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

Lecture 7: Introduction to Classification

Lecturer: Swaprava Nath *SG13, SG14*

7.1 Regularisation

As discussed in previous Lecture,2 types of Regularizers are commonly used in Regression.

- **Ridge regression** :

$$\underset{w}{\operatorname{argmin}} \left\{ \left\| \Phi w - y \right\|^2 + \lambda \left\| w \right\|_2^2 \right\}$$

Where λ is regularisation constant and $\left\|w\right\|_2$ is L_2 norm given by

$$\|\mathbf{w}\|_2 = \sqrt{w_1^2 + w_2^2 + \ldots + w_n^2}$$

Ridge regression mitigates overfitting by penalizing the magnitude of coefficients, controlled by a hyperparameter λ. The L_2 norm is chosen as the regularization term and below is an equivalent optimization problem.

Equivalent optimization Problem : $\min ||\Phi w - y||^2$ such that $||w||_2 \leq c_1$(constraint).

- **LASSO regression** :

$$\underset{w}{\operatorname{argmin}} \left\{ \left\| \Phi w - y \right\|^2 + \lambda \left\| w \right\|_1 \right\}$$

Where λ is regularisation constant and $\left\|w\right\|_1$ is L_1 norm given by

$$\|\mathbf{w}\|_1 = \sum_{i=1}^{n} |w_i|$$

LASSO regression mitigates overfitting by penalizing the magnitude of coefficients, controlled by a hyperparameter λ.The L_1 norm is chosen as the regularization term and below is an equivalent optimising problem.

Equivalent optimization Problem : $min||\Phi w - y||^2$ such that $||w||_1 \leq c_2$(constraint).

where $||w||_1 = \sum_{i=1}^{d} |w_i|$, $||w||_2 = \sqrt{\sum_{i=1}^{d} |w_i|^2}$, c_1 and c_2 are dependent on hyper parameter λ.

7.1.1 Geometric Interpretation of constraints

Consider the case of $d = 2$, i.e., $w = (w_1, w_2)$. Let the optimal solution of the unconstrained least squares error function $||\Phi w - y||^2$ be $\hat{\beta}$. The feasible regions for Ridge and LASSO are given by the blue regions in Figure 7.1.

- **Case 1:** $\hat{\beta}$ is already present in the feasible region, then $\hat{\beta}$ will also be the solution of the constrained optimization problem.
- **Case 2:** if $\hat{\beta}$ is outside the region, we need to draw contour diagrams of the error function in the plane of w.
  - **Ridge:** Our constraint represents the area bounded by a circle (Figure 7.1), the optimal solution will be the point where the contour touches the circle.
  - **LASSO:** Our constraint represents the area bounded by a square (Figure 7.1), the optimal solution will be the point where the contour touches the square.

From Figure 7.1 it is intuitive to see that for LASSO Regression, the optimal *feasible* solution usually tends to be concentrated around one of the corners. However, for Ridge Regression, there is no such bias towards any of the corners. Extending this phenomenon in higher dimensions, we see that LASSO always yields sparser solutions than Ridge (i.e., more components are zeros).

LASSO Regression is generally useful when some of the components of the parameter vector are more important compared to others and Ridge Regression is used when all components are important.

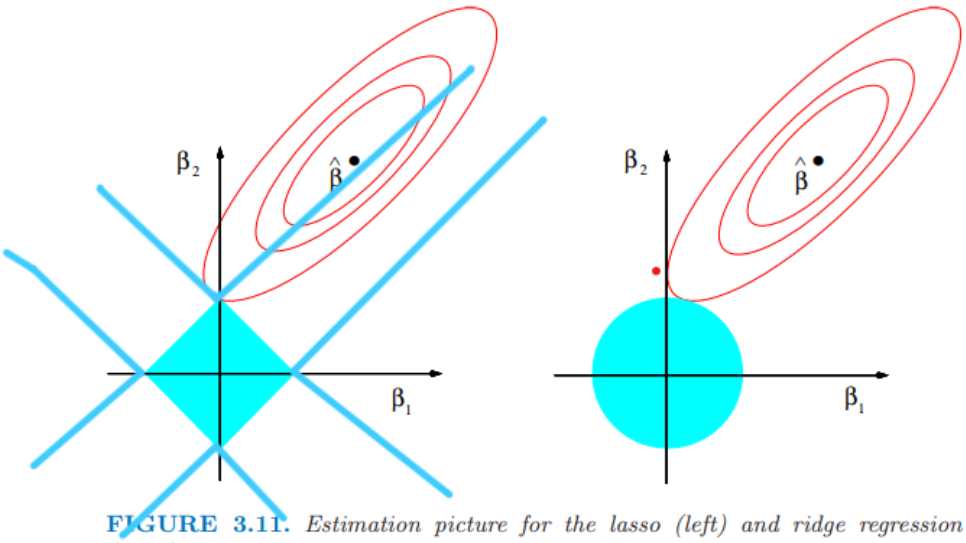

FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are the contours of the least squares error function.*

