Mathematics for Machine Learning - Module 3

Open Data Science Conference (ODSC West 2020)

Adewale (Wale) Akinfaderin
{@waleakinfaderin}

October 27, 2020

Talk Outline

1. About Me
2. Part I - Linear Regression Cost Function with MLE
3. Part II - Logistic Regression Cost Function with MLE
4. Part III - L2 Regularization Cost Function with MAP (Ridge)
5. Part IV - L1 Regularization Cost Function with MAP (LASSO)

Talk Outline

About Me

- Data Scientist, Amazon Web Services.
- Google Developer Expert, Machine Learning.
- Former experience developing and productionizing ML algorithms in the retail and renewable energy space.
- Doctoral level research in Magnetic Resonance Spectroscopy.
- Interested in democratizing ML, ML for Developing World and Neural Machine Translation for Low Resource Languages
- Committee and Reviewer NeurIPS Workshop Machine Learning for the Developing World (2019, 2020)
- I really like soccer and scrabble.

Talk Outline

Motivation

1. In supervised ML, cost functions are used to measure a trained model's performance.
2. Mean Squared Error (MSE) and Cross Entropy for binary classification problems are commonly used.
3. We'll shed some light on cost functions by deriving them using Maximum Likelihood Estimation (MLE) and Maximum a Posteriori (MaP).

Background

Background

Given a training set of n examples $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(3)})...(x^{(n)}, y^{(n)})$ where $x^{(i)}$ is the feature vector for the i^{th} training example and $y^{(i)}$ is its target, the goal of supervised learning is to learn a model $f : \mathcal{X} \rightarrow \mathcal{Y}$, a mapping that given $x \in \mathcal{X}$ outputs a prediction $y \in \mathcal{Y}$. $\mathcal{X}$ and $\mathcal{Y}$ are called the input space and output space respectively.

Background

Cost Function

In order to measure how well the model fits our training data, we define a loss function. For training example $(x^{(n)}, y^{(n)})$, the loss $\mathcal{L}(y^{(i)}, \hat{y}^{(i)})$ measures how different the model's prediction $\hat{y}^{(i)}$ is from the true label or value. The loss is calculated for all training examples, and its average taken. This value is called the cost function and is given by:

$$\text{Cost} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right)$$

Background

Parameter Estimate

The model f usually has some unknown parameter θ (In general, θ is a vector of parameters) which we will try to estimate using the training set. We can frame supervised learning as an optimisation problem - that is, we estimate the value of θ by picking the value that minimises the cost function we chose for our problem. Let us call this parameter estimate $\hat{\theta}$.

$$\hat{\theta} = \arg\min_{\theta}(\text{Cost})$$

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right)$$

Definitions

What is Maximum Likelihood Estimate?

Maximum Likelihood Estimation (MLE) is a method of estimating the unknown parameter θ of a model, given observed data. It estimates the model parameter by finding the parameter value that maximises the likelihood function. The parameter estimate is called the maximum likelihood estimate $\hat{\theta}_{MLE}$.

Definitions

Likehood Function

Given a set of independent and identically distributed (i.i.d) random variables $X_1, X_2, X_3, ...X_n$ from a probability distribution P_θ with parameter θ. We assume the random variables have a joint probability density function $f(x_1, x_2, \cdots, x_n \mid \theta)$. Suppose $x_1, x_2, ..., x_n$ are the observed values of the random variables, we define the likelihood function, a function of parameter θ as:

$$\mathcal{L}(\theta \mid x_1, x_2, \cdots, x_n) = f(x_1, x_2, \cdots, x_n \mid \theta)$$

$$\mathcal{L}(\theta \mid x_1, x_2, \cdots, x_n) = f(x_1 \mid \theta) \cdot f(x_2 \mid \theta) \cdots f(x_n \mid \theta)$$

$$= \prod_{i=1}^{n} f(x_i \mid \theta)$$

Log Likelihood

It is often easier to work with the natural logarithm of the likelihood function, called the log-likelihood. Recalling the product rule of logarithms, $\log(a.b) = \log(a) + \log(b)$. If we apply the product rule to the equation by taking its natural logarithm, it becomes:

$$\begin{aligned}\log\left(\mathcal{L}\left(\theta \mid x_1, x_2, \cdots, x_n\right)\right) &= \log\left(f\left(x_1 \mid \theta\right) \cdot f\left(x_2 \mid \theta\right) \cdots f\left(x_n \mid \theta\right)\right) \\ &= \log\left(f\left(x_1 \mid \theta\right)\right) + \cdots + \log\left(f\left(x_n \mid \theta\right)\right) \\ &= \sum_{i=1}^{n} \log\left(f\left(x_i \mid \theta\right)\right)\end{aligned}$$

