Foundations of Machine Learning

A lesson note by Sewade O. Ogun

November 22, 2019, AIMS Ghana

Symbols and Notations

θ_i – Parameter i

Θ – Set of Parametersθ

n – Number of examples

d – Number of features

x_i – A single example at row i

x_i^j – A feature j at row i

y – Array of targets

D – Dataset consisting of X, y

$L(\Theta)$ – Loss on the entire dataset $L(\Theta)$ – Loss on the entire dataset ∂ – Partial derivative

∇ – Gradient

λ – Learning rate or preconditioner

$\mathbf{U}$ – Universal Set

$\mathcal{O}(n)$ – Worse case complexity, BigO

Chapter 1

Supervised Learning

1.1 Regression Problem

Given a set of datapoints, $D = \{(X_i, y_i)\}_{i=1}^{n}$ where x_i are the features and y_i are the target, corresponding to the features, we can learn a function or hypothesis which maps the features x_i to the target y_i i.e $h : \mathbb{R}^d \to \mathbb{R}^c$

Given a supervised learning task, it is required to understand the given problem at hand to determine;

a. Data

b. Hypothesis or model or Hypothesis class e.g. Regression

c. Criteria (Loss, Cost)

d. Learning Algorithm

1.1.1 Hypothesis

The hypothesis function is

$$h_\theta(X) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n = \sum_{i=0}^{n} \theta_i x_i \qquad (x_0 = 1) \tag{1.1}$$

where θs are parameters of the model and θ_0 is referred to as the **bias/intercept term**. Since this model has parameters, and the goal is to find the best parameters for the data, this form of models are referred to as **parametric models**. The other form are called **non-parametric models**.

1.1.2 Criteria

The criteria for the linear regression model is called **Ordinary Least Squares (OLS) or Mean Squared Error** because of the form of the criteria. This is given as;

$$L(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(h_\theta(x_i) - y_i\right)^2 \tag{1.2}$$

1.1.3 Learning Algorithm

The goal of the learning algorithm is to find the parameters θ that minimizes the loss function i.e.

$$\min_\theta L(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(h_\theta(x_i) - y_i\right)^2$$

Two approaches (Analytical method or Numerical methods) can be used to solve this problem.

Numerical Approach

Let Θ be the vector of θs and $Theta_0$ the initial solution, we can update Θ by repeatedly subtracting the gradient of the loss function from Θ until convergence i.e.
Repeat

$$\theta_j := \theta_{j-1} - \alpha\frac{\partial}{\partial\theta_{j-1}}L(\theta_{j-1}) \tag{1.3}$$

where α is the **learning rate** and is an hyperparameter to be tuned. This form of learning in (1.3) is known as **gradient descent**. It involves taking steps in the opposite direction of steepest ascent gotten through the gradient.

Computing the Gradient of Loss Function

For one example, $h_\theta(x_i) = \Theta x_i$

$$\begin{aligned}\frac{\partial}{\partial\theta}L(\theta) &= \frac{\partial}{\partial\theta}\frac{1}{2}\left(h_\theta(x_i) - y_i\right)^2 \\ &= \frac{\partial}{\partial\theta}\frac{1}{2}\left(\theta_i x_i - y_i\right)\frac{\partial}{\partial\theta}\left(\theta x_i - y\right) \\ &= \left(\theta_i x_i - y_i\right)x_i^j \\ &= \left(h_\theta(x_i) - y_i\right)x_i^j\end{aligned}$$

where x_i^j is the jth feature of the ith example. Therefore the update rule can be modified as; $\theta_j = \theta_j + \alpha(y_i - h_\theta(x^{(i)}))x_i^j$. The equation indicates that for a fixed learning rate, the amount of update is dependent on the difference between predictions and target. This is called the **Widrow-Hoff update rule** or **least mean square rule**.

Batch Gradient Descent

Gradient descent in general assumes that the loss function is differentiable and convex. If the loss function is not convex, the solution might get stuck in a local optimum and not find the global optimum. For linear regression, we can show that the loss function is convex as it is a function of 2nd polynomial and positive.
The batch gradient descent uses all the examples on each iteration to compute the gradient. For a large set of examples, this can be computationally expensive and not used in practice. This also requires that the gradient is computed for every example before making one step of gradient descent.

