African Master's in Machine Intelligence

Course Tittle : Foundations of Machine Learning

Lecturer : Moustapha Cisse

Summarized By : Blessing Itoro Bassey

Course Structure

1. **Course Structure:**
   - **Project**
   - **Exam**
   - **Presentations**
   - **Mini Test**
2. **Recommended Materials/Tools**
   - **Pattern Recognition and Machine Learning**
   - **Understanding Machine Learning from Theory to Algorithm**
   - **Deep Learning Book, an MIT Press book**

December 6, 2019

Contents

1 Lecture 1

1.1 Introduction to Machine Learning

It's exciting living in the 21st Century! The world is constantly changing and we are exposed to new ideas each day. Professor Michael Osborne from the University of Oxford recent work addresses the societal consequences of machine learning and robotics. Osborne has analyzed how intelligent algorithms might soon substitute for human workers, and has predicted the resulting impact on employment. what then is Machine learning? several people had different definition for the term *Machine Learning* e.g Arthur Samuel (1959) described Machine learning as a field of study that gives the computer the ability to learn without being explicitly programmed. however, off all the difference destinations seen, the one i love most is that of Tom Mitchell (1998), his approach was a well posed problem, he said that "A machine is said to *learn* from experience $\boldsymbol{E}$ with respect to some given task $\boldsymbol{T}$ and some performance measure on $\boldsymbol{P}$, if its performance on $\boldsymbol{T}$, as measured by $\boldsymbol{P}$, improves with experience $\boldsymbol{E}$." what does each of these term mean, let's take a typical example. suppose my email program watches me as I mark or separate my in coming emails as spam and not spam, and based on that, it learns how to better filter any in coming emails as spam and not spam. now, what is ***E, T, and P***?

- The email program watching me label or separate the emails as spam or not spam is the *Experience* ($\boldsymbol{E}$)
- The email program labeling or separating the emails as spam or not spam is the *Task* ($\boldsymbol{T}$)
- The number of emails that the email program correctly label or classified as spam not spam is the *performance* ($\boldsymbol{P}$)

Broadly, there are 3 types of Machine Learning Algorithms

1. **Supervised Learning:** This algorithm consist of a target also known as output variable (or dependent variable) which is to be predicted from a given set of features (independent variables or input variable). Using these set of variables, we generate a function that map inputs to desired outputs. The training process continues until the model achieves a desired level of accuracy on the training data. Examples of Supervised Learning: Regression, Decision Tree, Random Forest, KNN, Logistic Regression etc.
2. **Unsupervised Learning:** In this algorithm, we do not have any target or outcome variable to predict or estimate. It is used for clustering population in different groups based on their similarities, this is widely used for segmenting customers in different groups for specific intervention. Examples of Unsupervised Learning: K-means.
3. **Reinforcement Learning**: In this algorithm, the machine is trained to make specific decisions. It works this way: the machine is exposed to an environment where it trains itself continually using trial and error. This machine learns from past experience and tries to capture the best possible knowledge to make accurate business decisions. Example of Reinforcement Learning: Markov Decision Process

1.2 Supervised Learning

As described above, in supervised learning, there is always a target as well as the features. it can be represented as follows:

$$h_0 : \mathbb{R}^d \longrightarrow \mathbb{R}^c \tag{1}$$

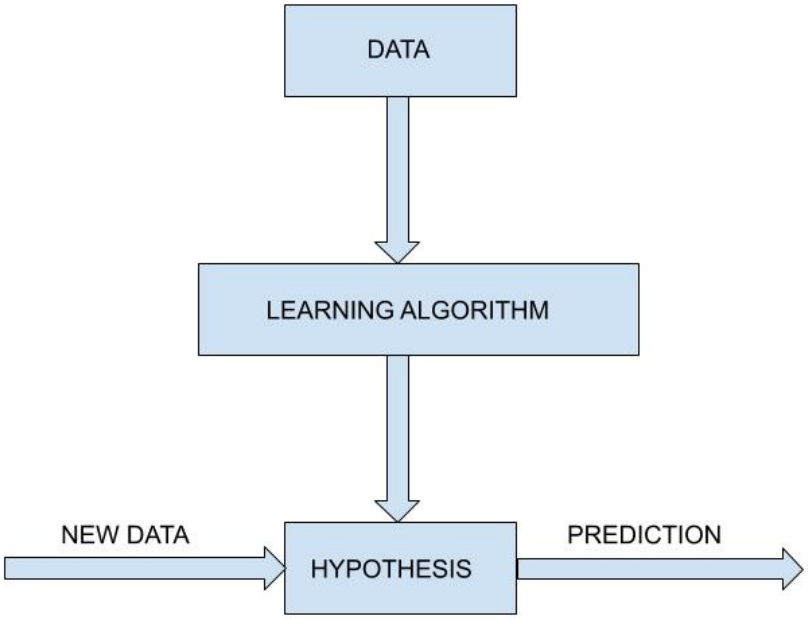
Where;
d : is the number of features or input variables and
c : is the number of target or output variable

Let's consider a typical example, given the data consisting the weight of fruits and price, how can we predict the price of another new fruit, when the weight is given? This is a regression problem, because the output variable or target variable to be predicted is a continuous variable, hence the hypothesis would be a linear model, and it is given as follows:

$$h_0(x) = \theta_0 + \theta_1 x_1 \tag{2}$$

θ_i are known as the *parameters*, however, θ_0 is called the *bias* term, and the x_1 is the input variable *weight of the fruit*. More generally, the model representation can can be seen in a flow chart bellow:

Figure1: Model Representation



For an supervised learning, when data is given, there are various steps we need to take in other to attain to our desired goal, below are the necessary steps:

- Get your data, (do some pre-proccesing if need be)
- Based on the given data, decide on the hypothesis to use.
- Since, the hypothesis was decided intuitively, there is a need to check if our model is learning well or not, hence we set a criteria to measure success, and this criteria is known as the *Loss function or cost function.*
- After the criteria has been set, we learn our algorithm by minimizing the criteria.

For a linear regression, the cost function is given as:

$$\mathbb{J}(\theta) = \frac{1}{2m}\sum_{i}^{m}\left(h_0(x_i) - y_i\right)^2 \tag{3}$$

m is the number of training examples, y_i is the number of target or output variable and $h_0(x)$ is the predictions. To minimize this cost function, we use an algorithm known as the *gradient descent.* The ***Gradient Descent*** is a supervised learning algorithm that helps find the best parameter that minimizes the cost function. Before the gradient descent algorithm can be implemented, we need to calculate the gradient, and to do this, we just differentiate the loss function and then set it to zero, after which we increment theta until it converges (when the difference between each update is very minimal). lets see how the algorithm looks like.

initializing θ:
we repeat until convergence

$$\theta_j := \theta_j - \alpha\frac{\partial d}{\partial\theta_j}\mathbb{J}(\theta) \tag{4}$$

$j = 0, 1, 2\cdots n$, where n is the number of features, and α is the learning rate.

Note that, irrespective of the initialization of theta, your solution would always converge, since linear regression is a convex optimization problem. i.e, it has just the global minimum, no local minimum.
There are three types of gradient descent algorithm, although they all help in minimizing the cost function, however, they differ in their mode of operations a little bit.

- **Batch Gradient Descent Algorithm**: With this type of algorithm, all the training examples are used for the update.

Advantages

  - It is computationally efficient, rather than a single example all training samples are used, hence making it converge at the global minimum
  - It can benefit from vectorization method, as this tend to increases the speed of processing all training samples together

Disadvantages

  - It is computationally expensive, as the entire training set can be too large to process in the memory
  - Depending on computer resources it can take too long for processing all the training samples as a batch

Differentiating equation 20, we'll have

$$\frac{\partial \mathbb{J}(\theta)}{\partial \theta} = \frac{1}{m} \sum_{i}^{m} \left(h_0(x_i) - y_i\right) * x_i \tag{5}$$

hence, equation 4 can be re-written as

initializing θ:

we repeat until convergence

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i}^{m} \left(h_0(x_i) - y_i\right) * x_i \tag{6}$$

Therefore, equation 6 is the update for the *Batch Gradient Descent.*

- **Mini Batch Gradient Descent Algorithm**: This type of algorithm uses a batch of a fixed number of training examples which is less than the actual dataset and call it a mini-batch.