Maximizing the Log Likelihood

To find the value of θ that maximises the likelihood function, we find its critical point, the point at which the function's derivative is 0. That is:

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta} = 0$$

Since logarithm is a monotononically increasing function, that is, if $x_1 > x_2$, then $\log(x_1) > (x_2)$, the value that maximises the likelihood is also the value that maximises the log-likelihood function.

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \mathcal{L}\left(\theta \mid x_1, \cdots, x_n\right) = \arg\max_{\theta} \log\left(\mathcal{L}\left(\theta \mid x_1, \cdots, x_n\right)\right)$$

Deriving Mean Squared Error(MSE) using MLE

Mean Squared Error is the cost function commonly used for regression. Given a set of n training examples of the form $(x^{(i)}, y^{(i)})$, MSE is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

where $x^{(i)}$ is the feature vector, $y^{(i)}$ is the true output value, and $\hat{y}^{(i)}$ is the regression model's prediction for the i^{th} training example.

Regression Model

Given a vector of predictor variables $X = (X_1, X_2, \cdots, X_p)$, and quantitative outcome variable Y, linear regression assumes that there is a linear relationship between the population mean of the outcome and the predictor variables. This relationship can be expressed by:

$$\mathbb{E}(Y \mid X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

$$\begin{aligned} Y &= \mathbb{E}(Y \mid X) + \epsilon \\ &= \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon \end{aligned}$$

$$\begin{aligned} y^{(i)} &= \beta_0 + \beta_1 x_1^{(i)} + \cdots + \beta_p x_p^{(i)} + \epsilon \\ &= \beta^\top x^{(i)} + \epsilon \\ &= \hat{y}^{(i)} + \epsilon \end{aligned}$$

Probability Density Function

Estimating $\beta_1, \beta_2, \cdots, \beta_p$ using the training data is an optimisation problem that we can solve using MLE by defining a likelihood function. Since ϵ is normally distributed $\epsilon \sim \mathcal{N}(0, \sigma^2)$, our outcome variable y will also be normally distributed. So, $y \sim \mathcal{N}(\beta^\top x, \sigma^2)$ The probability density function of the normal distribution (parameterised by μ: mean, and σ^2: variance) is given by:

$$f\left(y \mid \mu, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}$$

Therefore, if we replace the parameter μ with the mean of y, we get:

$$f\left(y \mid \beta^\top x, \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\beta^\top x)^2}{2\sigma^2}}$$

Derivation

The log-likelihood function will then be:

$$\log\left(\mathcal{L}\left(\beta \mid x^{(1)}, x^{(2)}, \cdots, x^{(n)}\right)\right) = \sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(y-\beta^\top x^{(i)}\right)^2}{2\sigma^2}}\right)$$

$$= n\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \sum_{i=1}^{n} \frac{\left(y-\beta^\top x^{(i)}\right)^2}{2\sigma^2}$$

Since we need to take the derivative of log-likehood function with respect to β to find the maximum likelihood estimate of β, we can remove all the terms without β, so our equation becomes:

$$\log\left(\mathcal{L}\left(\beta \mid x^{(1)}, x^{(2)}, \cdots, x^{(n)}\right)\right) = -\sum_{i=1}^{n} \left(y-\beta^\top x^{(i)}\right)^2$$

$$= -\sum_{i=1}^{n} \left(y-\hat{y}^{(i)}\right)^2$$

Cost Function Derivation

The value of β that maximises the equation is the maximum likelihood estimate $\hat{\beta}_{MSE}$. That is,

$$\hat{\beta}_{MSE} = \arg\max_{\beta} \left[-\sum_{i=1}^{n} \left(y - \hat{y}^{(i)} \right)^2 \right]$$

$$= \arg\min_{\beta} \sum_{i=1}^{n} \left(y - \hat{y}^{(i)} \right)^2$$

Taking the average across all n training examples, we get:

$$\hat{\beta}_{MSE} = \arg\min_{\beta} \frac{1}{n} \sum_{i=1}^{n} \left(y - \hat{y}^{(i)} \right)^2$$

which is exactly the mean squared error (MSE) cost function

Talk Outline

Binary Logistic Regression

Binary logistic regression is used to model the relationship between a categorical target variable Y and a predictor vector $X = (X_1, X_2, \cdot, X_p)$. The target variable will have two possible values, such as whether a student passes an exam or not, or whether a visitor to a website subscribes to the website's newsletter or not. The two possible categories are coded as '1', called the positive class, and '0', called the negative class. Binary logistic regression estimates the probability that the response variable Y belongs to the positive class given X.

