Imperial College
London

NOTES

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Machine Learning: Logistic Regression

Author:
W (CID: your college-id number)

Date: January 6, 2020

1 Introduction

Machine Learning can be categorized into two main categories:

1. Supervised Learning: Given input x and output y, the algorithm learns the mapping function $f(x) = y$.

2. Unsupervised Learning: Given only input x, the algorithm learns the underlying structure.

Supervised Learning can be categorized into two main categories:

1. Classification: Predicting a category or a label

2. Regression: Predicting a real value number or a vector

Logistic Regression is a machine learning algorithm for classification problems.

2 Logistic Regression

Logistic regression assumes the data can be separated by a line or plane. Therefore it is considered as a linear classifier.

2.1 Linear Classification

Given a set of input X that belongs to class 0 or class 1. Linear classification is a classification algorithm that makes classification based on a linear predictor function using the features of x.

In the 2 dimensional case, each input x has two features x_1 and x_2.

The general form of a line is

$$ax + by + c = 0$$

Notice that the points $\{(x, y) \mid ax + by + c < 0\}$ are on one side of the line, and the points $\{(x, y) \mid ax + by + c > 0\}$ are on the other side of the line.

We can define a linear predictor function by combining a set of weights w with the features of x, and use it to determine the class y that x belongs to.

$$h(x) = w_0 + w_1 x_1 + w_2 x_2 = w^T x$$

$$y = \begin{cases} 1, & h(x) > 0 \\ 0, & h(x) < 0 \\ unknown, & h(x) = 0 \end{cases}$$

In higher dimensions, $h(x)$ can be a plane or hyper-plane.

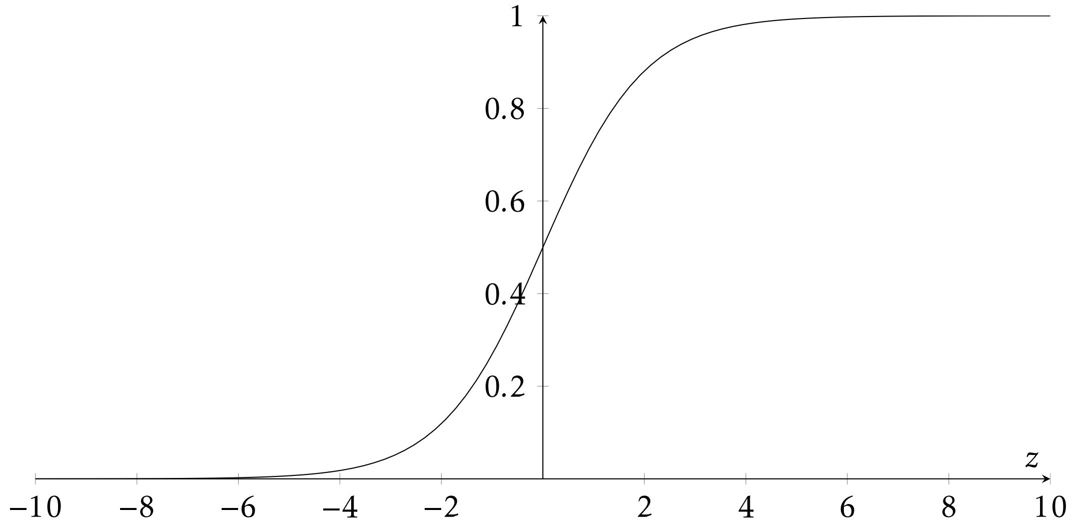
2.2 Logistic Function

In *Linear Classification*, we use a linear predictor function to predict the class of x.

$$h(x) = w^T x$$

Applying a regression function to $h(x)$ gives the probability of x belonging to a certain class. A common regression function to use is the **sigmoid function**

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



Applying the sigmoid function to $h(x)$ gives the probability of x belonging to class 1

$$\sigma(h(x)) = \sigma(w^T x) = P(y = 1 \mid x)$$

Since in binary classification class y can only be 0 or 1,

$$P(y = 0 \mid x) = 1 - P(y = 1 \mid x)$$

When $h(x) = 0$, $\sigma(h(x)) = 0.5$ meaning the probability that x belongs to each class is 50%. The more positive $h(x)$ is, the higher the probability x belongs to class 1. The more negative $h(x)$ is, the lower the probability x belongs to class 1 so x belongs to class 0.

