CS 182/282A: Designing, Visualizing and Understanding Deep Neural Networks **Spring 2020**

Section 1: Intro to ML, Stats Review, Bias-Variance

Notes by: CS 182 Course Staff

1.1 Course Logistics

- 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

1.1.1 Discussion Schedules

- Tuesday 12 PM - 1 PM 246 Dwinelle David
- Tuesday 2 PM - 3 PM 179 Dwinelle Phillipe
- Tuesday 4 PM - 5 PM 202 Wheeler Phillipe
- Wednesday 9 AM - 10 AM Dwinelle Aravind
- Wednesday 2 PM - 3 PM Barrows Roshan
- Wednesday 3 PM - 4 PM Barrows Roshan
- Wednesday 4 PM - 5 PM Wheeler David
- Friday 10 AM - 11 AM Moffitt Aravind
- Friday 11 AM - 12 PM Wheeler Haozhi
- Friday 1 PM - 2 PM Etcheverry Haozhi

If anyone wishes to attend discussion sessions but cannot make it in any of the sessions, please let us know asap in class/on piazza/through email. Also, if you have requests on topics we should include in any future discussions, please let us know.

1.1.2 Groups

Group projects shall be carried out in groups of 3-4 only. All group member needs to be registered in the same course code (i.e., All CS182 or CS282A) since CS282A has additional requirements for the project.

1.2 Statistics Review

1.2.1 Data

In this section, we will define datasets and probability distributions over a dataset.

Suppose we have a dataset X of size n. This dataset is composed of individual examples $\{x_1, \ldots, x_n\}$, where each example x_i is a vector. In general, these examples could represent any data type: scalar values, images, text, audio waves, etc.

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

$$P(x_1, \ldots, x_n) \tag{1.1}$$

1.2.2 Joint and Conditional Distributions

The joint distribution of two random variables A and B is the probability of both events co-occurring, and is written as $P(A, B)$.

The conditional probability of two random variables A and B is the probability of one occurring *given that* the other has occurred. For example the conditional probability of A given B is written $P(A|B)$.

If A and B are independent random variables, and their probabilities are $P(A)$ and $P(B)$, their joint probability $P(A, B) = P(A) \times P(B)$.

To test for independence, A and B are independent iff $P(A) = P(A|B)$.

We often assume that datasets consist of *independent, identically distributed (iid.)* samples. Notice what this does to the joint distribution of our data from Eq 1.1.

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i) \tag{1.2}$$

Finally, we have identity

$$P(A, B) = P(A|B)P(B) = P(B|A)P(A) \tag{1.3}$$

Dividing by $P(B)$ then gives us Bayes' Theorem.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1.4}$$

1.2.3 Estimators

Often, we want to estimate some function of our data. We call these functions *estimators*. A simple example of an estimator is the sample mean: $\bar{X} = \frac{1}{N} \sum_{i=1}^{n} x_i$.

The bias of an estimator is equal to $Bias(\hat{y}) = \mathbb{E}_y[\hat{y} - y]$, that is, how much does the expected value of the estimator differ from the true distribution. An **unbiased** estimator is one where $\mathbb{E}_y[\hat{y}] = y$. The variance of

a estimator is how much the estimator varies, that is $Var(\hat{y}) = \mathbb{E}_y[(\hat{y} - \mathbb{E}[\hat{y}])^2]$, or how much the estimator differs from the expected value of the estimator on average.

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.

1.2.4 Prior, Posterior and Likelihood

In this section, we will review prior, posterior and likelihood through an example: estimating the probability of you having flu given you are having a headache (which is normally the goal in a lot of inference problems), using the following notations:

- $P(H)$ - Probability of you having headache
- $P(F)$ - Probability of you having flu.
- $P(F|H)$ - Probability of you having flu given that you have headache.
- $P(H|F)$ - Probability of you having headache given you have a flu

With the Bayes Theorem:

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

$$\text{Posterior} \propto \text{Likelihood} \times \text{Prior}$$

F is the parameter since it characterizes the distribution of flu contractors, which we care about.

Prior $P(F)$ is the probability of your parameters in your distribution of interest.

Likelihood $P(H|F)$ is the probability of your observation given the parameters.

