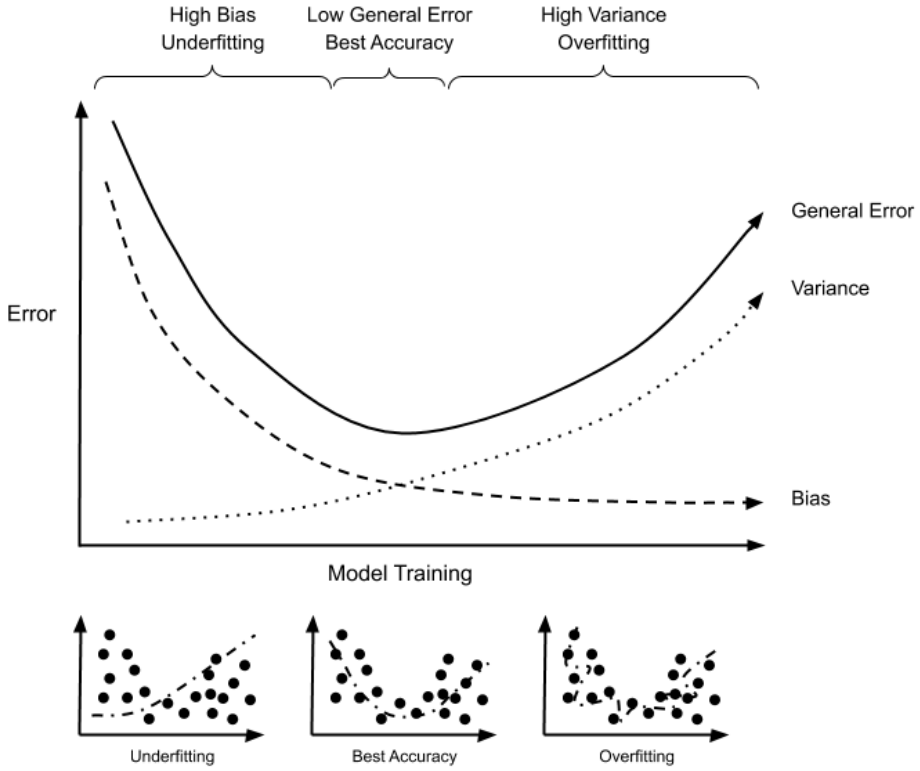

Figure 1: The bias-variance trade-off.

Notation

- $f(x)$: Ideal, true function
- $f_\theta(x)$: Learned, ML model
- $\varepsilon \sim \mathcal{N}(0, \sigma^2)$: Label noise (assumed to be Gaussian)
- $y|x = f(x) + \varepsilon$: Data label = true label + noise
- $f_D(x)$: ML model learned from dataset D
- $\bar{f}(x) = \mathbb{E}_D[f_D(x)]$: "Average" learned model over possible datasets, capturing the capacity of learning

The relationship between input, true function, and observed data can be viewed probabilistically:

$$x \to f(x) \to y.$$

Assuming Gaussian noise, the negative log-likelihood (NLL) loss can be written as:

$$\begin{aligned}
-\log P_\theta(y_i|x_i) &= -\log\left[\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_i - f_\theta(x_i))^2}{2\sigma^2}\right)\right] \\
&= \frac{1}{2}\log\Big(2\pi\sigma^2\Big) + \frac{(y_i - f_\theta(x_i))^2}{2\sigma^2} \\
&= \frac{1}{2\sigma^2}(f_\theta(x_i) - y_i)^2 + \text{const.}
\end{aligned}$$

This shows that under the Gaussian noise assumption, minimizing the NLL loss is equivalent to minimizing the squared ℓ_2 distance between the predicted values ($f_\theta(x_i)$) and the target values (y_i).