3 Solving for the Optimal Weights

Given a set of input X that belongs to class $y \in \{0, 1\}$. The loss function to minimize in logistic regression is the **cross entropy function**.

$$J = -\sum_{i=1}^{N} (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

where $\hat{y}_i$ is the output of logistic regression (A value between 0 and 1).

Notice that only one term matters for each prediction as y_i is either 0 or 1.

3.1 Gradient Descent

Gradient descent is a optimization method for finding local minimum or minimum in a function.

Instead of solving for w directly, we set w to some random value and take step to update w in the negative of the gradient in each iteration. (α is the learning rate)

$$w \leftarrow w - \alpha \frac{\partial J}{\partial w}$$

Since

$$\hat{y}_i = \frac{1}{1+e^{-z_i}} \qquad z_i = w^T x_i$$

We can solve for the derivative using Chain rule

$$\begin{aligned}
\frac{\partial J}{\partial w_j} &= \sum_{i=1}^{N} \frac{\partial J}{\partial \hat{y}_i} \frac{\partial y_i}{\partial z_i} \frac{\partial z_i}{\partial w_j} \\
&= \sum_{i=1}^{N} -\left(\frac{y_i}{\hat{y}_i} - \frac{1-y_i}{1-\hat{y}_i}\right)\left(\frac{e^{-z_i}}{(1+e^{-z_i})^2}\right) x_{ij} \\
&= \sum_{i=1}^{N} -\left(\frac{y_i}{\hat{y}_i} - \frac{1-y_i}{1-\hat{y}_i}\right)\left(\frac{1}{1+e^{-z_i}}\right)\left(\frac{e^{-z_i}}{1+e^{-z_i}}\right) x_{ij} \\
&= \sum_{i=1}^{N} -\left(\frac{y_i}{\hat{y}_i} - \frac{1-y_i}{1-\hat{y}_i}\right) \hat{y}_i (1-\hat{y}_i) x_{ij} \\
&= \sum_{i=1}^{N} -\big(y_i (1-\hat{y}_i) x_{ij} - (1-y_i)\hat{y}_i x_{ij}\big) \\
&= \sum_{i=1}^{N} (\hat{y}_i - y_i) x_{ij} \\
\frac{\partial J}{\partial w} &= \sum_{i=1}^{N} (\hat{y}_i - y_i) x_i = X^T (\hat{Y} - Y)
\end{aligned}$$

Hence the weight can be updated in each step by

$$w \leftarrow w - \alpha X^T (\hat{Y} - Y)$$

3.2 Interpreting the Weights

From *Logistic Function*,

$$P(y=0 \mid x) = 1 - P(y=1 \mid x)$$

In probability, **odds** is the ratio between the probability of success and the probability of failure.

$$\begin{aligned}
\frac{P(E)}{P(\sim E)} &= \frac{P(y=1\mid x)}{P(y=0\mid x)} \\
&= \frac{1}{1+e^{-(w^Tx)}} \times \frac{1+e^{-(w^Tx)}}{e^{-(w^Tx)}} = e^{(w^Tx)} \\
\log\left(\frac{P(E)}{P(\sim E)}\right) &= w^Tx = w_0x_0 + w_1x_1 + \ldots + w_Dx_D
\end{aligned}$$

Therefore w_i is amount that **log odds** will increase if x_i increased by 1.

3.3 Linear Discriminant Analysis (LDA)

In general we solve for the weights of a regression model using gradient descent. However, we can use linear discriminant analysis to find a closed-form solution if the data is distributed in a specific way.

