APL 405: Machine Learning for Mechanics

Lecture 6: Linear classification (logistic regression)

by

Rajdip Nayek

Assistant Professor, Applied Mechanics Department, IIT Delhi

Instructor email: rajdipn@am.iitd.ac.in

Recap of last lecture

- We introduced the linear regression model, which is a parametric model, for solving the regression problem
- Now we will look at basic parametric modelling techniques, particularly
 - Linear regression (covered in last lecture)
 - Logistic regression
- Linear regression
 - A loss-based perspective, using least squares error
 - A statistical perspective based on maximum likelihood, where the log-likelihood function was used
 - A closed form solution was derived
 - One-hot encoding to handle categorical inputs
- We will see that in logistic regression, we will not obtain a closed form solution

How to handle categorical input variables?

- We had mentioned earlier that input variables x can be numerical, catergorical, or mixed
- Assume that an input variable is categorical and takes only two classes, say A and B

• We can represent such an input variable x using 1 and 0
$$x = \begin{cases} 0, & \text{if } A \\ 1, & \text{if } B \end{cases}$$

For linear regression, the model effectively looks like

$$y = \theta_0 + \theta_1 x + \epsilon = \begin{cases} \theta_0 + \epsilon, & \text{if } \mathbf{A} \\ \theta_0 + \theta_1 + \epsilon, & \text{if } \mathbf{B} \end{cases}$$

If the input is a categorical variable with more than two classes, let's say A, B, C, and D, use one-hot encoding

$$\mathbf{x} = \begin{bmatrix} 1\\0\\0\\0 \end{bmatrix} \text{ if } \mathbf{A}, \quad \mathbf{x} = \begin{bmatrix} 0\\1\\0\\0 \end{bmatrix} \text{ if } \mathbf{B}, \quad \mathbf{x} = \begin{bmatrix} 0\\0\\1\\0 \end{bmatrix} \text{ if } \mathbf{C}, \quad \mathbf{x} = \begin{bmatrix} 0\\0\\0\\1 \end{bmatrix} \text{ if } \mathbf{D}$$

A statistical view of the Classification problem

- Classification \rightarrow learn relationships between some input variables $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_p \end{bmatrix}^T$ and a categorical output y
- The goal in classification is to take an input vector x and to assign it to one of M discrete classes 1,2 ..., M
- From a statistical perspective, classification amounts to predicting the conditional class probabilities

$$p(y = m | \mathbf{x})$$
 $y \to 1, 2, ..., M$

- $p(y = m | \mathbf{x})$ describes the probability for class *m* given that we know the input \mathbf{x}
- A probability over output y implies the output label y is a random variable (r.v.)



We consider y as a r.v. because the data (from real world) will always involve a certain amount of randomness (much like the output from linear regression that was probabilistic due to random error ε)

A statistical view of the Classification problem

- How to construct a classifier which can not only predict classes but also learn the class probabilities $p(y | \mathbf{x})$?
- Consider the simplest case of binary classification (M = 2) and y = -1 or 1
- In this binary classification case

 $p(y = 1 | \mathbf{x})$ will be modelled by $g(\mathbf{x})$

By the laws of probability,

$$p(y = 1|\mathbf{x}) + p(y = -1|\mathbf{x}) = 1$$

 $p(y = -1|\mathbf{x})$ will be modelled by $1 - g(\mathbf{x})$

- Since $g(\mathbf{x})$ is a model for a probability, it is natural to require that $0 \le g(\mathbf{x}) \le 1$ for any \mathbf{x}
- For a multi-class problem, the classifier should return a vector-valued function $g(\mathbf{x})$, where

$$\begin{bmatrix} p(y = 1 | \mathbf{x}) \\ p(y = 2 | \mathbf{x}) \\ \vdots \\ p(y = M | \mathbf{x}) \end{bmatrix} \text{ is modelled by } \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_M(\mathbf{x}) \end{bmatrix}$$