Advantages

  - It is computationally efficient and does not take large memory
  - It can also benefit from vectorization method.

Disadvantages

  - It is terribly inefficient and needs to loop over the entire dataset many times to find a good solution.
  - This method is a compromise that injects enough noise to each gradient update, while achieving a relative speedy convergence.

initializing θ:

we repeat until convergence

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i}^{k} \left(h_0(x_i) - y_i\right) * x_i \qquad\qquad k < m \tag{7}$$

Therefore, equation 7 is the update for the *Mini Batch Gradient Descent.*

- **Stochastic Gradient Descent Algorithm**: In Batch Gradient Descent we were considering all the examples for every update. But what if our dataset is very huge.This does not seem an efficient way. To tackle this problem we use the Stochastic Gradient Descent.(SGD), in this algorithm, we consider just one training example at a time for each update.

Advantages

- It is easier to fit into memory due to a single training sample being processed
- SGD can be used for larger datasets. It converges faster when the dataset is large as it causes updates to the parameters more frequently.

Disadvantages

- Since it randomly picks one training example, if a bad one is picked, then it will give a variation in accuracy.
- Due to frequent updates the steps taken towards the minima are very noisy. This can often lead the gradient descent into other directions.

initializing θ:

we repeat until convergence:

for j in range of m

$$\theta_j := \theta_j - \alpha \frac{1}{m} \big(h_0(x_i) - y_i\big) * x_i \tag{8}$$

Therefore, equation 8 is the update for the *Stochastic Gradient Descent.*

Note that, for all the 3 methods above, the choice of α which is known as the learning rate, (a tuning hyper parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a loss function) is something one should be careful with, if α is too small, is will take a longer time to reach convergence, if α is too large, is tends to diverge and oscillate around without converging, hence one can just play around with with, until the desired goal is achieved.

Now that we have seen the numerical way of minimizing the cost function, it would be nice to also see the ***Analytical Approach***. This method is quite different as it does not require any learning rate or update, however, it is best if the matrix is invertible. The loss function in equation 20 can be written in a vector form as follows:

$$\mathbb{J}(\theta) = \frac{1}{2}(X\theta - Y)^T(X\theta - Y) \tag{9}$$

Just like before, we need to get the value of θ.

$$\mathbb{J}(\theta) = \frac{1}{2}(\theta^T X^T - Y^T)(X\theta - Y)$$

expanding we have

$$\mathbb{J}(\theta) = \frac{1}{2}(\theta^T X^T X\theta - \theta^T X^T Y - Y^T X\theta - Y^T Y)$$

$$\mathbb{J}(\theta) = \frac{1}{2}(\theta^T X^T X\theta - 2Y^T X\theta - Y^T Y)$$

$$\frac{\partial \mathbb{J}(\theta)}{\partial \theta} = \frac{1}{2}(2X^T X\theta - X^T Y)$$

$$0 = X^T X\theta - X^T Y$$

$$X^T X\theta = X^T Y$$

$$\implies \theta = (X^T X)^{-1} X^T Y$$

Differences between the Gradient Descent and the Normal Equation Method of Optimization

Gradient Descent	Normal Equation
• Needs to choose α	• No need of α
• many iterations are required	• No iteration is required
• Works well even when n is large	• Slow if n is very large
• Complexity is $O(n^2)$	• Complexity is $O(n^3)$

The challenge with the normal equation is that, if the matrix is not invertible, then the method might not be possible hence, there is a need for *Regularization.* which will yield an equation of the form below:

$$\mathbb{J}(\theta) = \frac{1}{2}\|X\theta - Y\|^2 + \lambda\|\theta\| \tag{10}$$

λ is the regularized parameter. Don't worry, this would be discussed in detail in subsequent lectures.