In binary classification, each x belongs to a class $y \in \{0,1\}$, where each class is Gaussian distributed. These classes have a common covariance matrix Σ but different mean vectors μ_k.

From *Logistic Function*,

$$p(y=1\mid x) = \sigma(w^Tx+b) = \frac{1}{1+e^{-(w^Tx+b)}}$$

From Bayes Theorem,

$$\begin{aligned}
P(y=1\mid x) &= \frac{P(x\mid y=1)P(y=1)}{P(x)} \\
&= \frac{1}{1+\frac{P(x)-P(x\mid y=1)P(y=1)}{P(x\mid y=1)P(y=1)}} \\
&= \frac{1}{1+\frac{P(x\mid y=0)P(y=0)}{P(x\mid y=1)P(y=1)}}
\end{aligned}
\qquad P(x) = \sum_{k=0,1} P(x\mid y=k)P(y=k)$$

Equating the two formulas

$$\begin{aligned}
\frac{1}{1+e^{-(w^Tx+b)}} &= \frac{1}{1+\frac{P(x\mid y=0)P(y=0)}{P(x\mid y=1)P(y=1)}} \\
e^{-(w^Tx+b)} &= \frac{P(x\mid y=0)P(y=0)}{P(x\mid y=1)P(y=1)} \\
-(w^Tx+b) &= \ln\left(\frac{P(x\mid y=0)P(y=0)}{P(x\mid y=1)P(y=1)}\right) = \ln(P(x\mid y=0)) - \ln(P(x\mid y=1)) + \ln\left(\frac{P(y=0)}{P(y=1)}\right)
\end{aligned}$$

$$\begin{aligned}
w^T x + b &= \ln(P(x \mid y=1)) - \ln(P(x \mid y=0)) - \ln\left(\frac{\pi_0}{\pi_1}\right) \\
&= \ln\left(\frac{1}{\sqrt{(2\pi)^D|\Sigma|}} e^{-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)}\right) - \ln\left(\frac{1}{\sqrt{(2\pi)^D|\Sigma|}} e^{-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)}\right) - \ln\left(\frac{\pi_0}{\pi_1}\right) \\
&= -\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) + \frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0) - \ln\left(\frac{\pi_0}{\pi_1}\right) \\
&= x^T\Sigma^{-1}\mu_1 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 - x^T\Sigma^{-1}\mu_0 + \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 - \ln\left(\frac{\pi_0}{\pi_1}\right) \\
&= (\mu_1-\mu_0)\Sigma^{-1}x - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 - \ln\left(\frac{\pi_0}{\pi_1}\right) \\
\therefore w &= (\mu_1-\mu_0)\Sigma^{-1} \\
\therefore b &= \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0 - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 - \ln\left(\frac{\pi_0}{\pi_1}\right)
\end{aligned}$$

Keep in mind the solution is optimal only when the assumptions about the distributions are true. Since this is not always the case, using gradient descent is generally more applicable.

4 Regularization

In machine learning, regularization is the process of adding information to solve ill-posed problems and prevent over-fitting.

4.1 L1 Regularization

Sometimes we only want to select a few useful feature to predict the trend. L1 Regularization achieves **sparsity**, meaning the final weight will contains mainly zeros with a few non-zeros.

We simply add a penalty term to the original cost. So E becomes

$$\begin{aligned}
E &= \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \lambda|w| \\
\therefore \frac{\partial E}{\partial w} &= X^T(\hat{Y} - Y) + \lambda sign(w) \\
sign(w) &= \begin{cases} -1, w < 0 \\ 0, w = 0 \\ 1, w > 0 \end{cases}
\end{aligned}$$

4.2 L2 Regularization

Logistic regression uses a logistic function to predict the probability of a class that input x belongs to

$$\hat{y} = \frac{1}{1 + e^{-(w^T x)}}$$

To minimize the loss function J:

- If $y_i = 0$, then $\hat{y}_i = 1$ will yield to minimum loss since $\log(1 - \hat{y}_i) = 0$. The input into the sigmoid function must be infinity, so w approaches infinity.
- If $y_i = 1$, then $\hat{y}_i = 1$ will yield to minimum loss since $\log(\hat{y}_i) = 0$. The input into the sigmoid function must be infinity, so w approaches infinity.

