CS 182/282A Designing, Visualizing and Understanding Deep Neural Networks
Spring 2022 Marvin Zhang

Discussion 1

This discussion worksheet/note contains information from future lectures.

Welcome to CS 182/282A - 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 Piazza. 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.

Problem 0: Class Logistics

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

1. What times and where will lectures happen?
2. When and where are the midterms?
3. How much slip days will you be given?
4. Can you use slip days for the final project for CS282A?

Solution 0: Class Logistics

1. They will happen every Monday/Wednesday from 5:00PM to 6:30PM PST at Dwinelle 155. The first two weeks of lecture will be be online via Zoom.
2. Midterm 1 is on Wednesday, March 2 from 7-9 p.m in Pimentel 1. The second midterm, for 182 students only, will be scheduled for the last week of instruction, with more details forthcoming.
3. There will be 5 slip days, counted in day increments (i.e., if you submit your homework at 12:01AM after the deadline, that will count as one slip day)
4. No.

2 Machine Learning Overview

2.1 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 $\mathcal{D}$.

2.2 Solving Machine Learning Problems

To solve a machine learning problem, you must first define three "parameters".

1. Pick a model class (for example, do you want to use logistic regression or do you want to use a deep neural network?)
2. Pick a loss function (how do you want to determine the "badness" of your model performance?)
3. 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.

2.3 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:

- Only train your model on the training set, but not the validation set and test set.
- You should never tune your hyper-parameters on your test set or choose the best model based on the performance on the test set.
- The test set should only be run once after you have finalized your model, regardless of whether you use cross-validation or a single training-validation split. You should hold out your test set until you have finalized your model.
- You should use a new test set when you train a new model.

Problem 1: Validation Potpourri

1. Why should you never tune your hyperparameters on your test set?
2. What should your validation set be used for?
3. Describe a general ML workflow with datasets

Solution 1: Validation Potpourri

1. In tuning your hyperparameters to the test set, you are 'over-fitting' your model manually to the test set, and your test set error will no longer be an unbiased estimate of the true error on the dataset. You should only tune your hyper-parameters according to the validation set performance to preserve the validity of the test set.
2. Validation sets should be used to tune your hyperparameters.
3. First, split your training set into train/validation sets. Then, train your model using your training set. After training, tune your hyperparameter using the validation set. Finally, test your model using the test set (and do not tune hyperparameters).

3 Statistics Review

3.1 Probability Review

Definition 1 (Dataset)**.** *A dataset $\mathcal{D}$ of size n is composed of n individual examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$ represents the ith input feature and y_i represents the ith label. Datasets without the label y_i are called unlabeled datasets, and datasets with these labels are called labeled datasets.*

In general, each example could represent any data type: scalar values, images, text, audio waves and more.

Definition 2 (Joint Distribution)**.** *The joint distribution of two random variables A and B is the probability of both events co-occurring, and is written as $\mathbb{P}(A, B)$.*

Suppose we would like to model the probability distribution of our data. This will be a model of the *joint distribution* of our data, which is given by

$$\mathbb{P}(x_1, \ldots, x_n) \tag{1}$$

Definition 3 (Conditional Probability)**.** *The conditional probability of two random variables A and B is the probability of one occurring* given that *the other has occurred. The probability that A has occurred given that B has occurred is denoted $\mathbb{P}(A|B)$*

Definition 4 (Independence)**.** *If A and B are independent random variables, and their probabilities are $\mathbb{P}(A)$ and $\mathbb{P}(B)$, then their joint probability is $\mathbb{P}(A, B) = \mathbb{P}(A) \times \mathbb{P}(B)$. In other words, A and B are independent iff $\mathbb{P}(A) = \mathbb{P}(A|B)$.*

We often assume that datasets consist of *independent, identically distributed (i.i.d.)* samples. Notice what this does to the joint distribution of our data from Eq 1.

$$\mathbb{P}(x_1, \ldots, x_n) = \prod_{i=1}^{n} \mathbb{P}(x_i) \tag{2}$$

Finally, we have the identity

$$\mathbb{P}(A, B) = \mathbb{P}(A|B)\mathbb{P}(B) = \mathbb{P}(B|A)\mathbb{P}(A) \tag{3}$$

Dividing by $\mathbb{P}(B)$ then gives us Bayes' Theorem.

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(B|A)\mathbb{P}(A)}{\mathbb{P}(B)} \tag{4}$$

Problem 2: Do I Have a Flu?

Let $\mathbb{P}(H)$ be the probability you have a headache, and $\mathbb{P}(F)$ be the probability you have a flu. Calculate $\mathbb{P}(F)$, $\mathbb{P}(H)$, $\mathbb{P}(H|F)$. Then, calculate $\mathbb{P}(F|H)$ using Bayes' Theorem, given the following data:

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

Solution 2: Do I Have a Flu?