Repeat until convergence;

$\Theta = \Theta + \alpha\sum_{i=1}^{n}\left(y_i - h_\theta(x^{(i)})\right)x_i^j$

Stochastic Gradient Descent

Stochastic gradient descent selects a sample at random from the examples and uses this to update the gradient. This has the advantage of being fast and requiring less memory but does not guarantee convergence to the optimal solution (even though an approximate solution is usually good enough in practise) after the update. Also, update is not smooth as the direction of descent is dependent on only one data which may be noisy.

Repeat until convergence;
for i in 1 to n;
$\Theta = \Theta + \alpha \left(y_i - h_\theta(x^{(i)})\right) x_i^j$

Mini-batch Stochastic Gradient Descent

A compromise between batch gradient descent and stochastic gradient descent is the mini-batch gradient descent. It uses a mini-batch of examples samples at random from the dataset to update the parameters. This has the advantage of reducing the variance during update and therefor smoother. It also maximixes the usage of hardware and cpu vectorization for efficient computations.

Repeat until convergence;
for k in 1 to no. of minibatches;
$\Theta = \Theta + \alpha \sum_{i=1}^{mbs} \left(y_i - h_\theta(x^{(i)})\right) x_i^j$
where mbs is the minibatch size.

Analytical Method

The analytical method performs a single computation using the data matrix. It the does not require to be performed iteratively and does not require a learning rate.

$$
\begin{aligned}
L(\theta) &= \frac{1}{2}\|X\Theta - y\|^2 \\
&= \frac{1}{2}(X\Theta - y)^T(X\Theta - y) \\
&= \frac{1}{2}(\Theta^T X^T - y^T)(X\Theta - y) \\
&= \frac{1}{2}(\Theta^T X^T - y^T)(X\Theta - y) \text{ as } (A-B)^T = A^T - B^T \text{and } (AB)^T = B^T A^T \\
&= \frac{1}{2}(\Theta^T X^T X\Theta - 2y^T X\Theta + y^T y)
\end{aligned}
$$

The gradient of $L(\Theta)$ is then computed as;

$$\nabla_\theta = \frac{1}{2}\left(2X^T X\Theta - 2X^T y\right)$$

$$X^T X\Theta = X^T y$$

$$\Theta = \left(X^T X\right)^{-1} X^T y \tag{1.4}$$

(1.4) is known as the **normal equation** and requires the inverse of $X^T X$ to be computed. $X^T X$ may not be invertible and can be expensive to compute depending on the size of the dataset. To avoid the invertibility problem, the pseudoinverse can be computed and we can manually remove correlated features (by computing correlation matrix) if the features are linearly dependent. Another approach is to add regularization. To regularize the model, the sum of squares of the parameters is added to the loss function. This has the effect of also reducing the size of values of the parameters if the **regularization parameter** λ is set appropraitely.

$$L(\theta) = \frac{1}{2}\|X\Theta - y\|^2 + \lambda \|\Theta\|_2^2 \tag{1.5}$$

$$\Theta = \left(X^T X + \lambda I\right)^{-1} y^T X \tag{1.6}$$

The proof of these regularization is shown in a later chapter using the Maximum a posteriori extimation(MAP). I is the identity matrix and $\left(X^T X + \lambda I\right)^{-1}$ always exist

Newton's Method

The newton's method is a second-order optimization algorithm. Consider the diagram 1.1 below,

. We want to find the point on the curve where the function is a minimum. Newton's method gives the formula as $\Theta := \Theta - \frac{f(\theta)}{f_i}$. For our machine learning task, this implies looking for the parameters Θ where the gradient is zero.

$$g = f'$$

$$g' = f''$$

$$\Theta := \Theta - \frac{g}{g_i}$$

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

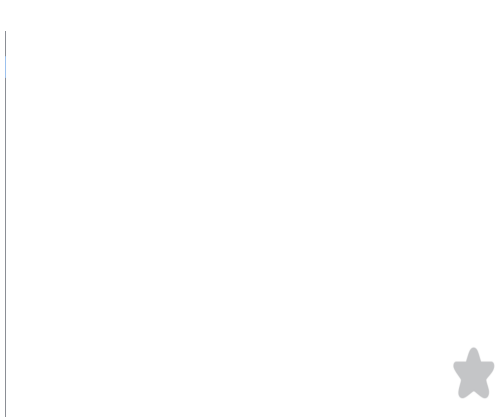

Figure 1.1:

In vecorized form, $\Theta := \Theta - H^{-1}\nabla_\theta \mathcal{L}$ where $H_{ij} = \dfrac{\partial}{\partial_{\theta_i}\theta_j}\mathcal{L}(\Theta)$. H_{ij} is the second partial derivative wrt all the Θs and is called Hessian matrix. If $x \in \mathbb{R}^{(n,d)}$, the $H \in \mathbb{R}^{(d,d)}$

1. Newton's method is faster than stochastic gradient descent and converges in polynolial time.
2. It requires no learning rate and instead takes into account the curvature of the loss function during optimization.
3. A disadvantage is that inverting the Hessian is computationally expensive and the bigO is dependent on the number of parameters; $\mathcal{O}(d^2)$
4. To make Hessian invertible where the matrix is singular, a preconditioner is applied; $H = H + \varepsilon I$ where ε is a small number and values range from $10e^{-8}$ to $10e^{-10}$ in practise.