Since $g(\mathbf{x})$ models a probability vector, each element $g_m(\mathbf{x}) \ge 0$ and $\sum_{m=1}^M g_m(\mathbf{x}) = 1$

Logistic Regression model for binary classification

- Logistic regression can be viewed as an extension of linear regression that does (binary) classification (instead of regression)
- We wish to learn a function $g(\mathbf{x})$ that approximates the conditional probability of the positive class, $p(y = 1 | \mathbf{x})$
- Idea of Logisitic Regression: we start with the linear regression model which, without the noise term ϵ
 - Define logit, $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_p x_p = \mathbf{x}^T \boldsymbol{\theta}$
 - Logit takes values on the entire real line, but we need a function that returns a value in the interval [0, 1]
 - Squash the logit $z = \mathbf{x}^T \boldsymbol{\theta}$ into the interval [0, 1] by using the *logistic function*, $h(z) = \frac{e^z}{1+e^z}$



Logistic Regression

- Idea of Logisitic Regression: we start with the linear regression model which, without the noise term
 - Define logit, $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_p x_p = \mathbf{x}^T \boldsymbol{\theta}$
 - Logit takes values on the entire real line, but we need a function that returns a value in the interval [0, 1]
 - Squash the logit $z = \mathbf{x}^T \boldsymbol{\theta}$ into the interval [0, 1] by using the *logistic function* $h(z) = \frac{e^z}{1+e^z}$
- Recall that $g(\mathbf{x})$ was used to model for $p(y = 1 | \mathbf{x})$
- Using the logistic function for $g(\mathbf{x})$ restricts the values between 0 and 1 and can be interpreted as a probability



• It implicitly means that a model for $p(y = -1|\mathbf{x})$ is $1 - g(\mathbf{x}; \boldsymbol{\theta}) = 1 - \frac{e^{\mathbf{x}^T \boldsymbol{\theta}}}{1 + e^{\mathbf{x}^T \boldsymbol{\theta}}} = \frac{1}{1 + e^{\mathbf{x}^T \boldsymbol{\theta}}} = \frac{e^{-\mathbf{x}^T \boldsymbol{\theta}}}{1 + e^{-\mathbf{x}^T \boldsymbol{\theta}}}$

Logistic Regression

Logisitic Regression: Essentially linear regression appended with logistic function

• Logit,
$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_p x_p = \mathbf{x}^T \boldsymbol{\theta}$$

•
$$p(y = 1 | \mathbf{x}; \boldsymbol{\theta}) = g(\mathbf{x}; \boldsymbol{\theta}) = \frac{e^{\mathbf{x}^T \boldsymbol{\theta}}}{1 + e^{\mathbf{x}^T \boldsymbol{\theta}}}, \quad p(y = -1 | \mathbf{x}; \boldsymbol{\theta}) = 1 - g(\mathbf{x}; \boldsymbol{\theta}) = \frac{e^{-\mathbf{x}^T \boldsymbol{\theta}}}{1 + e^{-\mathbf{x}^T \boldsymbol{\theta}}}$$

- Logistic regression is a method for classification, not regression!
- The randomness in classification is statistically modelled by the class probability $p(y = m | \mathbf{x})$, instead of additive noise ϵ
- Like linear regression, logistic regression is also a parametric model, and we learn the parameters θ from training data

Training binary classification model with Maximum Likelihood

- Logistic function is a nonlinear function
- Therefore, a closed-form solution to logistic regression cannot be derived
- Maximum likelihood perspective of learning *θ* from training data $\widehat{\theta} = \operatorname{argmax} p(\mathbf{y} | \mathbf{X}; \theta)$



Similar to linear regression, we assume that the training data points are independent, and we consider the logarithm of the likelihood function for numerical reasons

θ

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \ln p(\boldsymbol{y}|\boldsymbol{X}; \boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{N} \ln \left(p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \right) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} -\ln \left(p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \right)$$