Since flu occurs 3 in 6 times, we have,

$$\mathbb{P}(F) = \frac{3}{6} = \frac{1}{2}$$

Likewise, a headache happened 3 in 6 times, so we have,

$$\mathbb{P}(H) = \frac{3}{6} = \frac{1}{2}$$

Then, given that we have a flu (3 times), the person experienced a headache 2 times, so we have,

$$\mathbb{P}(H|F) = \frac{2}{3}$$

Finally, we recall the Bayes' Theorem,

$$\mathbb{P}(F|H) = \frac{\mathbb{P}(H|F)\mathbb{P}(F)}{\mathbb{P}(H)}$$

Using the Bayes' Theorem, we conclude,

$$\mathbb{P}(F|H) = \frac{\frac{2}{3} \times \frac{1}{2}}{\frac{1}{2}} = \frac{2}{3}$$

3.2 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 5 (Statistic)**.** *A statistic is any function $T(X)$ of the observed data X.*

Definition 6 (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 7 (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 we select an estimator denoted $\theta = \hat{\theta}(\mathcal{D})$. Then, for a particular test input x, the bias can be formulated as $Bias(f_\theta(x)) = \mathbb{E}_{y\sim p(y|x)}[f_\theta(x) - y]$. Variance of an estimator is a measure of how much the estimator differs from the expected value of the estimator on average, and can be formulated as $Var(f_\theta(x)) = \mathbb{E}_{y\sim p(y|x)}[(f_\theta(x) - \mathbb{E}[f_\theta(x)])^2]$.*

Specifically, a **unbiased** estimator is one where $\mathbb{E}_{y\sim p(y|x)}[f_\theta(x)] = y$ (using the "hat" notation, we can equivalently write $\mathbb{E}[\hat{y}] = y$). The best estimator, thus, has low bias, and low variance. So why don't we always use an unbiased estimator? Sometimes, we might want to introduce a little bit of bias if it significantly decreases the variance. We will see more of this and ask you to derive the **Bias-Variance Tradeoff** in the future.

4 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.

4.1 Loss Functions & Risk Functions

Definition 8 (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.

Problem 3: 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

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

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

Solution 3: Derivative of Sigmoid

We first rewrite $\sigma(x)$ as $(1 + e^{-x})^{-1}$, and proceed:

$$\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}(1 + e^{-x})^{-1} \\
&= -(1 + e^{-x})^{-2}(-e^{-x}) \\
&= \frac{e^{-x}}{(1 + e^{-x})^2} \\
&= \frac{1}{1 + e^{-x}} \cdot \frac{e^{-x}}{1 + e^{-x}} \text{ break up into different pieces} \\
&= \frac{1}{1 + e^{-x}} \cdot \left(1 - \frac{1}{1 + e^{-x}}\right) \\
&= \sigma(x) \cdot (1 - \sigma(x))
\end{aligned}$$

Problem 4: 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.

Solution 4: Derivative of Softmax (Challenge)

We recall the quotient role for derivatives, which states that for $f(x) = g(x)/h(x)$, we have that

$$f'(x) = \frac{g'(x)h(x) - h'(x)g(x)}{h(x)^2}$$

In our case, we let $g = e^{f_i}$ and $h = \sum_{j=1}^{n} f_j$.
Then, we start with the case where $i = k$,

$$\begin{aligned}
\frac{\partial p_i}{\partial f_k} &= \frac{e^{f_i} \sum_{j=1}^{n} e^{f_j} - e^{f_k} e^{f_i}}{\left(\sum_{j=1}^{n} e^{f_j}\right)^2} \\
&= \frac{e^{f_i}}{\sum_{j=1}^{n} e^{f_j}} \frac{\sum_{j=1}^{n} e^{f_j} - e^{f_k}}{\sum_{j=1}^{n} e^{f_j}} \\
&= p_i(1 - p_j)
\end{aligned}$$

Likewise, we solve the $i \neq k$ case,

$$\begin{aligned}
\frac{\partial p_i}{\partial f_k} &= \frac{0 - e^{f_k} e^{f_i}}{\left(\sum_{j=1}^{n} e^{f_j}\right)^2} \\
&= -\frac{e^{f_i}}{\sum_{j=1}^{n} e^{f_j}} \frac{e^{f_k}}{\sum_{j=1}^{n} e^{f_j}} \\
&= -p_i \cdot p_j
\end{aligned}$$

Then,

$$\frac{\partial p_i}{\partial f_k} = \begin{cases} p_i(1 - p_j) \text{ if } i = j \\ -p_i p_j \text{ if } i \neq j \end{cases}$$

Definition 9 (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 10 (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.

5 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.
- 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.