3 Deriving Bias-Variance Tradeoff

- $\theta(D)$: MLE estimate of θ on D
- x', y': test data point
- $\mathbb{E}_D[(f_D(x') - y')^2]$: expected loss on test data, over D

$$\begin{aligned}
\mathbb{E}_D[(f_D(x') - y')^2] &= \mathbb{E}_D[(f_D(x') - f(x') + f(x') - y')^2] \\
&= \mathbb{E}_D[(f_D(x') - f(x'))^2] + \mathbb{E}_D[(f(x') - y')^2] \\
&\quad + \mathbb{E}_D[2(f_D(x') - f(x'))(f(x') - y')] \\
&= \mathbb{E}_D[(f_D(x') - f(x'))^2] + \sigma^2 \\
&\quad + \mathbb{E}_D[2\underbrace{(f_D(x')f(x') - f_D(x')y')}_{0} + \underbrace{(y'f(x') - f(x')^2)}_{0}] \\
&= \mathbb{E}_D[((f_D(x') - \bar{f}(x') + \bar{f}(x') - f(x'))^2] + \sigma^2 \\
&= \mathbb{E}_D[(f_D(x') - \bar{f}(x'))^2] + (\bar{f}(x') - f(x'))^2 \\
&\quad + \underbrace{2\mathbb{E}_D[(f_D(x') - \bar{f}(x'))(\bar{f}(x') - f(x'))]}_{0} + \sigma^2 \\
&= \underbrace{\mathbb{E}_D[(f_D(x') - \bar{f}(x'))^2]}_{\text{Variance}} + \underbrace{(\bar{f}(x') - f(x'))^2}_{\text{Bias}} + \underbrace{\sigma^2}_{\text{Irreducible error}}
\end{aligned}$$

3. Machine Learning Overview

Formulating Learning Problems In this course, we will discuss 2 main types of learning problems:

- Supervised Learning

- Unsupervised Learning

In supervised learning, you are given a dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ containing input vectors and labels, and attempt to learn $f_\theta(\cdot)$ such that $f_\theta(x)$ approximates the true label y.

In unsupervised learning, your dataset is unlabeled, and $\mathcal{D} = \{x_1, \ldots, x_n\}$, and you attempt to learn properties of the underlying distribution of $\mathcal{D}$.

Solving Machine Learning Problems To solve a machine learning problem, you must first define three "parameters".

(a) Pick a model class (for example, do you want to use logistic regression or do you want to use a deep neural network?)

(b) Pick a loss function (how do you want to determine the "badness" of your model performance?)

(c) Pick your optimizer (how are you going to optimize your model parameters θ to minimize the loss?)

Then, you typically run this on a big CPU or GPU.

Dataset Splits During Training In the case when hyper-parameter tuning is possible (e.g., learning rate of deep nets), in addition to training and test sets, you should hold out a validation set. The following policies should be taken when using training/validation/test sets:

(a) Why should you never tune your hyperparameters on your test set?

(b) What should your validation set be used for?

(c) Describe a general ML workflow with datasets

4. Statistics Review

(a) Let $\mathbb{P}(H)$ be the probability you have a headache, and $\mathbb{P}(F)$ be the probability you have a flu. Given the following data:

Headache	Flu
N	N
Y	N
N	N
Y	Y
Y	Y
N	Y

Calculate $\mathbb{P}(F)$, $\mathbb{P}(H)$, $\mathbb{P}(H|F)$. Then, calculate $\mathbb{P}(F|H)$ using Bayes' Theorem.

(b) Bias and variance of estimators

In statistics, we often observe $X \sim P_\theta$ where P_θ is a class of probability distribution parameterized by θ. Here, X is the data and observed, and θ is a parameter and unobserved. Then, the goal of estimation is the following:

$$\text{We observe } X \sim P_\theta \text{ and estimate the value of some estimand } g(\theta)$$

Definition 1 (Statistic). A statistic is any function $T(X)$ of the observed data X.

Definition 2 (Estimator). Estimator $f_\theta(X)$ are rules to calculate an estimate of some function of observed data. In other words, an estimator is any statistic meant to guess an estimand $g(\theta)$. We also often use the "hat" notation, $\hat{Y}$ to denote an estimator.

For example, a common estimator of the population mean is the sample mean defined by: $\bar{X} = \frac{1}{N}\sum_{i=1}^{n} X_i$.

Definition 3 (Bias and Variance of Estimator). Bias of an estimator is a measure of how much does the expected value of the estimator differ from the true distribution. Suppose we have a randomly sampled training set $\mathcal{D}$ and a test input x, and we select an estimator denoted $\theta = \hat{\theta}(\mathcal{D})$.