The optimal solution is when w reaches infinity. One solution to prevent weight going to infinity is to add the squared magnitude of the weights multiplied by a constant to the error function to punish large weights. So J becomes

$$J = -\sum_{i=1}^{N} (y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) + \lambda |w|^2$$

The squared magnitude $|w|^2$ is $\sum w_i^2$, which is the dot product $w^T w$.

In *Gradient Descent* we saw that

$$\frac{\partial}{\partial w} - \sum_{i=1}^{N} (y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) = X^T(\hat{Y} - Y)$$

$$\therefore \frac{\partial E}{\partial w} = X^T(\hat{Y} - Y) + \lambda w$$

4.3 L1 Regularization vs L2 Regularization

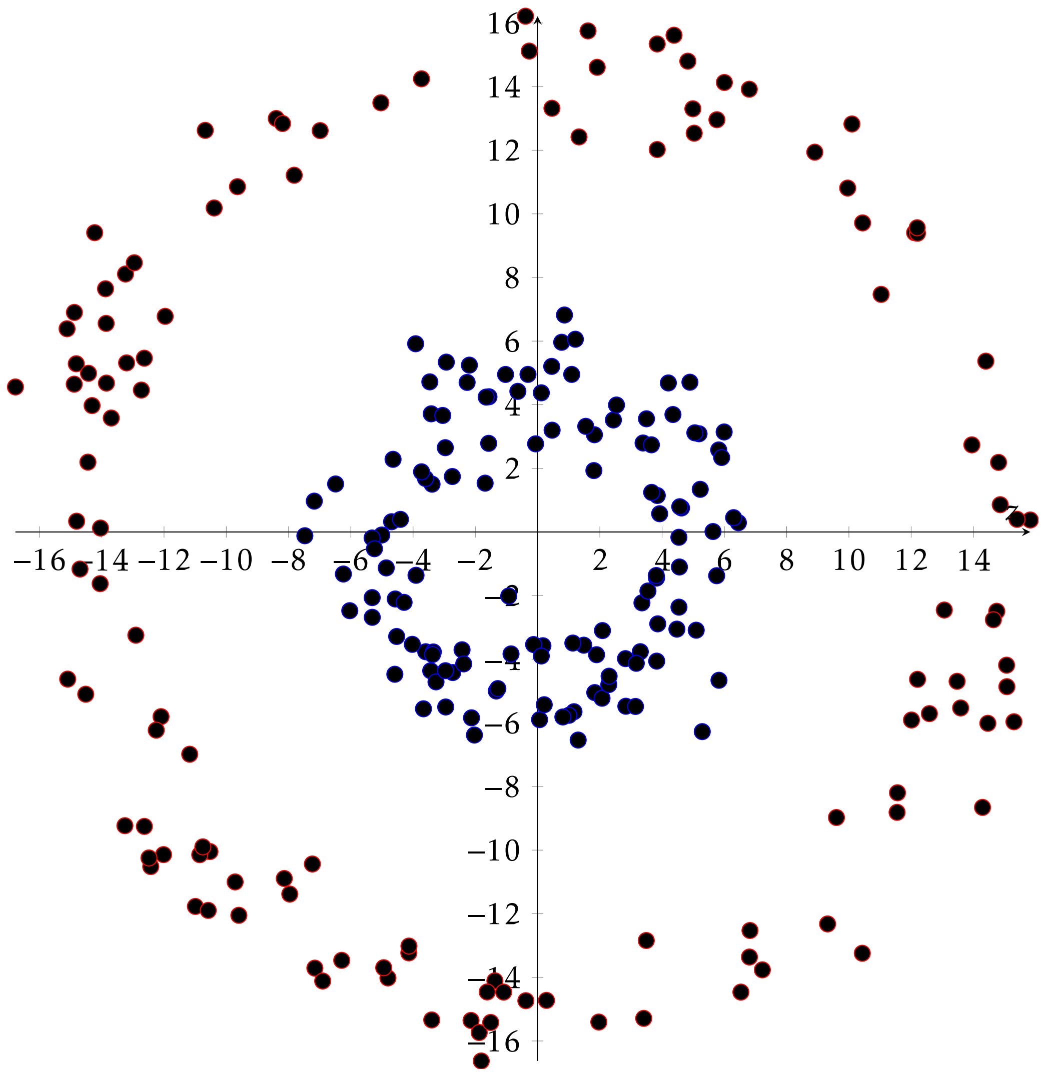
L1 Regularization encourages sparse weights whereas L2 Regularization encourages small weights.