$$p(X) = \Pr(Y = 1 \mid X)$$

Logit Function

The most commonly used link function for binary logistic regression is the logit function (or log-odds2), given as:

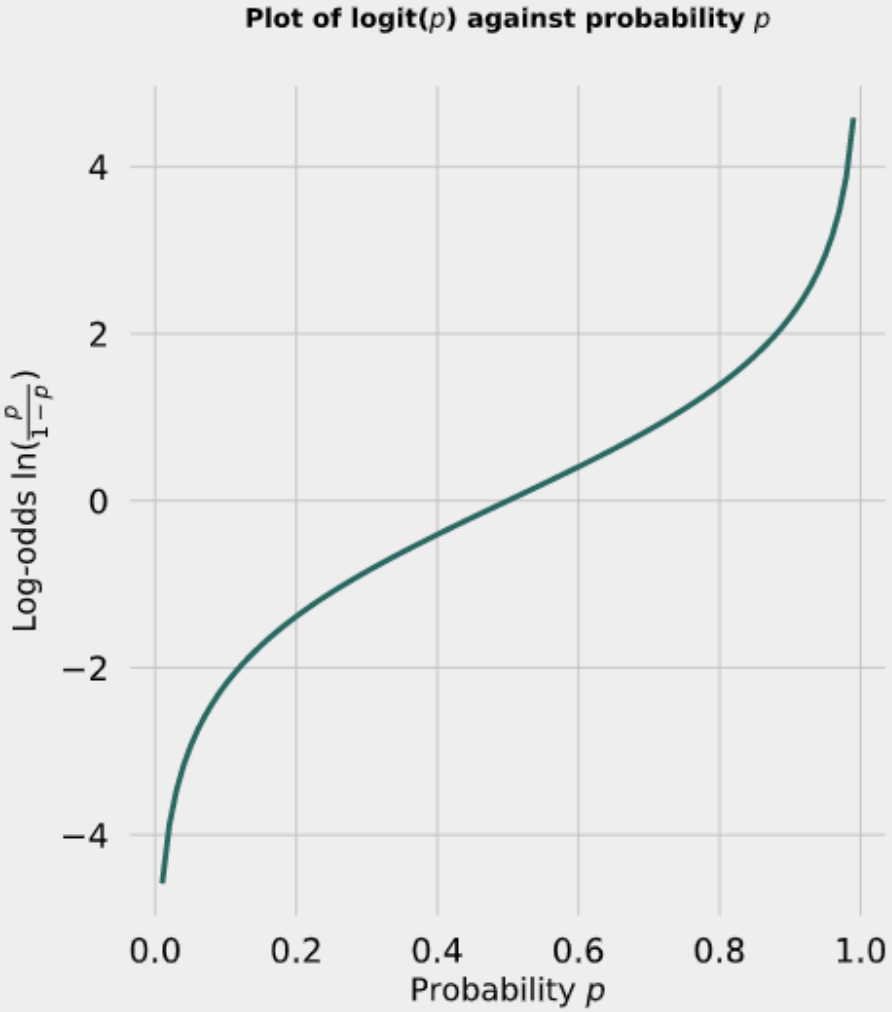
$$\text{logit}(p(X)) = \log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