1.1.4 Maximum Likelihood Estimation

Maximum Likelihood estimation is a way of estimating the best parameters of a model given the data by maximizing the probability of the data.
Given that $y_i = \Theta^T x^i + \varepsilon_i$ where $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$; the features are assumed to be centered with $\mu = 0$ and ε_i is the difference between predictions and target which is a random variable and is Independently and Identically Distributed (IID).

We know that for a normally distibuted random variable, $PDF = \frac{1}{\sqrt[2]{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$

Since ε_i is assumed to be normally distributed,

$$P(\varepsilon_i) = \frac{1}{\sqrt[2]{2\pi\sigma^2}} e^{-\frac{\varepsilon_i^2}{2\sigma^2}}$$

$$P(\varepsilon_i) = \frac{1}{\sqrt[2]{2\pi\sigma^2}} e^{-\frac{(y_i - h_\theta(x_i))^2}{2\sigma^2}}$$

For one example, $P(y_i|x_i;\Theta) = P(\varepsilon_i)$ and is read as Probability of y_i given x_i parameterized by theta equals probability of the error in estimation. For all examples,

$$\prod_{i=1}^{n} P(y_i|x_i;\Theta) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(y_i - h_\theta(x_i))^2}{2\sigma^2}}$$

Taking the log of both sides, (to eliminate the exponent),

$$log \sum_{i=1}^{n} P(y_i|x_i;\Theta) = log \sum_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} + \sum_{i=1}^{n} -\frac{(y_i - h_\theta(x_i))^2}{2\sigma^2}$$

Combining all the constant elements of the equation,

$$P(y_i|x_i;\Theta) = \sum_{i=1}^{n} -\frac{(y_i - h_\theta(x_i))^2}{2\sigma^2} + C$$

since logarithm of a function is a monotonically increasing transformation of the function.

Maximixing the log likelihood of $P(y_i|x_i;\Theta)$ is equivalent to **minimizing** the loss.

$$\min_\theta L(\Theta) = \sum_{i=1}^{n} (y_i - h_\theta(x_i))^2$$

and is same output as the criteria specified in (1.2).

1.2 Classification Problem

Given a set of datapoints, $D = \{(X_i, y_i)\}_{i=1}^{n}$ where x_i are the features and $y_i \in \{0, 1\}$ or $y_i \in \{\text{-}1, 1\}$ are the targets, corresponding to the features. In regression, the goal is to estimate a continuous variable while the estimated targets in classification are discrete. If y has two classes, it is called **binary classification** while for classes greater than two, it is called **multi-class classification**. The goal is to learn a function or hypothesis which maps the features to classes y.

1.2.1 Hypothesis

Logistic Regression

Logistic regression aims to fit a linear hyperplane separating the datapoints. An **hyperplane** is 1 dimension less than the plane in which it lives.

The hypothesis is given by $h_\theta(x) = \frac{1}{1 + e^{-\Theta^T x}}$. This assures that the predictions are bounded by 0 and 1. The form of the hypothesis is referred to as the sigmoid function or the logistic function. The sigmoid function is shown in 1.2 and the general form is

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

Criteria

$$P(y = 1|x; \Theta) = h_\theta(x)$$

Figure 1.2:

$$P(y = 1|x; \Theta) = 1 - h_\theta(x)$$

In compact form,

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

This equation relates to the bernoulli distribution equation. We can use the maximum likelihood estimation principle to get the criteria for the logistic regression as follows.

$$\mathcal{L}(\Theta) = P(y|X; \Theta)$$

$$\mathcal{L}(\Theta) = \prod_{i=1}^{n} P(y_i|x_i; \Theta)$$

$$\mathcal{L}(\Theta) = \prod_{i=1}^{n} h_\theta(x_i)^{y_i} * (1 - h_\theta(x_i))^{1-y_i}$$

$$\mathcal{L}(\Theta) = \sum_{i=1}^{n} (y_i log(h(x_i)) + (1 - y_i) log(1 - h_\theta(x_i)))$$

In similar terms, maximizing the likelihood is equivalent to minimizing the negative log likelihood.

$$arg\max_{\theta} \mathcal{L}(\Theta) = arg\min_{\theta} \ -\mathcal{L}(\Theta)$$

$$L(\Theta) = -\sum_{i=1}^{n} (y_i log(h(x_i)) + (1 - y_i) log(1 - h_\theta(x_i))) \tag{1.7}$$

(1.7) is known as the cross-entropy loss function. $\mathcal{L}(\Theta)$ is the log-likelihood and $L(\Theta)$ is the loss.