Posterior $P(F|H)$ is the probability of your parameters given the observations.

Evidance $P(H)$ is the probability of your variables in your observations.

Now we will work through a simple dataset to estimate $P(H|F)$ and $P(F)$ by the distributions of the dataset. (note to TA: work through all probabilities to complete a Naive Bayes classification)

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

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

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

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

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

Note that sometimes when optimizing the parameters using EM algorithm (e.g., when some labels are missing), or when comparing various values of the parameters to output a decision, $P(H)$ is dropped from the computation since it does not change across various choices of parameter values.

1.2.5 Probability Density Function

Probability density function $f(x)$ is the relative likelihood where a random variable takes on certain values within a range, such that:

$$P(a < x < b) = \int_a^b f(x)dx$$

See figure 1.1 for an example.

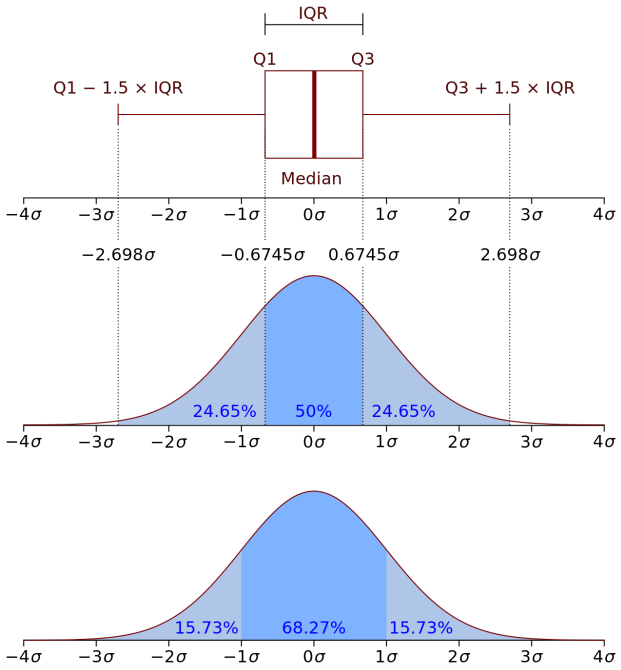

Figure 1.1: A diagram of a pdf for normal distribution. Figure from Wikipedia.

Note that Probability Mass function has similar defini

1.3 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. In the case of classification, we are trying to learn $p(y|x)$, the probability of some class y given

the 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 $\hat{y}$ of a true distribution y.

The way that we find this estimator $\hat{y}$ is by adjusting the weights and biases in the network, often called the **parameters** of the network in order to minimize the distance between the estimated distribution $\hat{y}$ and the true distribution y. But how do we specify this distance?

One of the most common distance metrics, or **Risk functions** for evaluating the quality of an estimator is the mean-squared error:

$$MSE(\hat{y}) = \mathbb{E}_y[(y - \hat{y})^2] \tag{1.5}$$

This is the expected squared deviation of the estimator from the true distribution (over the true distribution).

Because we don't have access to the true distribution y, we cannot directly optimize this objective, however, we minimize the **empirical risk**, where the true distribution y is replaced by the **empirical distribution** (Samples from the true distribution y). We will see echos of the MSE risk function throughout the material presented in the class.

1.3.1 Bias-Variance Tradeoff

Even though we can't directly optimize the MSE, we can still look at the MSE equation to better understand sources of error in our estimation. In particular, we can derive the **bias-variance** decomposition of the MSE, and relate it to **over-fitting** and **under-fitting** in the network.

A simple of the bias-variance decomposition is given below:

$$\begin{aligned} MSE(\hat{y}) &= \mathbb{E}_y[(\hat{y} - y)^2] && (1.6)\\ &= \mathbb{E}_y[\hat{y}^2 - 2y\hat{y} + y^2] && (1.7)\\ &= \mathbb{E}_y[\hat{y}^2] - 2y\mathbb{E}_y[\hat{y}] + \mathbb{E}_y[y^2] && (1.8)\\ &= Var(\hat{y}) + \mathbb{E}_y[\hat{y}]^2 - 2y\mathbb{E}_y[\hat{y}] + y^2 && (1.9)\\ &= Var(\hat{y}) + (\mathbb{E}_y[\hat{y}] - y)^2 && (1.10)\\ &= Var(\hat{y}) + Bias(\hat{y})^2 && (1.11) \end{aligned}$$

We can thus see that the MSE estimator is exactly to the variance of the estimator plus the square of it's bias. But what this mean?