Understand the bias-variance tradeoff

i. Decompose the total variance of an estimator as a sum of bias and variance.

ii. How do the bias and variance relate to overfitting and underfitting? (*HINT: See lecture 3*)

(c) According to the derivation, we can see that the best estimator, should have low bias and low variance. So why don't we always use an unbiased estimator?

5. Function Approximation & Risk Functions

There is a lot of hype surrounding deep neural networks, but at their core they are just ways of learning functions. For example, in the case of classification, we try to learn $\mathbb{P}(y|x)$, that is the probability our true label is some class y given input features x. In the case of regression, it's a similar continuous response variable. In the case of generative models, we are trying to learn to approximate a whole distribution. In all of the cases, we are trying to find an estimator $f_\theta(x)$ of a true distribution y.

To find $f_\theta(x)$, we must adjust the weights and biases in the network, often called the **parameters** θ of the network, in order to minimize the *distance* between the estimated distribution $f_\theta(x)$ and the true distribution y.

But how do we define these distance metrics? It turns out we can use **Risk function** to evaluate how well an estimator performs.

Loss Functions & Risk Functions

Definition 4 (Loss Function). Loss function $\mathcal{L}(x, y, \theta)$ measures the "badness" of an estimator, and is often measured in terms of some distance between the estimate and true estimator.

For example, the zero-one loss is $\sum_{i=1}^{n} \delta(f_\theta(x_i) \neq y_i)$ where we add one if the estimate is off, and add zero is the estimate is correct.

Derivative of Sigmoid

Sigmoid function is a popular activation function in neural networks (we will learn more about what this means in due course). Let us denote the sigmoid function as

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

(a) Calculate the partial derivative of the sigmoid function with respect to x in terms of $\sigma(x)$.

(b)

Derivative of Softmax (Challenge)

Recall the softmax function, defined by

$$p_i = \frac{e^{f_i(x)}}{\sum_{j=1}^{n} e^{f_j(x)}}$$

Softmax can be thought of as a multi-class extension to sigmoid function, and its derivative is often used for optimization. Calculate the partial derivative of the softmax function with respect to $f_k(x)$ for each k.

Definition 5 (Risk Function). The risk function is the expected loss (known as the risk), measured as a function of the parameter θ, so

$$R(\theta; f(\cdot)) = \mathbb{E}_{x\sim p(x), y\sim p(y|x)}[\mathcal{L}(x, y, \theta)]$$

For example, if $\mathcal{L}(x, y, \theta) = (y - f_\theta(x))^2$ (squared error loss), then $R(\theta; f(\cdot)) = \mathbb{E}_{x\sim p(x), y\sim p(y|x)}[(y - f_\theta(x))^2]$, also known as the **mean squared error** (or **MSE**). This is the expected squared deviation of the estimator from the true distribution (over the true distribution).

That said, we cannot directly optimize this objective (i.e., minimize the risk), since we do not have access to the true distribution, so we cannot sample $x \sim p(x)$ and we only have the dataset $\mathcal{D}$. Instead, we use **empirical risk minimization** where we replace the true distribution by the **empirical distribution** from $\mathcal{D}$.

Definition 6 (Empirical Risk). The empirical risk is the risk evaluated on **samples** from the true distribution, and approximates the true risk. It is given by:

$$\frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(x_i, y_i, \theta)$$

Supervised learning is (usually) empirical risk minimization, and we must ask: is this the same as true risk minimization? To answer this question, we will analyze the bias-variance tradeoff in next week's discussion section.

6. Summary

- We discuss two main types of ML problems: supervised and unsupervised learning.
- Solving ML problems requires us to pick a model class, loss function and an optimizer.
- Recall the Bayes Theorem,
$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)}$$
- Recall that an estimator are rules to calculate an estimate of some function of the observed data, and will often be denoted by $f_\theta(X)$ where X is the data and θ are parameters
- Loss functions measure the "badness" of an estimator, and the risk is the expected loss.
- Bias-Variance tradeoff states that errors can be decomposed into variance of the estimator plus the square of its bias (and irreducible error).

- Bias and Variance are intimately related to overfitting and underfitting problems. Overfitting occurs when the model fits the data too well and is often the result of an overly complicated model. Underfitting occurs when the model fails to capture the trend, and does not fit the data well enough.
- Divide your data into training, validation and test sets. Use training set to train your model, validation to tune your hyperparameters and test set to calculate the final accuracy.