• Note that $p(y = 1 | \mathbf{x}; \boldsymbol{\theta})$ is modelled using $g(\mathbf{x}; \boldsymbol{\theta})$ which implies

$$-\ln p(y^{(i)}|\mathbf{x}^{(i)};\boldsymbol{\theta}) = \begin{cases} -\ln g(\mathbf{x}^{(i)};\boldsymbol{\theta}) & \text{if } y^{(i)} = 1\\ -\ln \left(1 - g(\mathbf{x}^{(i)};\boldsymbol{\theta})\right) & \text{if } y^{(i)} = -1 \end{cases}$$

Training binary classification model with Maximum Likelihood

Assume that the training data points are independent, and we consider the logarithm of the likelihood function for numerical reasons

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \ln p(\boldsymbol{y}|\boldsymbol{X}; \boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{N} \ln \left(p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \right) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} -\ln \left(p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \right)$$

• $p(y = 1 | \mathbf{x}; \boldsymbol{\theta})$ is modelled using $g(\mathbf{x}; \boldsymbol{\theta})$

$$-\ln p(y^{(i)}|\mathbf{x}^{(i)};\boldsymbol{\theta}) = \begin{cases} -\ln g(\mathbf{x}^{(i)};\boldsymbol{\theta}) & \text{if } y^{(i)} = 1\\ -\ln \left(1 - g(\mathbf{x}^{(i)};\boldsymbol{\theta})\right) & \text{if } y^{(i)} = -1 \end{cases}$$

Cross-entropy loss function, $L\left(y^{(i)}, g(\mathbf{x}^{(i)};\boldsymbol{\theta})\right)$

- Cross entropy loss can be used for any binary classifier, not just logistic regression, that predicts class probabilities $g(\mathbf{x}; \boldsymbol{\theta})$
- The corresponding cost function (or average loss function)

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} -\ln g(\mathbf{x}^{(i)}; \boldsymbol{\theta}) & \text{if } y^{(i)} = 1\\ -\ln \left(1 - g(\mathbf{x}^{(i)}; \boldsymbol{\theta})\right) & \text{if } y^{(i)} = -1 \end{cases}$$

Training Logistic Regression model with Maximum Likelihood

We can write the cost function in more detail for logistic regression

For
$$y^{(i)} = 1$$
, $g(\mathbf{x}^{(i)}; \boldsymbol{\theta}) = \frac{e^{(\mathbf{x}^{(i)})^T \boldsymbol{\theta}}}{1 + e^{(\mathbf{x}^{(i)})^T \boldsymbol{\theta}}} = \frac{e^{y^{(i)} (\mathbf{x}^{(i)})^T \boldsymbol{\theta}}}{1 + e^{y^{(i)} (\mathbf{x}^{(i)})^T \boldsymbol{\theta}}}$

For
$$y^{(i)} = -1$$
, $1 - g(\mathbf{x}^{(i)}; \boldsymbol{\theta}) = \frac{1}{1 + e^{(\mathbf{x}^{(i)})^T \boldsymbol{\theta}}} = \frac{e^{-(\mathbf{x}^{(i)})^T \boldsymbol{\theta}}}{1 + e^{-(\mathbf{x}^{(i)})^T \boldsymbol{\theta}}} = \frac{e^{y^{(i)}(\mathbf{x}^{(i)})^T \boldsymbol{\theta}}}{1 + e^{y^{(i)}(\mathbf{x}^{(i)})^T \boldsymbol{\theta}}}$

Hence, we get the same expression in both cases and can write the cost function compactly as:

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} -\ln g(\mathbf{x}^{(i)}; \boldsymbol{\theta}) & \text{if } y^{(i)} = 1\\ -\ln \left(1 - g(\mathbf{x}^{(i)}; \boldsymbol{\theta})\right) & \text{if } y^{(i)} = -1 \end{cases}$$
$$= \frac{1}{N} \sum_{i=1}^{N} -\ln \frac{e^{y^{(i)}(\mathbf{x}^{(i)})^{T} \boldsymbol{\theta}}}{1 + e^{y^{(i)}(\mathbf{x}^{(i)})^{T} \boldsymbol{\theta}}} = \frac{1}{N} \sum_{i=1}^{N} -\ln \frac{1}{1 + e^{-y^{(i)}(\mathbf{x}^{(i)})^{T} \boldsymbol{\theta}}} = \frac{1}{N} \sum_{i=1}^{N} \ln \left(1 + e^{-y^{(i)}(\mathbf{x}^{(i)})^{T} \boldsymbol{\theta}}\right)$$