Figure 7.1: Contour diagrams for Ridge and LASSO regression.

7.1.2 MAP Estimate

- If we use the Guassian prior for the parameter vector w to obtain the MAP estimate, we get ridge regression formula. The derivation is available in the previous lecture notes.

- If we use the Laplacian prior for the parameter vector w to obtain the MAP estimate, we get the LASSO regression formula.

$$Laplace(w_i|\mu, b) = \frac{1}{2b} \exp\left(-\frac{|w_i - \mu|}{b}\right)$$

$$Guassian(w_i|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(w_i - \mu)^2}{2\sigma^2}\right)$$

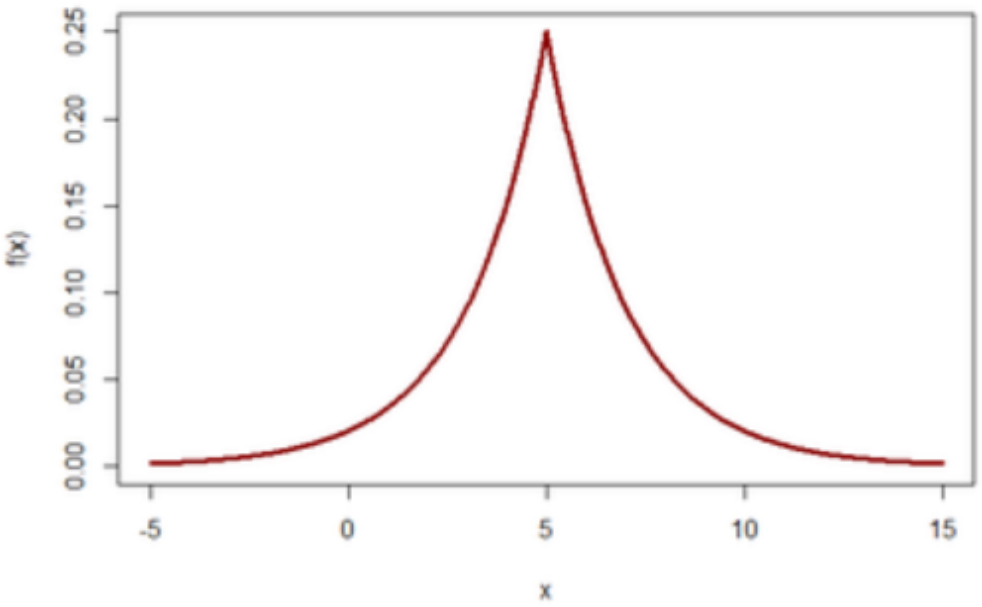

Figure 7.2: Graph of Laplace function

7.2 Classification

In classification, the dimensions of data points are the same as in regression, $x \in \mathbb{R}^d$ and $y \in S$ where S is a *finite* set. If $|S| = 2$, then it is known as *binary classification*. To begin with, we will use a probabilistic approach for classification.

7.2.1 Probabilistic approach

The value of y that maximizes the following probability for a given data point $\hat{x}$ is decided to be the class of $\hat{x}$.

$$\hat{y} = \underset{y}{\text{argmax}}\ P(Y = y|\hat{x}), \text{ where } \hat{x} = (\hat{x_1}, \hat{x_2}, \ldots, \hat{x_d})^T$$

Bayes rule allows us to write the above probability as:

$$P(Y = y|\hat{x}) = \frac{P(\hat{x}|Y = y)P(Y = y)}{P(\hat{x})}.$$

7.3 Naïve Bayes Classifier

In Naïve Bayes Classifier, it makes a naïve assumption which might not be always true. The assumption is that all components of an input data point $\mathbf{x}$ are *conditionally* independent given the class of the data. That is, if $\mathbf{x} = (x_1, x_2, ..., x_d)^T$, then $x_1, x_2, \ldots, x_d$ are conditionally independent given $Y = y$.