2 Lecture 2

2.1 The Probabilistic Approach for Linear Regression

Having seen the numerical and analytical way of approaching the Linear Regression problem, let's dive into the ***Probabilistic Approach***. In this method the errors are assumed to follow a Gaussian Normal Distribution. this is simple because when there is loads of data suddenly randomness or errors appears and it comes from many different sources (e.g. measurement error, data entry error, classification error, even data corruption, Still the question remains, why always Gaussian? Why not something else? The answer lies in the "Central Limit Theorem". This gives us an idea that when you take a bunch of random numbers from almost any distribution and add them together, you get something which is normally distributed. The more numbers you add, the more normally distributed it gets. The probability density of the normal distribution is given as:

$$f(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{11}$$

where: μ is the mean, and σ^2 is the standard deviation therefore, from equation 2 we have:

$$y_i = \theta^T x_i + \epsilon_i \tag{12}$$

$$\epsilon_i = y_i - \theta x_i$$

$$\epsilon_i \sim \mathcal{N}(0,\sigma) \tag{13}$$

$$p(\epsilon_i) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_i-\theta^T x_i)^2}{2\sigma^2}\right)$$

The ***Likelihood function*** is the probabilistic distribution of θ, and it is given as follows, for m training examples:

$$L(\theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta)$$

The Principle of Maximum Likelihood Estimation: states that we need to choose the parameter that helps maximizes the probability of out data. in this case, our parameter is θ, since maximizing the likelihood is the same as maximizing the log likelihood, hence, we can go ahead and maximize the log likelihood, this is just to eliminate the exponential in the function.

$$\mathcal{L}(\theta) = \log L(\theta)$$

$$= \log\prod_{i=1}^{m}\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_i-\theta^T x_i)^2}{2\sigma^2}\right)$$

$$= \sum_{i=1}^{m}\log\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y_i-\theta^T x_i)^2}{2\sigma^2}\right)$$

$$= m \log \frac{1}{\sqrt{2\pi\sigma^2}} \sum_{i=1}^{m} - \frac{(y_i - \theta^T x_i)^2}{2\sigma^2} \tag{14}$$

Maximizing θ in the above equation is the same as maximizing it without its constant. hence, equation 14 can be written as

$$\sum_{i=1}^{m} -(y_i - \theta^T x_i)^2 \tag{15}$$

Since Maximizing a function is also the same as minimizing it's negative (-ve), hence equation 15 above can be written as follows:

$$\sum_{i=1}^{m} (y_i - \theta^T x_i)^2 \tag{16}$$

2.2 Logistic Regression

A solution for classification is logistic regression. Instead of fitting a straight line as in equation 2, the logistic regression model uses the *logistic function* to squeeze the output of a linear equation between 0 and 1. The logistic function is defined as:

$$g(z) = \frac{1}{1 + \exp^{(-z)}} \tag{17}$$

The sigmoid function (g(z)) looks like this:

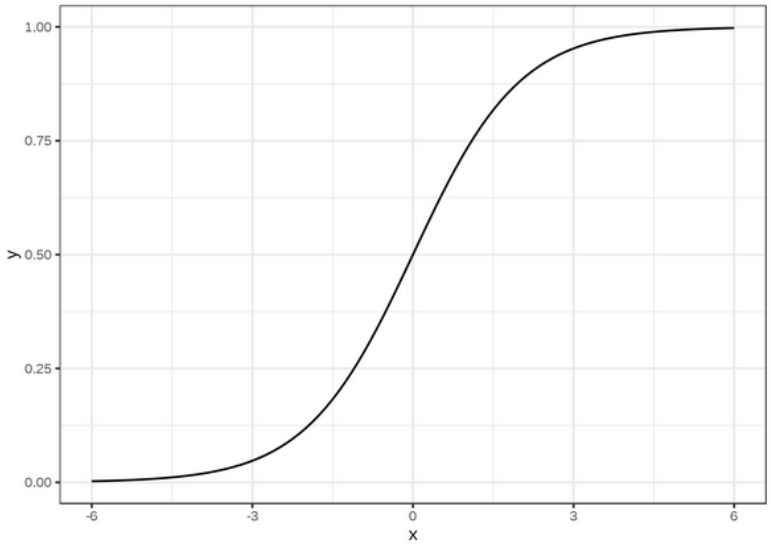


Figure 1

For classification, we prefer probabilities between 0 and 1, equation 2 is wraped into the logistic function. This forces the output to assume only values between 0 and 1. and hence, we have:

$$P(y^{(i)} = 1) = \frac{1}{1 + \exp\left(-(\theta_0 + \theta_1 x_1)\right)} \tag{18}$$

Given a data:

$$D = \left\{ (x_i, y_i) \right\}_{i=1}^{n}$$

where $y \in \mathbb{R}^d$

And $y_i \in \{0, 1\}$ (discrete set), then we need a *Binary Classification.*

Most often, we would want to predict our outcomes as YES/NO (1/0). For example: Is your favorite football team going to win the match today? — yes/no (0/1) or Does a student pass in exam? — yes/no (0/1).

Thus, if the output is more tan 0.5 , we can classify the outcome as 1 (or YES) and if it is less than 0.5 , we can classify it as 0(or NO).

Some of the characteristics of the sigmoid function are as follows:

1. This curve has a finite limit of 0 as x approaches $-\infty$
2. This curve has a finite limit of 1 as x approaches ∞
3. On like the perceptron wit it outputs as 0 or 1 (discrete value), a sigmoid function outputs a more smooth or continous range of values between 0 and 1. hence, It is required in Neural Networks that the output changes very slowly with input.

For the logistic regression model the hypothesis is given by the following:

$$h_\theta(x) = g(\theta^T x) \tag{19}$$

Instead of solving a regression problem as we have we have seen in linear regression, logistic regression is used for a classification(i.e the output is discrete: 0 and 1) problem. So, it can be written as follow using the hypothesis.

$$p(y = 1; \theta) = h_\theta(x); \qquad p(y = 0; \theta) = 1 - h_\theta(x) \tag{20}$$

So, from 20, we have :

$$p(y; \theta) = h_\theta^y(x) * (1 - h_\theta(x))^{1-y}$$

Logistic regression with Maximum likelihood estimation

Just like the case of Linear regression, we also need to apply the maximum likelihood Estimation

$$L(\theta) = p(y; \theta) = \prod_{i=1}^{m} p(y_i; \theta)$$

$$= \prod_{i=1}^{m} h_\theta^{y_i}(x_i) * (1 - h_\theta(x_i))^{1-y_i}$$

The log-likelihood is is given by:

$$\mathcal{L}(\theta) = \log L(\theta)$$

$$= \sum_{i=1}^{m} \log(h_\theta^{y_i}(x_i) * (1 - h_\theta(x_i))^{1-y_i})$$

Hence, then the cost function for logistic regression is given as:

$$\sum_{i=1}^{m} (y_i \log h_\theta(x_i) + (1 - y_i) \log(1 - h_\theta(x_i))) \tag{21}$$

Now that we have our cost function, we can now proceed to calculation the gradient descent for out logistic regression. but before then, we need to fine the derivative of our hypothesis in equation 19

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + \exp(-\theta^T x)}$$

$$g(z) = \frac{1}{1 + \exp(-z)}$$

Using the chain rule differentiation

$$g'(z) = -(1 + \exp(-z)^{-2}). - \exp(-z)$$

$$g'(z) = (1 + \exp(-z)^{-2}). \exp(-z)$$

$$\frac{\exp(-z) + 1 - 1}{(1 + \exp(-z))^2}$$

$$\frac{\exp(-z) + 1 - 1}{1 + \exp(-z)} * \frac{1}{1 + \exp(-z)}$$

This can also be written as:

$$\left(\frac{\exp(-z)+1}{1+\exp(-z)} - \frac{1}{1+\exp(-z)}\right) * \frac{1}{1+\exp(-z)}$$