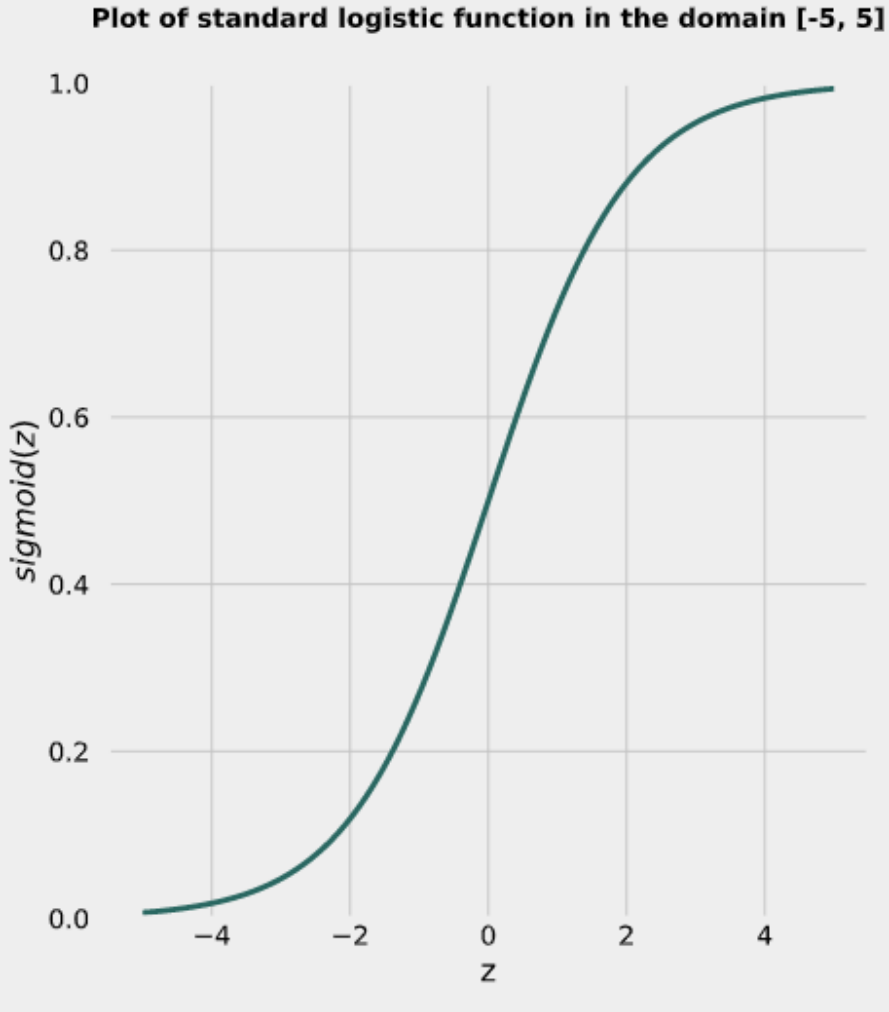
Recall that the inverse function of the natural logarithm function is the exponential function, so if we take the inverse of the equation

$$\frac{p(X)}{1-p(X)} = e^{(\beta_0+\beta_1 X_1+\cdots+\beta_p X_p)}$$

$$p(X) = \frac{e^{(\beta_0+\beta_1 X_1+\cdots+\beta_p X_p)}}{e^{(\beta_0+\beta_1 X_1+\cdots+\beta_p X_p)}+1} = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\cdots+\beta_p X_p)}}$$

Logit Function vs Probability





Deriving Cross Entropy Using MLE

Given a set of n training examples $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(3)})...(x^{(n)}, y^{(n)})$, binary cross-entropy is given by:

$$\text{Cross-Entropy} = -\left[\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)}\log p^{(i)} + \left(1 - y^{(i)}\right)\log\left(1 - p^{(i)}\right)\right)\right]$$

where $x^{(i)}$ is the feature vector, $y^{(i)}$ is the true label (0 or 1) for the i^{th} training example, and $p(i)$ is the predicted probability that the i^{th} training example belongs to the positive class, that is, $Pr(Y = 1 \mid X = x^{(i)})$.

Bernoulli Distribution and PMF

$y^{(i)}$ is a realisation of the Bernoulli random variable Y. The Bernoulli distribution is parameterised by p, and its probability mass function (PMF) is given by:

$$\Pr\left(Y = y^{(i)}\right) = \begin{cases} p & \text{if } y^{(i)} = 1 \\ 1 - p & \text{if } y^{(i)} = 0 \end{cases}$$

Which can also be written as:

$$\Pr\left(Y = y^{(i)}\right) = p^{y^{(i)}}(1-p)^{1-y^{(i)}} \text{ for } y^{(i)} \in \{0, 1\}$$

Likelihood Function

We then define our Likelihood function. The estimates of $\beta_1, \beta_2, \cdots, \beta_p$ will be the ones that maximise the likelihood function.