Bias and variance can be summarized by the following graphic:

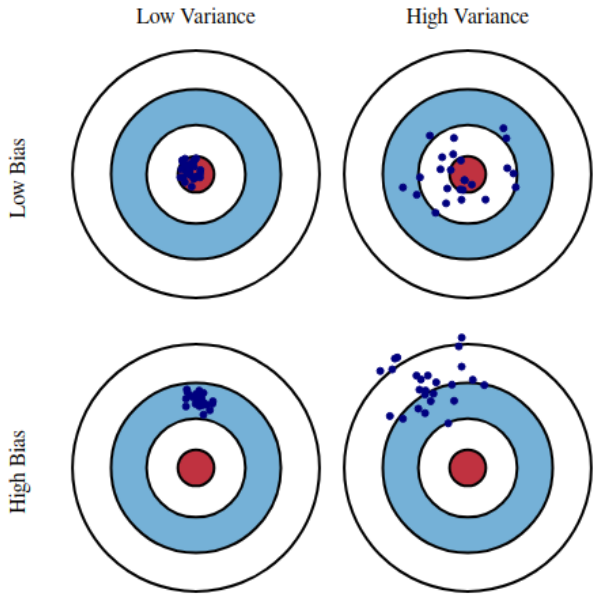

Figure 1.2: A visual explanation of the bias and variance. Figure from [1].

Exercise: Is the Nearest Neighbor estimator unbiased? How about a k-Nearest Neighbor estimator?

1.3.2 Bias and Variance in Neural Networks

There are two terms that often come up when we discuss bias and variance in neural networks. The first term is **Overfitting**. Overfitting occurs when a statistical model or machine learning algorithm captures the noise of the data. Intuitively, overfitting occurs when the model or the algorithm fits the data too well. Specifically, overfitting occurs if the model or algorithm shows low bias but high variance. Overfitting is often a result of an excessively complicated model, and it can be prevented by fitting multiple models and using validation or cross-validation to compare their predictive accuracies on test data [2].

The other term that often comes up is **underfitting**. Underfitting occurs when a statistical model or machine learning algorithm cannot capture the underlying trend of the data. Intuitively, underfitting occurs when the model or the algorithm does not fit the data well enough. Specifically, underfitting occurs if the model or algorithm shows low variance but high bias [2].

1.4 Deriving Logistic Regression

In class, we discussed the basics of generative and discriminative classifiers, as well as showed an introduction to logistic regression. Recall that Logistic regression is a method for predicting a binary target value $y \in \{0, 1\}$ from an m-dimensional binary input vector $x \in \{0, 1\}^m$. In class, we derived the function

$$f(x) = \frac{1}{1 + exp(-w^T x + b)} \tag{1.12}$$

Where does this function come from? Why do we pick the logistic function from all of the possible nonlinear functions? Our goal is to show that this model can be derived from the perspective of a Naive Bayes model with class-conditional Gaussians.

The Naive Bayes model is exemplified by a generative process for the data:

1. We choose a class $Y = 0$ or $Y = 1$ depending on a Bernoulli random process
2. We sample the class features $\{X_1, \ldots, X_m\}$ from sampling m unique gaussian distributions which have different parameters depending on the value of Y.

If we follow this process and we make some conditional independence assumptions, we find that we get a joint probability for any sample (x, y) in our dataset is:

$$P(x, y|\theta) = p(y|\pi) \prod_{j=1}^{m} p(x_j|y, \theta_j) \tag{1.13}$$

Where θ is a set of parameters. More formally, let $Y \in 0, 1$ be a Bernoulli random variable with parameter π. Thus for a dataset element in our distribution:

$$p(y|\pi) = \pi^y (1 - \pi)^{1-y} \tag{1.14}$$

Given the conditional independence assumption expressed by the Naive Bayes model, the probability $p(x|y)$ will factor into a product over conditional probabilities $p(x_j|y)$.

