Chapter 1

Machine Learning Basics

Learning Algorithms

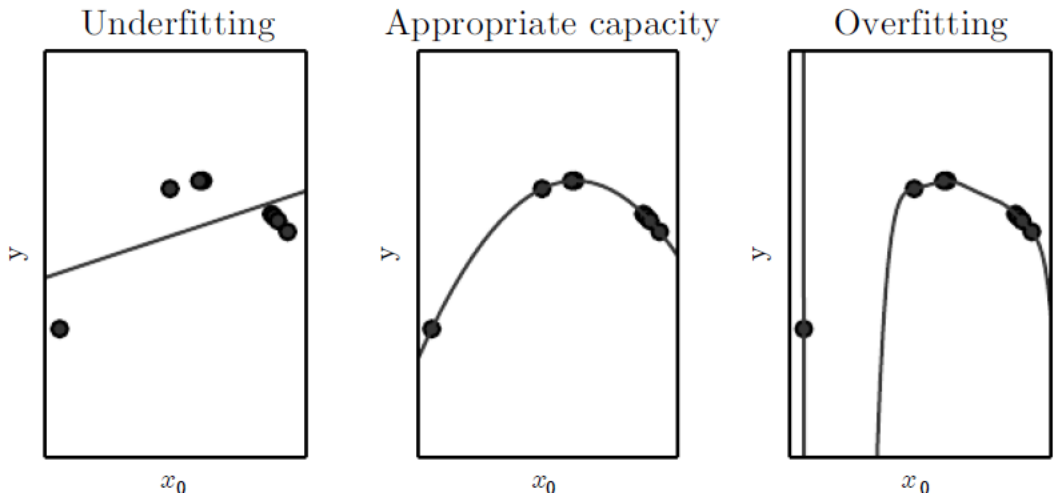
- A machine learning algorithm is an algorithm that is able to learn from data. A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.
- Common ML tasks are Classification, Regression, Clustering, Machine translation, Anomaly detection etc.
- To evaluate a ML algorithm, need a quantitative measure of its performance. Usually this performance measure P is specific to the task T being carried out by the system. For classification, we often use the accuracy of the model and for regression we commonly use RMSE. Performance measured must be measured on a test set of data that the model has not been trained in order to tell how well the model generalizes.
- Most learning algorithms experience an entire dataset– a collection of many examples. Datasets can be commonly described with a design matrix X, where rows are examples and columns are features.

Capacity, Overfitting and Underfitting

- The central challenge in machine learning is that we must perform well on new, previously unseen inputs—not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called generalization.
- Typically, when training a machine learning model, we have access to a training set, we can compute some error measure on the training set called the training error, and we reduce this training error. So far, what we have described is simply an optimization problem. What separates machine learning from optimization is that we want the generalization error, also called the test error, to be low as well.
- How can we affect performance on the test set when we get to observe only the training set? The field of statistical learning theory provides some answers.
- We typically make a set of assumptions known collectively as the i.i.d. assumptions while generating training and test sets. Under this assumption, expected training error of a randomly selected model is equal to the expected test error of that model.
- The factors determining how well a machine learning algorithm will perform are its ability to:
    - ✓ Make the training error small.

- ✓ Make the gap between training and test error small.

- These two factors correspond to the two central challenges in machine learning: Underfitting and Overfitting. Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set. Overfitting occurs when the gap between the training error and test error is too large.
- We can control whether a model is more likely to overfit or underfit by altering its capacity. Informally, a model's capacity is its ability to fit a wide variety of functions. Models with low capacity may struggle to fit the training set. Models with high capacity can overfit by memorizing properties of the training set that do not serve them well on the test set.
- Machine learning algorithms will generally perform best when their capacity is appropriate for the true complexity of the task they need to perform and the amount of training data they are provided with. Models with insufficient capacity are unable to solve complex tasks. Models with high capacity can solve complex tasks, but when their capacity is higher than needed to solve the present task they may overfit.