$$\mathcal{L}\left(p \mid \left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \cdots, \left(x^{(n)}, y^{(n)}\right)\right) = \prod_{i=1}^{n} f\left(y^{(i)} \mid p\right)$$

$$= \prod_{i=1}^{n} p^{y^{(i)}} (1-p)^{1-y^{(i)}}$$

$$\log\left(\mathcal{L}\left(p \mid y^{(1)}, y^{(2)}, \cdots, y^{(n)}\right)\right) = \log\left(\prod_{i=1}^{n} p^{y^{(i)}} (1-p)^{1-y^{(i)}}\right)$$

$$= \sum_{i=1}^{n} \log\left(p^{y^{(i)}} (1-p)^{1-y^{(i)}}\right)$$

$$= \sum_{i=1}^{n} \left(y^{(i)} \log p^{(i)} + \left(1-y^{(i)}\right) \log\left(1-p^{(i)}\right)\right)$$

Cross-Entropy with MLE

The maximum likelihood estimate $\hat{\beta}$ is therefore the value of the parameters that maximises the log-likelihood function.

$$\hat{\beta} = \arg\max_{\beta} \left[\sum_{i=1}^{n} \left(y^{(i)} \log p^{(i)} + \left(1 - y^{(i)}\right) \log \left(1 - p^{(i)}\right) \right) \right]$$

We also know that maximising a function is the same as minimising its negative.

$$\hat{\beta} = \arg\min_{\beta} \left[- \sum_{i=1}^{n} \left(y^{(i)} \log p^{(i)} + \left(1 - y^{(i)}\right) \log \left(1 - p^{(i)}\right) \right) \right]$$

Taking the average across our n training examples, we get:

$$\hat{\beta} = \arg\min_{\beta} \left[-\frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} \log p^{(i)} + \left(1 - y^{(i)}\right) \log \left(1 - p^{(i)}\right) \right) \right]$$

Talk Outline

Bayes Theorem

$$P(\theta \mid y) = \frac{P(y \mid \theta)P(\theta)}{P(y)}$$

Where,
$P(\theta \mid y) \rightarrow$ Posterior
$P(y \mid \theta) \rightarrow$ Likelihood
$P(\theta) \rightarrow$ Prior
$P(y) \rightarrow$ Evidence

Maximum a posteriori

Using MAP (the maximum a posteriori probability estimate), we will find the maximum of the posterior:

$$\hat{\theta}_{MAP} = \arg\max_{\theta} P(\theta \mid y)$$

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \frac{P(y \mid \theta)P(\theta)}{P(y)}$$

$$\hat{\theta}_{MAP} = \arg\max_{\theta} [P(y \mid \theta)P(\theta)]$$

We have removed the evidence because it is a constant with respect to maximization

Joint Distributions

$$P(\theta \mid y) = P(\theta \mid y_1, y_2, \cdots, y_n) = P(\theta \mid y_1)(\theta \mid y_1) \cdots P(\theta \mid y_n)$$

$$P(\theta \mid y) = \prod_{i=1}^{n} P(\theta \mid y_i)$$

It is much better to take the logarithm of the posterior

$$\log P(\theta \mid y) = \log \prod_{i=1}^{n} P(\theta \mid y_i) = \log \sum_{i=1}^{n} P(\theta \mid y_i)$$

Therefore, the MAP estimate becomes:

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \log[P(y \mid \theta)P(\theta)]$$

$$\hat{\theta}_{MAP} = \arg\max_{\theta} [\log P(y \mid \theta) + \log P(\theta)]$$

Zero-Mean Normally Distributed Prior

For linear regression, the errors are normally distributed, which means the outcome variable will also be normally distributed, $y \sim \mathcal{N}(\beta^\top x, \sigma^2)$. The likelihood pdf, which is parameterized by the mean(μ) and variance (σ^2) is given as:

$$P(y \mid \theta) = P(y \mid \beta^\top x, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\beta^\top x)^2}{2\sigma^2}}$$