For $Y = 0$, let each X_j have a Gaussian distribution:

$$P(x_j|Y = 0, \theta_j) = \frac{1}{(2\pi\sigma_j^2)^{1/2}} exp\left\{-\frac{1}{2\sigma_j^2}(x_j - \mu_{0j})^2\right\} \tag{1.15}$$

where μ_{0j} is the j'th component of the mean vector for the class $Y = 0$. For the class $Y = 1$, we have similarly:

$$P(x_j|Y = 1, \theta_j) = \frac{1}{(2\pi\sigma_j^2)^{1/2}} exp\left\{-\frac{1}{2\sigma_j^2}(x_j - \mu_{1j})^2\right\} \tag{1.16}$$

Notice that we assume that the variances remain the same between both distributions - while they can vary along the component distribution, they are not allowed to vary between classes. Thus, we have $\theta_j = (\mu_{0j}, \mu_{1j}, \sigma_j^2)$.

Let's now then calculate the posterior probability $P(Y = 1|x, \theta)$ for $x = (x_1, \ldots, x_m)$. The math is a bit ugly, but greatly simplified if we work with matrix notation. Thus let:

$$p(x|y = k, \theta) = \frac{1}{(2\pi)^{m/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1}(x - \mu_k)\right\} \tag{1.17}$$

for both of the classes $k \in \{0, 1\}$. We have $\mu_k = (\mu_{k1}, \mu_{k2}, \ldots, \mu_{km})$ as the vector of means for the k'th Gaussian, and where $\Sigma = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_m^2)$ is the diagonal covariance matrix. We thus have:

$$p(Y=1|x,\theta) = \frac{p(x|y=1,\theta)p(Y=1|\pi)}{p(x|Y=1,\theta)p(Y=1|\pi)+p(x|Y=0,\theta)p(Y=0|\pi)} \tag{1.18}$$

$$= \frac{\pi \exp\{-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\}}{\pi \exp\{-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\}+(1-\pi)\exp\{-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\}} \tag{1.19}$$

$$= \frac{1}{1+\exp\{-\log\frac{\pi}{1-\pi}+\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\}} \tag{1.20}$$

$$= \frac{1}{1+\exp\{-(\mu_1-\mu_0)^T\Sigma^{-1}x+\frac{1}{2}(\mu_1-\mu_0)^T\Sigma^{-1}(\mu_1+\mu_0)-\log\frac{\pi}{1-\pi}\}} \tag{1.21}$$

$$= \frac{1}{1+\exp\{-\beta^T x+\gamma\}} \tag{1.22}$$

Where the final equation defines the parameters β and γ:

$$\beta \equiv \Sigma^{-1}(\mu_1-\mu_0) \tag{1.23}$$

and

$$\gamma \equiv \frac{1}{2}(\mu_1-\mu_0)^T\Sigma^{-1}(\mu_1+\mu_0)-\log\frac{\pi}{1-\pi} \tag{1.24}$$

We can recall that the formula presented in class uses the same parameters γ and β, however we make no probabilistic assumptions on the two parameters. Thus, we can see that the logistic function in logistic regression arises from some simple probabilistic assumptions given by the Naive Bayes classifier.

** This derivation is courtesy Dr. Michael I. Jordan (From our very own UC Berkeley, Not the Basketball player)

1.5 Side-note on Dataset splits during training

In the case when hyper-parameter tuning is possible (e.g., λ in SVMs, learning rate of deep nets), in addition to training and test sets, you should hold out validation set for tuning hyper-parameters in your model. 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, and choose the best model based on the performance on the test set, because in doing so, you are 'over-fitting' your model manually to the test set. You should only tune your hyper-parameters according to the validation set performance.
- 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.

References

[1] *Bias and Variance*. URL: http://scott.fortmann-roe.com/docs/BiasVariance.html.

[2] *Machine Learning Lesson of the Day - Overfitting and Underfitting*. Mar. 2014. URL: https://chemicalstatistician.wordpress.com/2014/03/19/machine-learning-lesson-of-the-day-overfitting-and-underfitting/.