After factorization, we have:

$$\left(1 - \frac{1}{1+\exp(-z)}\right) * \frac{1}{1+\exp(-z)}$$

hence, the differentials of $g(z)$ is given as;

$$g'(z) = (1 - g(z)) * g(z) \tag{22}$$

How, we can go back to equation 21.

$$\frac{\partial}{\partial\theta}\mathcal{L}(\theta) = \left(y.\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right).\frac{\partial}{\partial\theta^T x}$$

Substituting equation 22 for $\frac{\partial}{\partial\theta^T x}$ recall that $z = \theta^T x$, hence, we have;

$$= \left(y.\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)(1-g(\theta^T x))g(\theta^T x) * \frac{\partial}{\partial\theta_i}\theta^T x$$

$$= \Big(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\Big)x_i$$

Finally, we have

$$(y - g(\theta^T x))x \tag{23}$$

Recall that,

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)}$$

Therefore, equation 23 can be written as:

$$\frac{\partial}{\partial\theta}\mathcal{L}(\theta) = (y - h_\theta(x)).x \tag{24}$$

Using the gradient of the cost function above, the gradient descent for logistic regression. is defined as;

Repeat until convergence {

$$\theta_j = \theta_j + \alpha\frac{\partial}{\partial\theta}\mathcal{L}(\theta)$$

}

3 Lecture 3

3.1 Risk Minimization

Ideally, we don't know how an algorithm would work in practice ***True Risk***, this is because we don't know the true distribution of the data, however, we measure the performance on a known set of training data and this is known as the ***Empirical Risk***. (the expected loss over a given example) But in trying to minimize the risk one may end up overfitting. *Risk Minimization* is simply the process of doing everything possible to reduce the probability of risk towards zero.

$$y = h^*_\theta(x) + \epsilon \tag{25}$$

$$y = h_\theta(x) \tag{26}$$

Suppose equation 25 is known as the *True Model* and equation 26 is the Model gotten for our training example, hence the Expected risk is given as:

$$\mathbb{E}\left[h_\theta^*(x) + \epsilon - h_\theta(x))^2\right] \tag{27}$$

The equation 27 can also be written as follows for easy computation

$$\mathbb{E}\left[h_\theta^*(x) - h_\theta(x) + \epsilon)^2\right] \tag{28}$$

Recall from our elementary expansion that $(a+b)^2 = a^2 + b^2 + 2ab$, now using this analogy, we have our $a = h_\theta^*(x) - h_\theta(x)$ and $b = \epsilon$, then we have

$$\mathbb{E}\left[(h_\theta^*(x) - h_\theta(x))^2 + \epsilon^2 - 2(h_\theta^*(x) - h_\theta(x))\epsilon\right]$$

Since Expectation is linear, hence we can have:

$$\mathbb{E}[(h_\theta^*(x) - h_\theta(x))^2] + \mathbb{E}[\epsilon^2] - 2\mathbb{E}(h_\theta^*(x) - h_\theta(x)) * \mathbb{E}[\epsilon]$$

Recall that our data are **(iid)**, meaning that they are *Identically*(all drawn from the same distribution) and *Independently* (the error on one does not affect the other i.e one is not a linear combination of the other) distributed, the follow a normal distribution as seen in equation 13. hence $\mathbb{E}[\epsilon] = 0$ and we are left with;

$$\mathbb{E}[(h_\theta^*(x) - h_\theta(x))^2] + \mathbb{E}[\epsilon^2] \tag{29}$$