- ML algorithms generally perform best when their capacity is appropriate for the true complexity of the task they need to perform and the amount of training data provided.
- Capacity is not determined only by the choice of model. The model specifies which family of functions the learning algorithm can choose from when varying the parameters in order to reduce a training objective. This is called the representational capacity of the model.
- Learning algorithm's effective capacity may be less than the representational capacity of the model family.

Hyperparameters and Validation Sets

- Most machine learning algorithms have several settings that we can use to control the behavior of the learning algorithm. These settings are called hyperparameters. The values of hyperparameters are not adapted by the learning algorithm itself.
- Though we can design a nested learning procedure where one learning algorithm learns the best hyperparameters for another learning algorithm.
- In the polynomial regression, there is a single hyperparameter: the degree of the polynomial, which acts as a capacity hyperparameter. The λ value used to control the strength of weight decay in regularization is another example of a hyperparameter.
- Hyperparameters may be chosen and not learned because they are difficult to optimize more often. It is not appropriate to learn the hyperparameter on the training set. This applies to all hyperparameters controlling model capacity. If learned on the training set, the hyperparameter would always be set to the maximum capacity, resulting in Overfitting.
- For example, we can always fit the training set better with a higher degree polynomial and a weight decay setting of $\lambda = 0$ than we could with a lower degree polynomial and a positive weight decay setting.
- To solve this problem, we need a validation set of examples that the training algorithm does not observe.
- The validation set needs to be distinct from the test set, since no aspects of the model, including hyperparameters, should be determined based on the test set.
- We always construct the validation set from the training data. Specifically, we split the training data into two disjoint subsets. One of these subsets is used to learn the parameters. The other subset is our validation set, used to estimate the generalization error during or after training, allowing for the hyperparameters to be updated accordingly.
- Typically, one uses about 80% of the training data for training and 20% for validation.

Estimators, Bias and Variance

Statistical concepts such as parameter estimation, bias, and variance are useful to formally characterize the machine learning concepts of generalization, Underfitting, and Overfitting.

Point estimation is the attempt to provide the single "best" prediction of some quantity of interest. For example, a single parameter, a vector of parameters, or even a whole

function. Let $\hat{\theta}$ is an estimate of the true value θ. Let $x^{(1)}, \ldots, x^{(m)}$ be a set of m i.i.d data points. A point estimator or statistic is any function of the data.

$$\hat{\theta}_m = g\left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\right)$$

Point estimation can also refer to the estimation of the relationship between input and target variables, i.e. function estimation. Function estimation is attempt of trying to predict a variable y given an input vector x. Assume there is a function f(x) describing the approximate relationship between y and x.

y = f(x) + ϵ, where ϵ represents the part of y not predictable from x

In function estimation, we are interested in approximating f with a model or estimate $\hat{f}$. Function estimation is really just the same as estimating a parameter θ. The linear regression example and the polynomial regression example are both examples of scenarios that may be interpreted either as estimating a parameter w or estimating a function $\hat{f}$ mapping from x to y.

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Let x be a point, $f(x)$ is true value and $\hat{f}(x)$ is value predicted by model. Then

$$Bias = E[\hat{f}(x)] - f(x)$$

Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

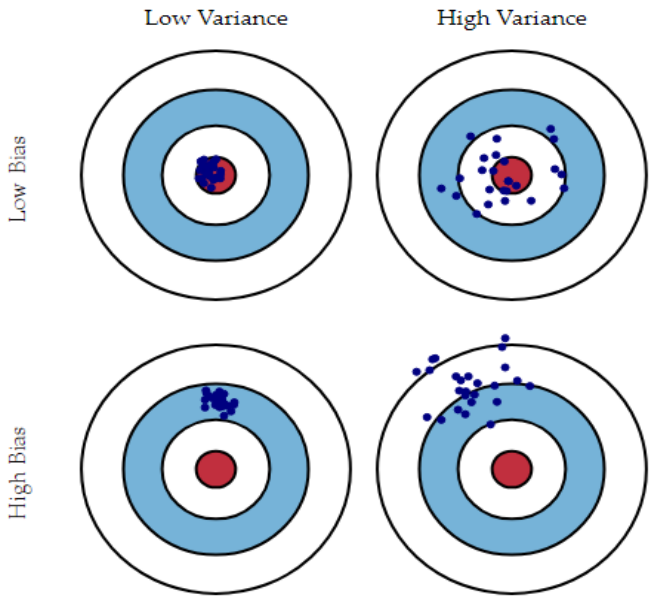
Bias is the tendency of an estimator to pick a model for the data that is not structurally correct. For example, suppose that we use a linear regression model on a cubic function.