Now, for each of the coefficient (β_i), we will place a zero-mean normally distributed prior, $y_i \sim \mathcal{N}(0, \delta^2)$. $delta^2$ is the variance:

$$P(\theta) = \frac{1}{\sqrt{2\pi\delta^2}} e^{-\frac{\beta^2}{2\delta^2}}$$

MAP Estimate

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \left[\log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\beta^\top x^{(i)})^2}{2\sigma^2}} + \log \prod_{j=0}^{m} \frac{1}{\sqrt{2\pi\delta^2}} e^{-\frac{\beta_j^2}{2\delta^2}} \right]$$

Added the product sign because we assume that the random variables have a joint probability density function

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \left[\sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\beta^\top x^{(i)})^2}{2\sigma^2}} + \sum_{j=0}^{m} \log \frac{1}{\sqrt{2\pi\delta^2}} e^{-\frac{\beta_j^2}{2\delta^2}} \right]$$

$$= \arg\max_{\theta} \left[n \log \frac{1}{\sqrt{2\pi\sigma^2}} - \sum_{i=1}^{n} \frac{(y-\beta^\top x^{(i)})^2}{2\sigma^2} + m \log \frac{1}{\sqrt{2\pi\delta^2}} - \sum_{j=0}^{m} \frac{\beta_j^2}{2\delta^2} \right]$$

From MAP Estimate to Ridge Regression

$$\hat{\theta}_{MAP} = -\arg\max_{\theta} \frac{1}{2\sigma^2} \left[\sum_{i=1}^{n} \left(y - \beta^\top x^{(i)} \right)^2 + \frac{\sigma^2}{\delta^2} \sum_{j=0}^{m} \beta_j^2 \right]$$

$$\hat{\theta}_{MAP} = \arg\min_{\theta} \left[\sum_{i=1}^{n} \left(y - \hat{y}^{(i)} \right)^2 + \lambda \sum_{j=0}^{m} \beta_j^2 \right]$$

$$\hat{\theta}_{MAP} = \arg\min_{\theta} \left[\text{Mean Square Error} + \text{L2 Regularization} \right]$$

Now, we can see that using a normally distributed prior with MAP leads to the L2 regularization (Ridge Regression)

Talk Outline

Laplacean of Double Exponential Prior

A random variable has a Laplace(μ, b) distribution if the probability density function is:

$$f(x \mid \mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}$$

Therefore, we will place a zero-mean Laplacean distributed prior on the coefficients:

$$P(\theta) = \frac{1}{2b} e^{-\frac{|\beta_j|}{b}}$$

MAP Estimate with the zero-mean Laplacean distributed prior

Using the same process with the MAP

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \left[\log \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\beta^\top x^{(i)})^2}{2\sigma^2}} + \log \prod_{j=0}^{m} \frac{1}{2b} e^{-\frac{|\beta_j|}{b}} \right]$$

$$\hat{\theta}_{MAP} = \arg\max_{\theta} \left[\sum_{i=1}^{n} \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\beta^\top x^{(i)})^2}{2\sigma^2}} + \sum_{j=0}^{m} \log \frac{1}{2b} e^{-\frac{|\beta_j|}{b}} \right]$$

$$= \arg\max_{\theta} \left[n \log \frac{1}{\sqrt{2\pi\sigma^2}} - \sum_{i=1}^{n} \frac{(y - \beta^\top x^{(i)})^2}{2\sigma^2} + \frac{m}{2b} - \sum_{j=0}^{m} \frac{| \beta_j |}{b} \right]$$