$$\mathbb{E}[(h_\theta^*(x) - h_\theta(x))^2] + \sigma^2 \tag{30}$$

Recall that $Var(p) = \mathbb{E}[p^2] - (\mathbb{E}[p])^2$

hence,

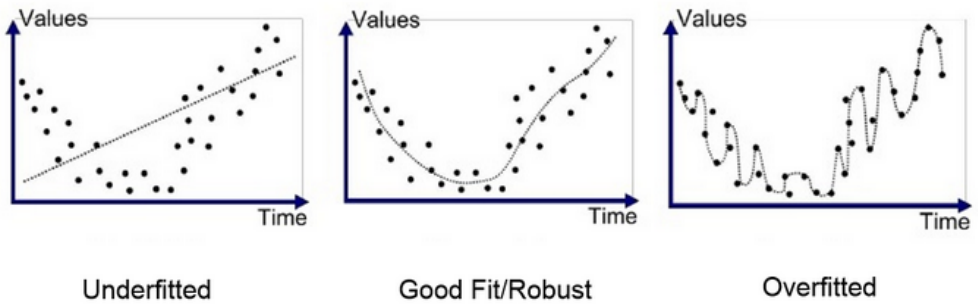
$$\mathbb{E}[p^2] = Var(p) + (\mathbb{E}[p])^2$$

Comparing the above analogy with equation 30, were our $p = (h_\theta^*(x) - h_\theta(x))^2$, we'll have;

$$\mathbb{E}[(h_\theta^*(x) - h_\theta(x))^2] = Var(h_\theta^*(x) - h_\theta(x)) + (\mathbb{E}[h_\theta^*(x) - h_\theta(x)])^2 + \sigma^2 \tag{31}$$

- σ^2 is known as the ***noise***, although we have no control over it, but however, if the value of σ^2 is too large, it means that the data is too noisy
- $\mathbb{E}[h_\theta^*(x) - h_\theta(x)]$ is known as the ***Bias***, this, we have control over. as it tells us the accuracy of ur model. if the value of $\mathbb{E}[h_\theta^*(x) - h_\theta(x)]$ is too large, it means that the model is underfitting meaning that, it is too simple.
- $Var(h_\theta^*(x) - h_\theta(x))$ is known as the ***Variance***, this, we also have control over. as it tells us the accuracy of ur model on unseen data. if the value of $Var(h_\theta^*(x) - h_\theta(x))$ is too large, it means that the model is overfitting meaning that, it is too complex.

When do we say a model is overfitting, or underfitting? the diagrams bellow would explain in details. The



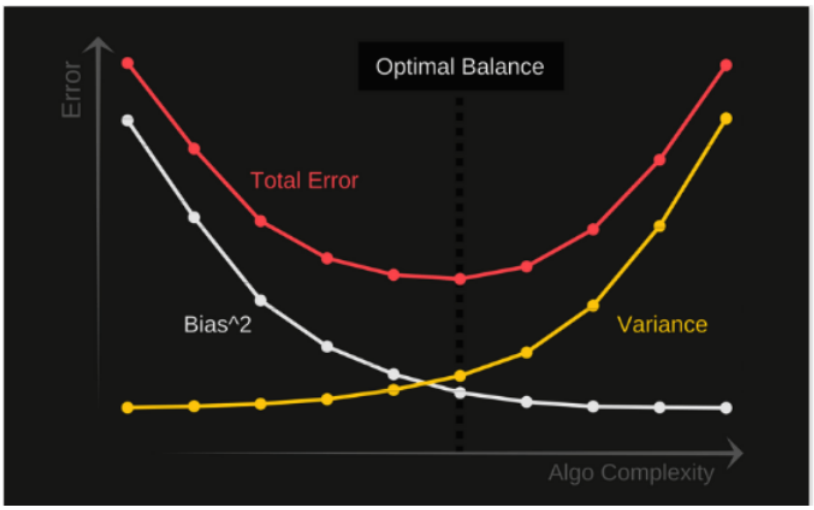
plot above displays the model of the time spent by a group of persons and the values the place on each movie watched. The first shows that the model underfiting because by looking at it, the line does not cover all the points shown in the graph. hence, the model sufficiently complex to capture the information on the trained data. it also performs poorly on both the trained and new data set. Some of the ways to reduce the variance is to train and train again for a long time; use a more complex model; and do some feature engineering by multiplying some of the features together to create more features that would yield better predictions.

The graph in the middle is called a good fit, because it captures almost all the information in the data set, it performance on both the trained and the test set is good.