Both methods prevents over-fitting. L1 Regularization accomplishes this by only choosing important features; L2 Regularization accomplishes this by preventing one of the weights to go extremely large.

5 Problems with Logistic Regression

5.1 The Donut Problem

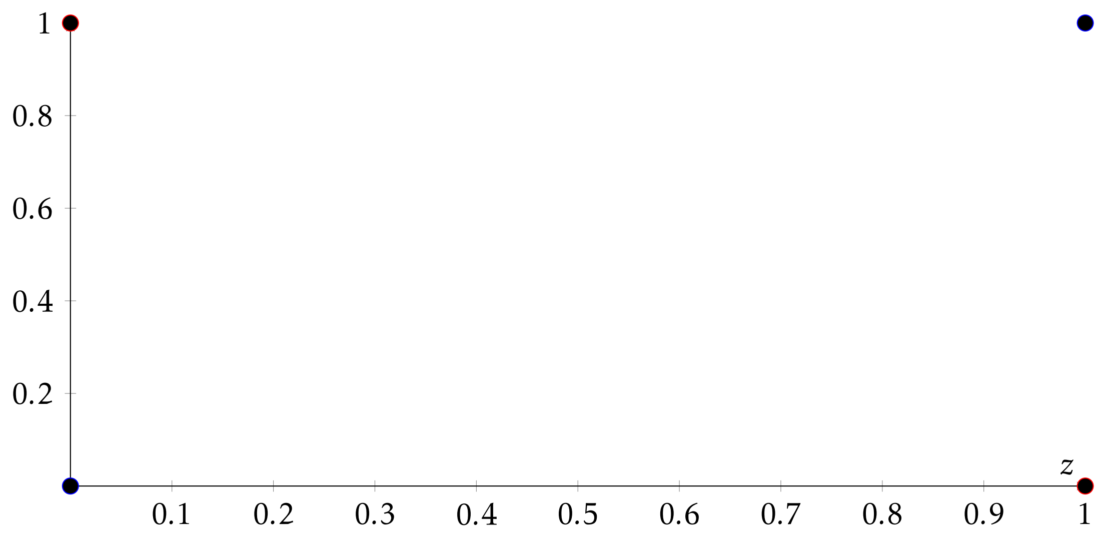
Logistic regression is a linear classifier and can only classify data that can be separated by a line or a plane. The problem arise when data points are clustered in a donut shape of different radius, hence the donut problem.



To address the problem, include the **distance of the dot from origin** as another feature of input *x*. The prediction will depend mainly on the added feature.

5.2 The XOR Problem

XOR is a logical operator meaning exclusive OR. The output is 1 is one if only one of the features is 1, the output is 0 otherwise.



From the scatter chart, it is clear that the data cannot be separated by any linear classifier. To address the problem, include the **product of x_1 and x_2** as another feature to input x. This will make the inputs separable using a linear plane.

5.3 Neural Networks

Manually adding features is called **feature engineering**. However, manually doing feature engineering is tedious work.

In machine learning, a machine should be able to learn from patterns without needing us to explicitly add features to the input. This problem can be solved using **neural networks**.

6 A Deeper Look into Cross Entropy

In logistic regression, minimizing the cross entropy function is the same as maximizing the likelihood function of a Bernoulli distribution.

