CS 217: Artificial Intelligence and Machine Learning **Jan-Apr 2024**

Lecture 4: Regression

Lecturer: Swaprava Nath *Scribe(s): Tejas Sharma*

4.0.1 Recap

Let $\vec{x} = (1, x_1, x_2, \ldots, x_d)$. If we are given n points $(\vec{x}_1, y_1)$, $(\vec{x}_2, y_2)$, $\ldots$ $(\vec{x}_n, y_n)$ then the solution vector w is the optimum solution for $\|\mathbf{X} \cdot \vec{w} - \vec{y}\|^2$.

Note that $\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{bmatrix}$ and $\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$. We are to optimize $f(x) = \vec{w}^T \cdot \mathbf{X}$.

From this and a bit of linear algebra, we got $\vec{w} = \left(\mathbf{X^T} \cdot \mathbf{X}\right)^{-1} \cdot \mathbf{X^T} \cdot \vec{y}$.

4.1 Solutions for Linear equation $\mathbf{A}\vec{x} = \vec{b}$

If $\vec{a_1}, \vec{a_2}, \ldots \vec{a_d}$ are the columns of $\mathbf{A}$, can we say that $\sum_{i=1}^{d} x_i \vec{a_i} = \vec{b}$?

We plot $\vec{a_i}$ and $\vec{b}$ in the d dimentional space. If the span of vectors $a_i \quad \forall i$ covers b then we are done. This is the geometric representation.

One Linear Algebra method is to pre-multiply both sides by $\mathbf{A^T}$ and then pre-multiply by $\left(\mathbf{A^T} \cdot \mathbf{A}\right)^{-1}$.

Not always will it give an answer though. Sometimes we may get a pseudo-inverse (left-inverse) matrix but the matrix $\mathbf{A^T} \cdot \mathbf{A}$ may be singular. This is especially true if $n < d$.

Also, this method may give a solution even if there is none, especially if $n > d$, wherein we project the scenario onto a lower dimentional space and find a solution there.

Alternatively, we could use Gauss-Jordan Algorithm. In fact, there is an algorithmic way to optimize convex functions in deep learning, known as **gradient descent**.

4.2 Basis Function

Let us assume the input variable n sample points (x_i, y_i) with input variable x as scalar (1-dimentional) for now. Let us now try to equate y with a polynomial in x. Specifically, Let $\vec{w} = (w_0, w_1, \ldots w_m)$.

We have, $y_i = w_0 + w_1 \cdot x_i + w_2 \cdot x_i^2 + \cdots + w_m \cdot x_i^m$. In other words, $y_i = \sum_{j=0}^{m} \left(w_j x_i^j\right)$.

The above is for a single point (x_i, y_i). We try to optimize $\vec{w}$ to fit all points in a polynomial curve.

Let $\vec{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_m \end{bmatrix}$ and $\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^m \end{bmatrix}$ and $\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$. Vector $\vec{w}$ would look like $\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{bmatrix}$.

We are to find the same $\vec{w}$ so as to minimize the error function $\|\mathbf{X} \cdot \vec{w} - \vec{y}\|^2$.

Over d dimentions, $y_i = \sum_{j=0}^{m} \left(\vec{w_j}^T \cdot \vec{x_i}^j \right)$ where $\vec{w_j} = \begin{bmatrix} w_{j,1} \\ w_{j,2} \\ \vdots \\ w_{j,d} \end{bmatrix}$ and $\vec{x_i}^j = \begin{bmatrix} x_{i,1}^j \\ x_{i,2}^j \\ \vdots \\ x_{i,d}^j \end{bmatrix}$.

If we define $\mathbf{X_k} = \begin{bmatrix} 1 & x_{1,k} & x_{1,k}^2 & \ldots & x_{1,k}^m \\ 1 & x_{2,k} & x_{2,k}^2 & \ldots & x_{2,k}^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,k} & x_{n,k}^2 & \ldots & x_{n,k}^m \end{bmatrix}$, we need to find $\vec{w_k} = \begin{bmatrix} w_{0,k} \\ w_{1,k} \\ \vdots \\ w_{m,k} \end{bmatrix} \quad \forall k \in [1, d]$

such that we minimize $\|\vec{y} - \sum_{k=1}^{d} \mathbf{X_k} \cdot \vec{w_k}\|^2$

4.3 Probabilistic Model of Linear Regression

Let the linear model be $y_i = wx_i + \epsilon_i$ where $\epsilon_i \sim N(0, \sigma^2)$ is a gaussian random variable, for the noise.

Our objective is to extract w to fit this noise. **Note**: all noises are independent i.e. covariance $Cov(\epsilon_i, \epsilon_j) = 0$.

How do we estimate from this?

4.3.1 Machine Learning Estimation

S set of n independent and identical random variables or observations $\{y_i\}$ are generated by a probabilistic model parameterized by θ.

$y_i \sim P(y|\theta)$. This is the likelihood function. We look at the logarithm of the likelihood function often.

We try to maximize $log\left(P(y|\theta)\right)$ by varying θ. We have done this in Data Analysis course.

There are two reasons why we use log-likelihood:

- It is mathematically nicer in many scenarios. Moreover, the probability is the product of the probabilities given θ for all points y_i. Instead, the logarithm makes it a sum.
- Numerical advantage. The probability goes close to zero even at the maximum for large n. However, the logarithm is still just a large negative number that we want to maximize (with sign, so minimize magnitude of negative number).

Let the estimate of $\theta = \theta_{MLE} = max_\theta \sum_{i=1}^{n} log\left(P(y_i|\theta)\right)$.

4.3.2 Case: Bernoulli Distribution, Coin Toss

Let a biased coin have $P(H) = \theta$. Out of n observations each of m tosses, let y_i of them be heads.

Given θ, we have $P(y|\theta) = \theta^y \cdot (1-\theta)^{m-y}$ and $\theta_{MLE} = max(\sum_{i=1}^{n} (y_i \cdot log(\theta) + (m - y_i) log(1-\theta))$.

On solving by differentiating, we get $\theta = \frac{\sum_{i=1}^{n} y_i}{mn}$ as the answer.

4.3.3 Back to Maximum Likelihood for Regression

$$\begin{aligned} y_j &= wx_j + \epsilon_j \\ \implies \quad y_j &\sim N\left(wx_j, \sigma^2\right) \\ P\left(y_j | x_j, \vec{w}\right) &= \frac{1}{\sqrt{2\pi\sigma^2}} \cdot exp\left(-\frac{(y_j - wx_j)^2}{2\sigma^2}\right) \\ L(\vec{w}) &= \text{const} - \sum_{j=1}^{n} \left(\frac{(y_j - wx_j)^2}{2\sigma^2}\right) \end{aligned}$$

On minimizing this function, we get a quadratic expression in w. It can be solved easily.

However, this gets increasingly tough if we add a constant to the linear expression i.e. $y = w_0 + w_1 x + \epsilon$ where $\epsilon \sim N(0, \sigma^2)$ or a polynomial that we try to find with epsilon given, for quadratic equations in multiple variables are hard to solve.

4.4 Gradient Descent

We keep subtracting the product of a constant η and the derivative expression ∇ until we reach near the minimum. We have seen something like this in CS101.

In other words, we have a point $\vec{x}$. We iteratively find $\vec{x} - \eta\vec{\nabla}x$. How is η, another parameter defined, is a different problem and is something that requires the machine to figure out based on the context.

On doing so, the point $\vec{x}$ approach the minima of a d dimentional convex function, assuming it exists. Of course, this may not find the exact minima but with something very similar to binary search towards the end, we reduce η to get $\vec{x}$ arbitrarily close.