Learning Algorithm

Recall that $\Theta := \Theta + \alpha \nabla_\theta \mathcal{L}(\Theta)$ and $h_\theta(x) = g(\Theta^T \boldsymbol{x}) = \dfrac{1}{1 + e^{-\Theta^T \boldsymbol{x}}}$

$$\mathcal{L}(\Theta) = \sum_{i=1}^{n} (y_i log(h(x_i)) + (1 - y_i) log(1 - h_\theta(x_i)))$$

Computing the Gradient

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

But the derivative of sigmoid $g^1(z) = g(z).(1 - g(z))$ This implies

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

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

$$\frac{\partial}{\partial \theta} \mathcal{L}(\Theta) = (y - h_\theta(x)) x^j$$

The update rule for one example then becomes;

Repeat until convergence;
for i = 1 to n;

$\Theta := \Theta + \sum_{i=1}^{n} \alpha(y_i - h_\theta(x_i))x_i^j$

Derivative of the sigmoid function

$$
\begin{aligned}
g(x) &= \frac{1}{1+e^{-x}} \\
g^{'}(x) &= \frac{\delta}{\delta x}\,(1+e^{-x})^{-1} \\
&= -1(1+e^{-x})^{-2}(-e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{(1+e^{-x})}.\frac{(1+e^{-x})-1}{(1+e^{-x})} \\
&= \frac{1}{1+e^{-x}}.\left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) \\
&= \frac{1}{1+e^{-x}}.\left(1 - \frac{1}{1+e^{-x}}\right) \\
g^{'}(x) &= g(x).(1-g(x))
\end{aligned}
$$

1.3 Risk Minimization

Given $(x_i, y_i) \sim P$ i.e. x_i, y_i are samples observed from the probability distribution P, we want to minimize the expectation of squared errors. This is called the **True Risk Minimization**.

$$\min_\theta \mathbb{E}_{(x,y)\sim P}(h_\theta(x) - y)^2$$

Since we do not know the distribution and only have samples from it, we instead minimize the empirical equivalent known as the **Empirical Risk Minimization**.

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} (h_\theta(x) - y)^2$$

$y = h_{\theta^*} + \varepsilon$
h_{θ^*} is the best prediction possible which we desire to find.
h_θ is the prediction made by our model
$\varepsilon \sim \mathbb{N}(0, \sigma^2)$ called the bayes (optimal) error

$$\begin{aligned}
R_{h_\theta} &= \mathbb{E}[(y - h_\theta(x))^2] \\
&= \mathbb{E}[(h_{\theta^*} + \varepsilon - h_\theta(x))^2] = \mathbb{E}[(\varepsilon + (h_{\theta^*} - h_\theta(x))^2] \\
\text{Let } b = (h_{\theta^*} - h_\theta(x))^2 & \\
&= \mathbb{E}[\varepsilon_i^2 + 2\varepsilon_i b + b^2] \\
&= \mathbb{E}[\varepsilon^2] + \mathbb{E}[2\varepsilon_i b] + \mathbb{E}[b^2] \\
\text{Recall;} & \\
Var(x) &= \mathbb{E} - [\mathbb{E}(x)]^2 \\
&= \mathbb{E}[(x - \mu)^2] \text{ where } \mathbb{E}(x) = \mu \\
&= \mathbb{E}(x^2 - 2\mu x + \mu^2) \\
&= \mathbb{E}(x^2) - 2\mu\mathbb{E} + \mu^2 \\
&= \mathbb{E}(x^2) - 2\mu^2 + \mu^2 \\
&= \mathbb{E}(x^2) + \mu^2 \\
&= \mathbb{E}(x^2) - [\mathbb{E}(x)]^2 \\
\text{Plugging in Var}(x); R_{h_\theta} &= Var(\varepsilon) + [\mathbb{E}(\varepsilon)]^2 + 2\mathbb{E}(\varepsilon)\mathbb{E}(b) + Var(b) + [\mathbb{E}(b)]^2 but \mathbb{E} = 0 \\
&= Var(\varepsilon) + Var(b) + [\mathbb{E}(b)]^2 \\
&= \sigma^2 + Var(b) + [\mathbb{E}(b)]^2 \\
&= \sigma^2 + Var(h_{\theta^*}(x) - h_\theta x) + [\mathbb{E}(h_{\theta^*}(x) - h_\theta(x))]^2
\end{aligned} \tag{1.8}$$