Variance of an estimator is the "expected" value of the squared difference between the estimate of a model and the "expected" value of the estimate.

$$Variance = E(\hat{f}(x) - E[\hat{f}(x)])^2$$

Variance is taken as the variability of a model prediction for a given data point. It is the result of Overfitting. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.

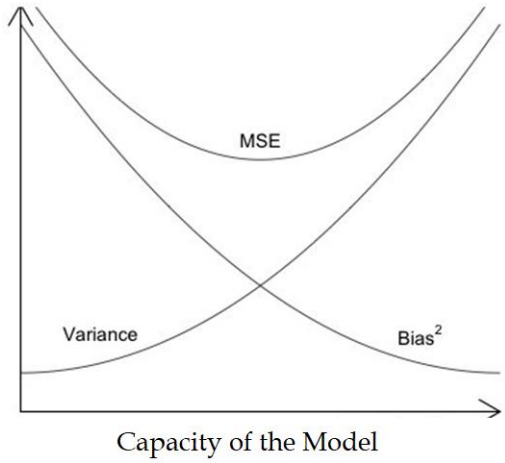
Variance is error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs.



Bias and variance measure two different sources of error in an estimator. Therefore, balancing between these two is necessary while choosing estimators. MSE of estimators is given as below:

$$MSE = Bias^2 + Variance + Irreducible\ Error$$

Trading off balance between bias and variance keeps MSE small as demonstrated in the figure below.



Maximum Likelihood Estimation (MLE)

MLE is a technique used for estimating the parameters of a given distribution using some observed data.

Suppose we have random sample $x_1, x_2, \ldots, x_n$ whose assumed probability distribution depends on some unknown parameter θ and our primary goal is to find point estimator f such hat $f(x_1, x_2, \ldots, x_n)$ is good estimate of θ, where, $x_1, x_2, \ldots, x_n$ are observed values of the random sample.

Prepared By: Arjun Singh Saud

For example, if a population is known to follow a normal distribution but the mean and variance are unknown.
Suppose the weight of randomly selected Female College students are normally distributed with unknown μ mean and σ standard deviation .A random sample of 10 female students yielded the following weights (in pound).

115 122 130 127 149 160 152 138 149 180

Using this, identify maximum likelihood function and maximum likelihood estimator of μ and σ.

It seems reasonable that a good estimate of the unknown parameter θ would be the value of θ that maximizes the probability, that is, the likelihood of getting the data we observed.
The goal of MLE is to maximize the likelihood function:

$$L = f(x_1, x_2, \ldots., x_n|\theta) = \prod_{i=1}^{n} f(x_i|\theta)$$

For maximization, we have

$$\frac{dL}{d\theta} = 0 \qquad \text{and} \qquad \frac{d^2L}{d\theta^2} < 0$$

Since logarithm is a non-decreasing function, so for maximizing L, it is equivalently correct to maximize log L.

⇨ $\frac{dlogL}{d\theta} = 0$

$$logL = \sum_{i=1}^{n} \log(f(x_i|\theta))$$

The simplest case is when both the distribution and parameter space are discrete, meaning that there are finite number of probabilities for each, MLE can be determined by explicitly trying all possibilities.