Using the conditional independence, we get,

$$P(x|Y=y) = \prod_{i=1}^{d} P(x_i|Y=y)$$

$$\implies \operatorname*{argmax}_y P(Y=y|x) = \operatorname*{argmax}_y \prod_{i=1}^{d} P(x_i|Y=y)P(Y=y)$$

$$= \operatorname*{argmax}_y \left(\sum_{i=1}^{d} \log P(x_i|Y=y) + \log P(Y=y) \right).$$

Here the likelihood and prior can be estimated from the training data as follows.

$$P(x_i|Y=y) = \frac{\text{no. of times } x_i \text{ and } y \text{ happen together}}{\text{no. of times } Y=y \text{ happens}}$$

$$P(Y=y) = \frac{\text{no. of times Y = y}}{n}, \text{where } n \text{ is size of the training data.}$$

7.3.1 Advantages of NB classifier

Take a simple case where each component of x can only take binary values. The conditional distribution $P(x|Y=y)$ will then require $(2^d - 1)$ parameters since the state-space of x is 2^d. Similarly, we need $(k-1)$ parameters to represent $P(Y=y)$. Hence, the total number of parameters to represent $P(Y=y|x)$ is $(2^d - 1)(k-1)$ parameters without the NB asssumption.

With NB assumption, we need parameters to represent $P(x_i = 1|Y=y)$ for all $i = 1, \ldots, d$, for which we require 1 parameter each. In total, we require a total of $d \times 1 \times (k-1)$ parameters.

Hence, NB classifier reduces the number of parameters from exponential to a linear number.

7.3.2 An Application in NLP : Topic classification

Naïve Bayes classification is used in *natural language processing* to identify topic of a given document. It has training data on previous documents and their topics. The method treats the documents as a "bag of words", and creates the set of unique words and put arbitrary indices to them. This method is known as *tokenization*.

Given an article x, we can now treat its N words as $x_1, x_2, \ldots, x_N$ where x_i is the i^{th} word. To maximize $P(Y=y|x)$. We use the equation obtained above via Naïve Bayes assumption.

$$\operatorname*{argmax}_y P(Y=y|x) = \operatorname*{argmax}_y \left(\prod_{i=1}^{d} P(x_i|Y=y)\right) P(Y=y)$$

$$P(x|Y=y) = \prod_{i=1}^{N} P(x_i|Y=y)$$

The individual probabilities of a word appearing in a topic, $P(x_i|Y = y)$, can be estimated from the training dataset (text) using the approach below.

$$P(x_i|Y = y) = \frac{\text{no. of times } x_i \text{ appear in the document of class } y \ +1}{\sum_{w \in V} \text{no. of times } w \text{ appear in class } y \ + |V|}$$

$w \in V$ means for every word w in vocabulary V.

In above equation we added 1 to numerator because otherwise when a new word shows up in test data, its probability estimate becomes zero and that makes the entire likelihood to go to zero. But to normalize that we need to add $|V|$ to the denominator. This makes the likelihood of such a word in the topic small, but not zero. Finally,

$$P(Y = y) = \text{Relative fraction of y in all topics}$$

In this way, we can find out topic name for a given document. However, the assumption that each word in the given document and of a given topic is conditionally independent is not true in general. For instance, words like 'in', 'front', 'of' are highly correlated. To make the NB model more realistic, sometimes phrases (group of words) are used as x_i's. A phrase of a fixed length n is called an *n-gram*.

7.3.3 Disadvantage of NB Classifier

We have seen that NB classifier has an advantage in computation but it also has disadvantages. Here is an example to explain that.

Consider two binary random variables X_1 and Y whose true distributions are given by the tables below. Assume that the estimated prior and conditional probabilities are also the same the true values. According

Y=0	Y=1
0.8	0.2

Table 7.1: Prior $P(Y)$

	$X_1 = 0$	$X_1 = 1$
Y=0	0.7	0.3
Y=1	0.3	0.7

Table 7.2: Conditional probability $P(X_1|Y)$

to NB classifier the correct class of X_1 for both its realizations can be found by $\hat{y} = \underset{y}{\text{argmax}}\ P(Y = y|X_1) = \underset{y}{\text{argmax}}\ P(X_1|Y = y)P(Y = y)$