The risk is composed of noise σ^2, bias $[\mathbb{E}(h_{\theta^*}(x) - h_\theta(x))]^2$ and variance $Var[h_{\theta^*}(x) - h_\theta x]$ of the model. The noise σ^2 is derived from the data and not under the model's control, and is also the minimum loss value our model can have. It is also a requirement to have low values for both variance and bias.

Reducing the variance when training the model increases the bias of the model and vice-versa. The solution is to find a model with the right complexity which reduces the variance and bias simultaneously. This phenomenon is called the **Bias-Variance Tradeoff**.

For high bias model, we can (1) train the model for a longer period (2) use a more complex model (3) increase the learning rate
For high variance model, we can (1) get more training data (2) make the model simpler (3) reduce the number of features (4) add regularization.

In practise, the data is split into training and validation sets (but slitting the data into two folds also has the effect of making us loose information in the validation set that could have been used by the model).
A model is said to be overfitting if it has a high variance and is seen where the training set loss is low while the validation set loss is high. A model is said to be underfitting if training set loss is high and validation set loss is also high.
To cater for the effect of data splitting, **cross-validation** method is used to choose the best model.

K-Fold Cross Validation

Given the set of data $D = \{(x_i, y_i)\}_{i=1}^{m}$ and the criteria such as $L(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(h_\theta(x_i) - y_i\right)^2$ for linear regression;

1. Randomly split D into k disjoint subsets of size $\frac{m}{k}$

$$D = \bigcup_{i=1}^{k} D_i$$

2. For each model M_i,
   For each j = 1, ..., k
   Train $M_{ij} on D - D_j$
   Test on D_j and compute $L_{D_j}(M_{ij})$

3. Compute generalization error of M_i = average of $L_{D_j}(M_{ij})$

4. Pick the model M with the lowest generalization error i.e. model that has reduced risk R_{h_θ}

5. Put all the data together i.e retrain D using the best model.

An extreme case of the K-Fold cross validation where all the data are split into n folds is called **Leave-One-Out Cross Validation**. In this case, only one example is used for validation at each instance and can be used when the number of examples, n is not large.

Hints

1. Typically, $k = 10$ for large dataset and $k = 3$ for smaller dataset.
2. Number of folds should be less than number of examples $k \leq n$
3. If the number of examples is very large, say $n = 1000000$, cross validation may not be necessary provided the validation set can be representative.

1.3.1 Reducing the Model Complexity

Feature Selection

Feature selection is done to (1) reduce the model complexity (2) determine the features that are most predictive (3) reduce the number of features, preventing overfitting.
For d features, the number of feature subsets = 2^d. There are two feature selection methods, namely (1) Wrapper Methods (2) Filter Methods.

Feature Selection with Wrapper Methods

a. **Forward Search** It begins with an empty set and proceeds to find most predictive combination of features from all the feature subsets.

   1. Intialize $F = \emptyset$
   2. Repeat {
      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)

3. Select the best subset that was found during the entire procedure. }

For d features, complete forward search takes $\mathcal{O}(d^2)$

b. Backward Search

1. Initialize $F = \mathbf{U}$, the set of all features
2. Repeat {
   a. For i in range(d)
      - if i $\in F$, let $F_i = F - \{i\}$
      - Use cross-validation to evaluate F_i

   b. Set F to be the best subset found at (a)
3. Select the best subset that was found during the entire procedure. }

Feature Selection with Filter Methods

Filter methods measure correlation between each feature and target. The features with more correlation are more predictive of the target variable. A type of correlation commonly used is called **Mutual Information**.

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

where MI is the mutual information, KL is Kulback-Leibler Divergence (which is also known as relative entropy).

1. Compute MI
2. Take k set of features with largest correlation

Advantage: Complexity of filter methods is linear with time i.e. $\mathcal{O}(d)$ for d features.
Disadvantage: It does not take into account cross-correlation between features.

A combination of Wrapper method and filter method can be used to select the most predictive features, combining their relative strength.