The last model overfit's because it captures all the data points, leading to a high variance, this model might perform very well on the trained data set, but it performance on the test data is poor. hence, it cannot be generalized. Some of the ways to reduce overfitting is to reduce the complexity of the model; Do some feature selection; Regularization; and also adding more data.

Note that, while one is trying to reduce the variance, you might indirectly be increasing the bias, so how can one strike a balance between these two? this is what we call the ***Bias variance tradeoff***.

Total Error = Bias^2 + Variance + Irreducible Error



To build a good model, we need to find a good balance between bias and variance such that it minimizes the total error. This tradeoff in complexity is why there is a tradeoff between bias and variance.An optimal balance of bias and variance would never overfit or underfit the model.

4 Lecture 4

4.1 Newton's Method for Linear Regression

Newton's method has a quadratic rate of convergence and converges Newton's method has a quadratic rate of convergence and converges faster than gradient descent which has only sublinear rate of convergence. However, the drawback of Newton's method is the evaluation of the hessian matrix which we don't have to do for gradient descent.

$$\theta := \theta - \frac{f(\theta)}{f'(\theta)}$$

This method is actually faster than stochastic gradient descent, however, if the data is large, it is very expensive cause it's complexity is $O(d^3)$ compared to theta of gradient descent that is $O(d)$

4.2 Feature Selection

Features in the data set used in training machine learning models have a huge influence on the performance one can achieve. Having irrelevant or redundant features in ones data can decrease the accuracy of the models and make model learn based on irrelevant features. hence, the term ***Feature Selection*** is the process where one automatically or manually select those features which contribute most to ones prediction variable or output.

Benefits for making feature selection

- **Reduces Overfitting:** To reduce the complexity of a model, feature selection is one of the best recommended techniques, as the removal of redundant or irrelevant features can help reduce over fitting.
- **Improves Accuracy:** Less misleading data means modeling accuracy improves.
- **Reduces Training Time:** For algorithm to train faster, fewer data points helps reduce algorithm complexity and algorithms train faster.

Types of feature selection

- **Wrappers Method:** This method considers the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. A predictive model us used to evaluate a combination of features and assign a score based on model accuracy. There are 2 main types of the wrappers method. its complexity is $O(d^2)$
  - Forward Selection: Forward selection is an iterative method in which we start with having no feature in the model. In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model.
  - Backward Elimination: In backward elimination, we start with all the features and removes the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed on removal of features.
  - Recursive Feature elimination: It repeatedly creates models and keeps aside the best or the worst performing feature at each iteration. It constructs the next model with the left features until all the features are exhausted. It then ranks the features based on the order of their elimination.

  The algorithm is as follows (forward selection)
  Initialize $F_i = \emptyset$
  repeat until convergence:
  (a) for i in range (d):
  if $i \notin F$, let $F_i = F \cup i$
  use cross validation to evaluate F_i
  (b) Set F to be the best subset found at (a)

- **Filter Method:** For this method, the selection of features is independent of the machine learning algorithms. Instead, features are selected on the basis of their correlation tests. i.e the correlation between each feature with the outcome variable. The features with the most correlation with the target is considered to be the best. the Mutual information is given as:

$$MI\ (x^{(i)}, y) = KL\ (P(x^{(i)}, y))\ ||\ P(x^{(i)}P(y))$$

MI : Mutual inf

KL : Kulback Liebler Divergence.

Intuitively, mutual information measures the information that X X and Y shares. It measures how much knowing one of these variables reduces uncertainty about the other. For example, if X and Y are independent, then knowing X does not give any information about Y and vice versa, so their mutual information is zero. At the other extreme, if X is a deterministic function of Y and Y is a deterministic function of X then all information conveyed by X is shared with Y : knowing X determines the value of Y and vice versa.

5 References

1. Foundations of Machine Learning Class Note Book
2. Understanding Machine Learning book from Theory to Algorithm
3. what happens when probalility defines algebra?. By Hackernoon. https://hackernoon.com/probabilistic-interpretation-of-linear-regression-835de290dd46