	$P(X_1, Y = 0)$	$P(X_1, Y = 1)$	$\hat{y}$
$X_1 = 0$	0.7×0.8	0.3×0.2	0
$X_1 = 1$	0.3×0.8	0.7×0.2	0

Table 7.3: $P(Y = y|X_1)$

$$\begin{aligned}
\text{Expected error} &= P(Y \neq \hat{y}) \\
&= P(Y \neq \hat{y}, X_1 = 0) + P(Y \neq \hat{y}, X_1 = 1) \\
&= P(Y \neq 1, X_1 = 0) + P(Y \neq 0, X_1 = 1) \\
&= P(Y = 0, X_1 = 0) + P(Y = 1, X_1 = 1) \\
&= 0.06 + 0.14 = 0.2
\end{aligned}$$

Now consider an additional random variable X_2 which is highly correlated to X_1. Let us observe the case where X_2 is a copy of X_1, i.e., it always takes the same realization as X_1 (this is unknown to the algorithm since the algorithm can only observe the realized values). The NB algorithm will still treat these two variables as conditionally indepedent and we want to observe the expected error introduced in this case.

	$P(X_1, X_2, Y = 0)$	$P(X_1, X_2, Y = 1)$	$\hat{y}$
$X_1 = X_2 = 0$	$0.7^2 \times 0.8$	$0.3^2 \times 0.2$	0
$X_1 = X_2 = 1$	$0.3^2 \times 0.8$	$0.7^2 \times 0.2$	1
$X_1 = 0, X_2 = 1$	$0.7 \times 0.3 \times 0.8$	$0.3 \times 0.7 \times 0.2$	1
$X_1 = 1, X_2 = 0$	$0.3 \times 0.7 \times 0.8$	$0.7 \times 0.3 \times 0.2$	1

Table 7.4: $P(X_1, X_2, Y = y)$

$$\begin{aligned}\text{Expected error} &= P(Y \neq \hat{y}) \\ &= P(Y = 1, X_1 = 0) + P(Y = 1, X_1 = 1) = 0.3 \\ &= 0.3 \times 0.2 + 0.3 \times 0.8 \\ &= 0.3\end{aligned}$$

Even though we have more data points we see that the error increases so with the naïve-based assumption we will not get a good expected error.

7.4 Logistic Regression

In linear regression, we tried to make

$$y \approx \mathbf{w}^\top \mathbf{x}$$

Here, y can take any real value, but what if y is like a boolean value (true/false) or a finite number of values? In other words, how to design a classifier using a similar model. A simple method inspired by the linear regression model is by independently fitting linear models for the probabilities of belonging to each class. Say, we do it for a binary classifier to get two linear predictors.

$$w_1^T x \quad w_2^T x \quad x \in \mathbb{R}^d$$

where d is the number of features. To transform these into probabilities, we perform an exponentiation and normalization.

$$P(Y = 1|x, w) = \frac{e^{w_1^T X}}{e^{w_2^T X} + e^{w_1^T X}}$$

$$P(Y = 0|x, w) = \frac{e^{w_2^T X}}{e^{w_2^T X} + e^{w_1^T X}}$$

On simplifying,

$$P(Y = 1|x, w) = \frac{1}{1 + e^{(w_2 - w_1)^T X}}$$

Here if we let $w_2 - w_1 = -w$,

$$P(Y = 1|x, w) = \frac{1}{1 + e^{-w^T x}} = \sigma(w^T x)$$

$$\implies P(Y = 1|x, w) = \sigma(w^T x)$$

Here, $\sigma(x)$ is called the *logistic function* which varies with x as follows:

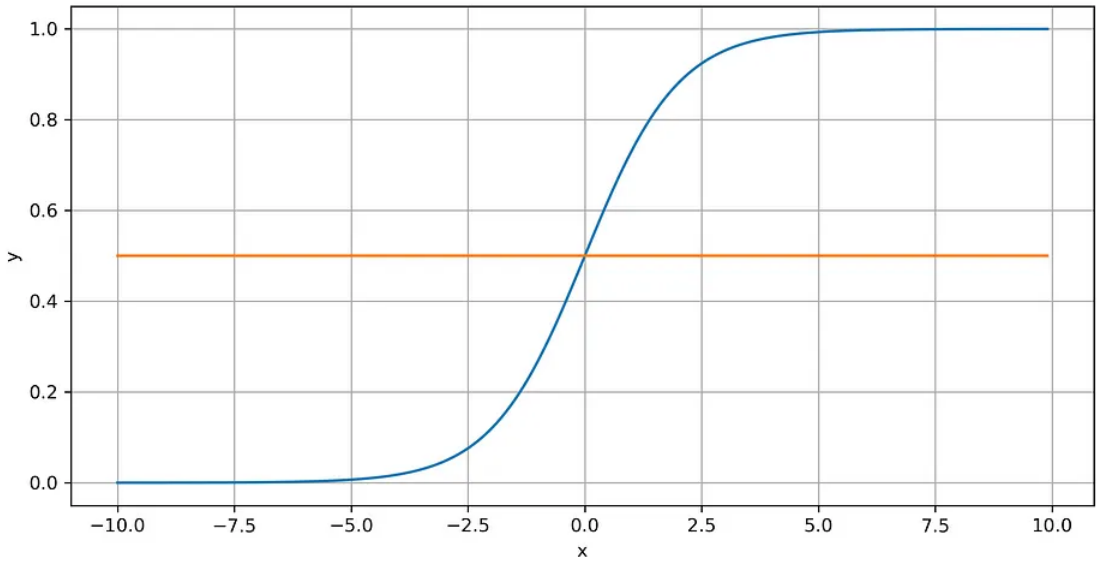

Figure 7.3: σ function

Our classifier predicts the class as 1 if $P(Y = 1|x, w) > P(Y = 0|x, w)$, ie,

$$\hat{y} = \begin{cases} 1 & P(Y = 1|x, w) > P(Y = 0|x, w) \\ 0 & P(Y = 1|x, w) \leq P(Y = 0|x, w) \end{cases}$$

On further simplification,

$$\begin{aligned} & P(Y = 1|x, w) > P(Y = 0|x, w) \\ \implies & \frac{1}{1 + e^{-w^T x}} > \frac{1}{1 + e^{w^T x}} \\ \implies & e^{2w_T x} > 1 \\ \implies & w^T x > 0 \end{aligned}$$

So, when our linear regression prediction is positive, this classifier classifies x as class 1 and otherwise class 0. This is a **Linear Classifier**. In essence, linear here signifies the linear nature of the classifying boundary, which in this case is the hyperplane $y = w^T x$.

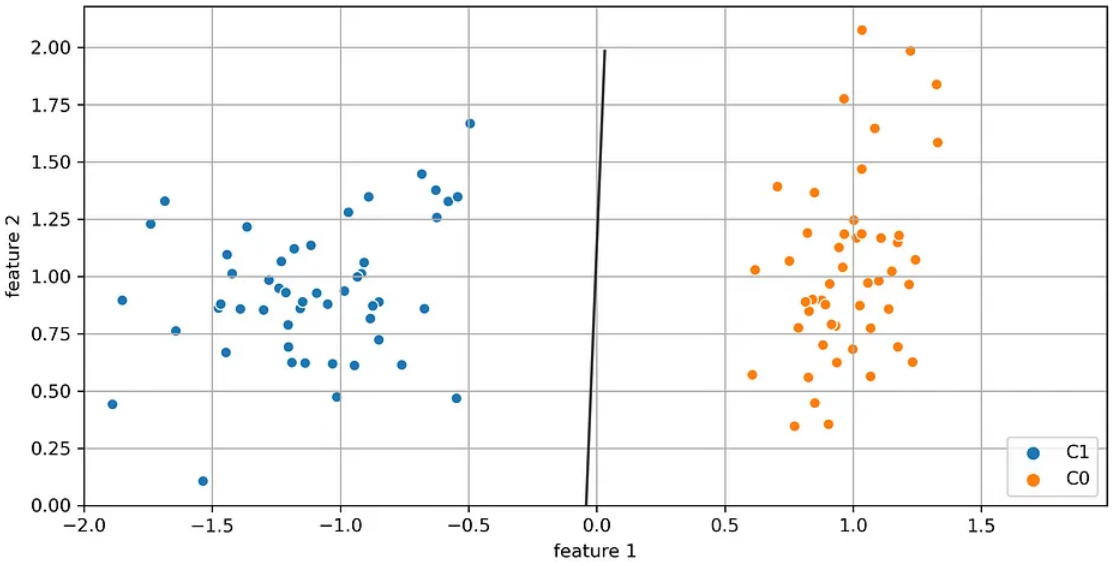

Figure 7.4: Linear boundary classifier