The likelihood function is the joint probability distribution of the random samples. Since all samples are independent, the likelihood is the product of individual probabilities.

If the sample belongs to class 0, we find $P(y = 0 \mid x)$.

If the sample belongs to class 1, we find $P(y = 1 \mid x)$.

$$\begin{aligned}\operatorname{argmax} L &= \operatorname{argmax}\left(\prod_{i=1}^{N} P(y = 1 \mid x_i)^{y_i} P(y = 0 \mid x_i)^{1-y_i}\right)\\ &= \operatorname{argmax}\left(\prod_{i=1}^{N} \hat{y}_i{}^{y_i}(1-\hat{y}_i)^{1-y_i}\right)\end{aligned}$$

$$= \operatorname{argmax}\left(\log\left(\prod_{i=1}^{N} \hat{y}_i{}^{y_i}(1-\hat{y}_i)^{1-y_i}\right)\right)$$
$$= \operatorname{argmax}\left(\sum_{i=1}^{N}(y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i))\right)$$

Maximizing an error function $-E$ is equivalent of minimizing E.

$$\operatorname{argmax}\left(\sum_{i=1}^{N}(y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i))\right) = \operatorname{argmin}\left(-\sum_{i=1}^{N}(y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i))\right)$$

Which is minimizing the cross entropy function.

7 A Deeper Look into Linear Discriminant Analysis

Given data from k classes where each class is Gaussian distributed. Assuming these classes have a common covariance matrix Σ but different mean vectors μ_k.

LDA focuses on the posterior probability $P(y = k \mid x)$. From Bayes theorem:

$$P(y=k\mid x) = \frac{P(x\mid y=k)P(y=k)}{P(x)}$$

We model $P(x \mid Y = k)$ as a multivariate normal distribution

$$P(x\mid y=k) = f_k(x) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}}e^{-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)}$$

$P(Y = k)$ is estimated by the fraction of training samples of class k

$$P(y=k) = \pi_k$$

In LDA the goal is to find k that maximize the posterior probability

$$\begin{aligned}
\operatorname*{argmax}_k P(y=k\mid x) &= \operatorname*{argmax}_k\left(\frac{P(x\mid y=k)P(y=k)}{P(x)}\right)\\
&= \operatorname*{argmax}_k(P(x\mid y=k)P(y=k))\\
&= \operatorname*{argmax}_k(f_k(x)\pi_k)\\
&= \operatorname*{argmax}_k(\log(f_k(x)\pi_k))\\
&= \operatorname*{argmax}_k(\log f_k(x)+\log\pi_k)\\
&= \operatorname*{argmax}_k\left(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)+\log\pi_k\right)
\end{aligned}$$

$$\begin{aligned}
&= \underset{k}{\operatorname{argmax}} \left(-\frac{1}{2} \left(x^T \Sigma^{-1} x - 2x^T \Sigma^{-1} \mu_k + \mu_k^T \Sigma^{-1} \mu_k \right) + \log \pi_k \right) \\
&= \underset{k}{\operatorname{argmax}} \left(x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k \right)
\end{aligned}$$

Notice that this is a linear equation in x.

Finally we predict x belongs to class k with the highest posterior probability.

8 A Deeper Look into Regularization

In **Bayes Theorem**, given θ is the parameters we are estimating, and y is the observed data.

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

$P(\theta|y)$ is the **posterior**. $P(y|\theta)$ is the **likelihood**.

$P(\theta)$ is the **prior**. $P(y)$ is the **evident**.

Classical statistics focuses on maximizing likelihood function. However in a probabilistic point of view, we would want to **maximize a posterior probability (MAP)**.

We can regularize the parameters θ by adding prior knowledge about the expected distribution of θ. This prior is something we explicitly choose that is not based on the data.