MLE for Uniform Distribution

Suppose the data $x_1, x_2, \ldots, x_n$ is drawn from a N(μ, σ2) distribution, where μ and σ are unknown. Find the maximum likelihood estimate for the pair (μ, σ2).
In case of uniform distribution, PDF for each x_i is given as below:

$$f(x_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(\frac{(x_i-\mu)^2}{2\sigma^2})}$$

Since the x_i are independent their joint pdf is the product of the individual pdf's:

$$f(x_{1\ldots\ldots}x_n|\mu, \sigma) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n e^{-\sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}}$$

The log likelihood is

$$\log(f(x_{1.....}x_n|\mu,\sigma)) = \log\left(\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n e^{-\sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}}\right)$$

$$= -n\log(\sqrt{2\pi}) - nlog(\sigma) - \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{2\sigma^2}$$

Since $\log(f(x_{1.....}x_n|\mu,\sigma))$ is a function of the two variables μ, σ we use partial derivatives to find the MLE.

$$\frac{\partial(f(x_{1.....}x_n|\mu,\sigma)}{\partial\mu} = \sum_{i=1}^{n}\frac{(x_i-\mu)}{\sigma^2}$$

$$\Rightarrow \quad \sum_{i=1}^{n}(x_i-\mu) = 0$$

$$\Rightarrow \quad \mu = \frac{\sum_{i=1}^{n}x_i}{n} = \bar{x}$$

To find σ, we differentiate and solve for σ:

$$\frac{\partial(f(x_{1.....}x_n|\mu,\sigma)}{\partial\sigma} = -\frac{n}{\sigma} + \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{\sigma^3}$$

$$\Rightarrow \quad -\frac{n}{\sigma} + \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{\sigma^3} = 0$$

$$\Rightarrow \quad \sigma^2 = \sum_{i=1}^{n}\frac{(x_i-\mu)^2}{n}$$

$$\Rightarrow \quad \sigma^2 = \sum_{i=1}^{n}\frac{(x_i-\bar{x})^2}{n}$$

Bayesian Statistics

Frequentist statistics is a type of statistical inference that draws conclusions from sample data by emphasizing the frequency or proportion of the data. On the other hand, Bayesian statistics is a statistical approach that utilizes Bayes' theorem for data analysis and parameter estimation. What sets Bayesian statistics apart is that all observed and unobserved parameters in a statistical model are assigned a joint probability distribution, known as the prior and data distributions. Bayes rule and joint probabilities used for Bayesian inference are given as below.

Let x=<x_1,x_2,……..,x_n> is data sample data and θ is the parameter to be estimated, Bayes rule for estimating θ is given as below.

$$P(\theta|<x_1,x_2,\ldots\ldots,x_n>) = \frac{P(<x_1,x_2,\ldots\ldots,x_n>|\theta)P(\theta)}{P(<x_1,x_2,\ldots\ldots,x_n>)}$$

$$P(<x_1,x_2,\ldots\ldots,x_n>) = \prod_{i=1}^{n} P(x_i)$$

Suppose, out of all the 4 championship races between Ram and Hari, Ram won 3 times while Hari managed only 1. So, if you were to bet on the winner of the next race, definitely you will bet on Ram. Here is the twist. What if you are told that it rained once when Hari won and once when Ram won, and it is definite that it will rain on the next date? So, who would you bet your money on now? By intuition, it is easy to see that the chances of winning for Hari have increased drastically. This type of inference is possible only with Bayesian Statistics.

Building an ML Algorithm

- Nearly all deep learning algorithms combine specification of a dataset, a cost function, an optimization procedure, and a model (E.g. linear regression).
- The cost function typically includes at least one term that causes the learning process to perform statistical estimation. The most common cost function is the negative log-likelihood (minimizing NLL = maximum likelihood estimation).
- The cost function may also include additional terms, such as regularization terms like weight decay (aka L2, or ridge regression).
- If the model is nonlinear, then the cost function usually cannot be optimized in closed form. This requires an iterative numerical optimization procedure, e.g. gradient descent.
- In some cases, the cost function may be a function that we cannot actually evaluate, for computational reasons. In these cases, we can still approximately minimize the function using iterative numerical optimization if we have some way of approximating its gradients.