Training Logistic Regression model with Maximum Likelihood

Cost function in <u>logistic regression</u> is given by:

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \ln\left(1 + e^{-y^{(i)} (\mathbf{x}^{(i)})^{T} \boldsymbol{\theta}}\right)$$

Logistic loss function, $L(y^{(i)}, \mathbf{x}^{(i)}; \boldsymbol{\theta})$

• The logistic loss $L(y^{(i)}, \mathbf{x}^{(i)}; \boldsymbol{\theta})$ above is a <u>special case of the cross-entropy loss</u>

• Learning a logistic regression model thus amounts to solving the optimization problem:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} J(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \ln \left(1 + e^{-y^{(i)} \left(\mathbf{x}^{(i)} \right)^{T} \boldsymbol{\theta}} \right)$$

 Contrary to linear regression with squared error loss, the above problem has no closed-form solution, so we have to use numerical optimization instead

Predictions using Logistic Regresion

- Logistic regression predicts class probabilities for a test input x*
 - by first learning θ from training data, and
 - then computing $g(\mathbf{x}^*)$, which is the model for $p(y^* = 1 | \mathbf{x}^*)$
- However, sometimes we want to make a "hard" prediction for the test input \mathbf{x}^*
 - E.g., whether is $\hat{y}(\mathbf{x}^*) = 1$ or $\hat{y}(\mathbf{x}^*) = -1$ in binary classification?
 - Recall, in kNN and decision trees, we made "hard" predictions
- To make hard predictions with logistic regression model, we add a final step, in which the predicted probabilities are turned into a class prediction
- The most common approach is to let $\hat{y}(\mathbf{x}^*)$ be the most probable class \leftarrow the class having the highest probability
- For binary classification, we can express this as:

r = 0.5 minimises the so-called misclassification rate

$$\hat{y}(\mathbf{x}^*) = \begin{cases} 1 & \text{if } g(\mathbf{x}^*) > r \\ -1 & \text{if } g(\mathbf{x}^*) \le r \end{cases} \text{ with decision threshold } r = 0.5 \text{ (why?)}$$

Decision Boundaries of Logistic Regression

■ Decision boundary ← The point(s) where the prediction changes from from one class to another



- The decision boundary for binary classification can be computed by solving the equation $g(\mathbf{x}) = 1 - g(\mathbf{x})$ meaning $p(y = 1 | \mathbf{x}; \boldsymbol{\theta}) = p(y = -1 | \mathbf{x}; \boldsymbol{\theta})$
- The solutions to this equation are points in the input space for which the two classes are predicted to be equally probable

Decision Boundaries of Logistic Regression

The decision boundary for binary classification can be computed by solving the equation

$$g(\mathbf{x}) = 1 - g(\mathbf{x})$$
 meaning $p(y = 1 | \mathbf{x}; \boldsymbol{\theta}) = p(y = -1 | \mathbf{x}; \boldsymbol{\theta})$

- The solutions to this equation are points in the input space for which the two classes are predicted to be equally probable
- For binary logistic regression, it means

$$\frac{e^{\mathbf{x}^T \boldsymbol{\theta}}}{1 + e^{\mathbf{x}^T \boldsymbol{\theta}}} = \frac{1}{1 + e^{\mathbf{x}^T \boldsymbol{\theta}}} \iff e^{\mathbf{x}^T \boldsymbol{\theta}} = 1 \iff \mathbf{x}^T \boldsymbol{\theta} = 0$$

- The equation $\mathbf{x}^T \boldsymbol{\theta} = 0$ parameterises a (linear) hyperplane
- Therefore, the decision boundaries in logistic regression always have the shape of a (linear) hyperplane