In linear regression, the goal is to find the w that maximizes the posterior. We can ignore $P(y)$ because it is constant with respect to maximization. We can take log of the inner function since log is monotonically increasing.

$$\begin{aligned}
\underset{w}{\operatorname{argmax}} P(w|y) &= \underset{w}{\operatorname{argmax}} \left(\frac{P(y|w)P(w)}{P(y)} \right) \\
&= \underset{w}{\operatorname{argmax}} \left(P(y|w)P(w) \right) \\
&= \underset{w}{\operatorname{argmax}} \left(\log \left(P(y|w)P(w) \right) \right) \\
&= \underset{w}{\operatorname{argmax}} \left(\log P(y|w) + \log P(w) \right)
\end{aligned}$$

8.1 L1 Regularization

L1 Regularization is equivalent to having a Laplacean Prior of 0 mean. The PDF of the Laplace distribution is

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

We know how to find the likelihood $P(y|w)$ from *A Deeper Look into Cross Entropy*, and the prior $P(w)$ is the joint probability of each weight component from $Laplace(0,b)$

$$\begin{aligned}
\underset{w}{\operatorname{argmax}}(P(w|y)) &= \underset{w}{\operatorname{argmax}}\left(\log\prod_{i=1}^{N}\hat{y}_i^{\,y_i}(1-\hat{y}_i)^{1-y_i} + \log\prod_{j=1}^{D}\frac{1}{2b}e^{-\frac{|w_j|}{b}}\right)\\
&= \underset{w}{\operatorname{argmax}}\left(\sum_{i=1}^{N}(y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i)) + \sum_{j=1}^{D}\log\left(\frac{1}{2b}e^{-\frac{|w_j|}{b}}\right)\right)\\
&= \underset{w}{\operatorname{argmax}}\left(\sum_{i=1}^{N}(y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i)) - \frac{1}{b}\sum_{j=1}^{D}|w_j|\right)\\
&= \underset{w}{\operatorname{argmax}}\left(\sum_{i=1}^{N}(y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i)) - \lambda\sum_{j=1}^{D}|w_j|\right)\\
&= \underset{w}{\operatorname{argmin}}\left(-\sum_{i=1}^{N}(y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i)) + \lambda\sum_{j=1}^{D}|w_j|\right)
\end{aligned}$$

This gives us the formula for L1 Regularization.

8.2 L2 Regularization

L2 Regularization is equivalent to having a Gaussian Prior. We add a prior knowledge that each component in w is normally distributed $N(0,\tau^2)$.

We know how to find the likelihood $P(y|w)$ from *A Deeper Look into Cross Entropy*, and the prior $P(w)$ is the joint probability of each weight component from $N(0,\tau^2)$

$$\begin{aligned}
\underset{w}{\operatorname{argmax}}(P(w|y)) &= \underset{w}{\operatorname{argmax}}\left(\log\prod_{i=1}^{N}\hat{y}_i^{\,y_i}(1-\hat{y}_i)^{1-y_i} + \log\prod_{j=1}^{D}\frac{1}{\sqrt{2\pi\tau^2}}e^{-\frac{w_j^2}{2\tau^2}}\right)\\
&= \underset{w}{\operatorname{argmax}}\left(\sum_{i=1}^{N}(y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i)) + \sum_{j=1}^{D}\log\left(\frac{1}{\sqrt{2\pi\tau^2}}e^{-\frac{w_j^2}{2\tau^2}}\right)\right)\\
&= \underset{w}{\operatorname{argmax}}\left(\sum_{i=1}^{N}(y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i)) - \frac{1}{2\tau^2}\sum_{j=1}^{D}w_j^2\right)\\
&= \underset{w}{\operatorname{argmax}}\left(\sum_{i=1}^{N}(y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i)) - \lambda\sum_{j=1}^{D}w_j^2\right)\\
&= \underset{w}{\operatorname{argmin}}\left(-\sum_{i=1}^{N}(y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i)) + \lambda\sum_{j=1}^{D}w_j^2\right)
\end{aligned}$$

This gives us the formula for L2 Regularization.

9 Datasets

This is a link to datasets with facial expression for logistic regression.

https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-rec
data