From MAP Estimate to LASSO Regression

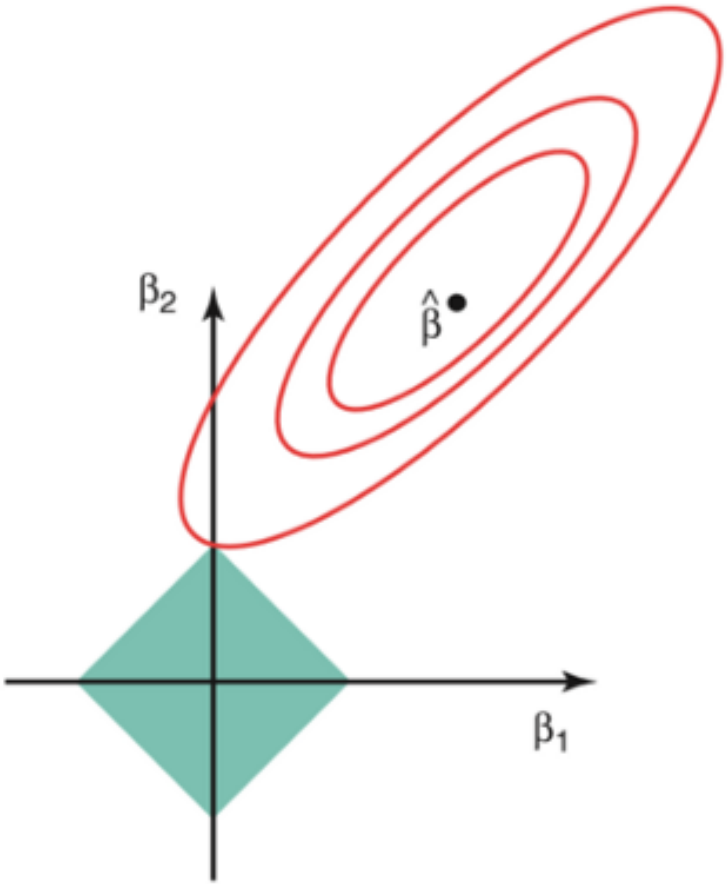
$$\hat{\theta}_{MAP} = -\arg\max_{\theta} \frac{1}{2\sigma^2} \left[\sum_{i=1}^{n} \left(y - \beta^\top x^{(i)} \right)^2 + \frac{2\sigma^2}{b} \sum_{j=0}^{m} \mid \beta_j \mid \right]$$

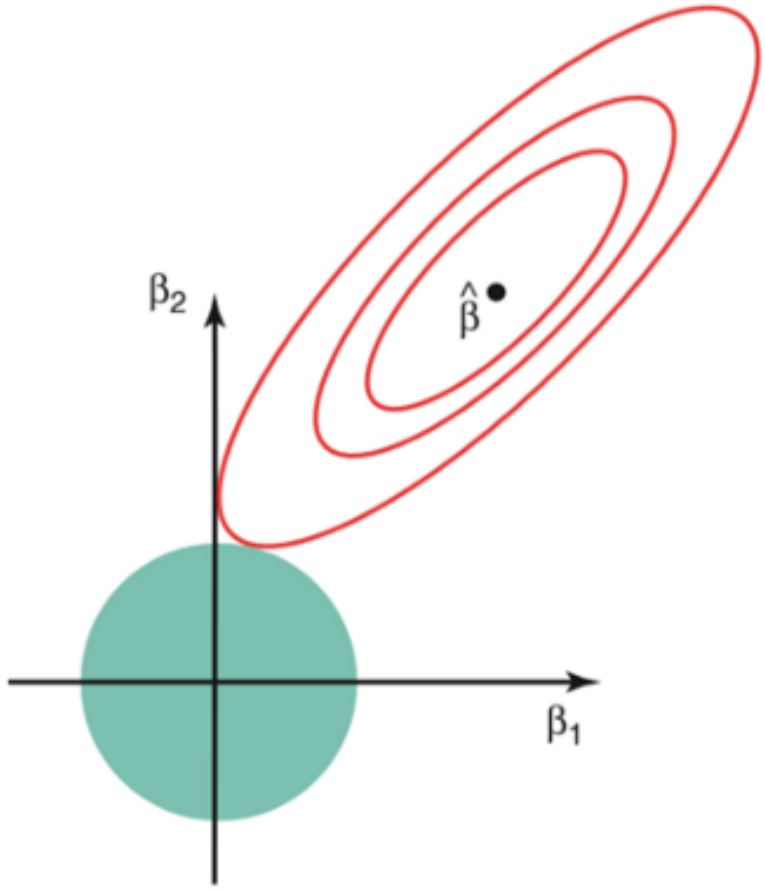
$$\hat{\theta}_{MAP} = \arg\min_{\theta} \left[\sum_{i=1}^{n} \left(y - \hat{y}^{(i)} \right)^2 + \lambda \sum_{j=0}^{m} \mid \beta_j \mid \right]$$

$$\hat{\theta}_{MAP} = \arg\min_{\theta} \left[\text{Mean Square Error} + \text{L1 Regularization} \right]$$

Now, we can see that using a normally distributed prior with MAP leads to the L1 regularization (LASSO Regression)

Ridge vs LASSO





Mathematics for Machine Learning - Module 3

Open Data Science Conference (ODSC West 2020)

Adewale (Wale) Akinfaderin
{@waleakinfaderin}

October 27, 2020