Prediction and Decision Boundaries of Logistic Regression

• For binary classification, we can express this as:

$$\hat{y}(\mathbf{x}^*) = \begin{cases} 1 & \text{if } g(\mathbf{x}^*) > r \\ -1 & \text{if } g(\mathbf{x}^*) \le r \end{cases} \text{ with decision threshold } r = 0.5$$

Choosing r = 0.5 minimises the so-called misclassification rate

• The decision boundary for logistic regression lies at $\mathbf{x}^T \boldsymbol{\theta} = \mathbf{0}$

 \Rightarrow The sign of the expression $\mathbf{x}^T \boldsymbol{\theta}$ determines if we are predicting the <u>positive</u> (1) or the <u>negative</u> (-1) class

Compactly, one can write the test output prediction for a test input x* from a logistic regression as

 $\hat{y}(\mathbf{x}^*) = \operatorname{sign}(\mathbf{x}^{*T}\boldsymbol{\theta})$

Linear vs Non-linear classifiers

- A classifier whose <u>decision boundaries are linear</u> hyperplanes is a *linear classifier*
- Logistic regression is a linear classifier
- kNN and Decision Trees are non-linear classifiers



- Note that the term 'linear' has a different sense for linear regression and for linear classification
 - Linear regression is a model which is linear in its parameters,
 - Linear classifier is a model linear whose decision boundaries are linear

Logistic Regression for more than two classes

- For the binary problem, we used the logistic function to design a model for $g(\mathbf{x})$
 - $g(\mathbf{x})$ a scalar-valued function representing $p(y = 1 | \mathbf{x})$
- For a multi-class problem (M classes), the classifier should return a vector-valued function g(x), where

$$\begin{bmatrix} p(y = 1 | \mathbf{x}) \\ p(y = 2 | \mathbf{x}) \\ \vdots \\ p(y = M | \mathbf{x}) \end{bmatrix} \text{ is modelled by } \boldsymbol{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_M(\mathbf{x}) \end{bmatrix}$$

Since $g(\mathbf{x})$ models a probability vector, each element $g_m(\mathbf{x}) \ge 0$ and $\sum_{m=1}^M g_m(\mathbf{x}) = 1$

- For this purpose, we define M different logits, $z_m = (\theta^m)^T \mathbf{x}$, m = 1, 2, ..., M
- The use the softmax function (a vector-valued generalization of logistic function)

softmax(
$$\mathbf{z}$$
) $\triangleq \frac{1}{\sum_{m=1}^{M} e^{\mathbb{Z}m}} \begin{bmatrix} e^{\mathbb{Z}_1} \\ e^{\mathbb{Z}_2} \\ \vdots \\ e^{\mathbb{Z}M} \end{bmatrix}$

- *z* is an *M*-dimensional vector
- softmax(z) also returns a vector of the same dimension
- By construction, the output vector always sums to 1, and each element is always ≥ 0

Multi-class Logistic Regression model

We have now combined linear regression and softmax function to model multi-class probabilities

$$g(\mathbf{z}) = \operatorname{softmax}(\mathbf{z}),$$
 where $\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_M \end{bmatrix} = \begin{bmatrix} (\boldsymbol{\theta}^1)^T \mathbf{x} \\ (\boldsymbol{\theta}^2)^T \mathbf{x} \\ \vdots \\ (\boldsymbol{\theta}^M)^T \mathbf{x} \end{bmatrix}$

• Equivalently, we can write out the individual class probabilities, that is, the elements of the vector $g_m(\mathbf{x})$

$$g_m(\mathbf{x}) = \frac{e^{(\boldsymbol{\theta}^m)^T \mathbf{x}}}{\sum_{j=1}^M e^{(\boldsymbol{\theta}^j)^T \mathbf{x}}} \qquad m = 1, 2, \dots, M$$

- This is the *multiclass* logistic regression model
- Note that this construction uses M parameter vectors θ^1 , ..., θ^M (one for each class)
 - Note the number of parameters to learn grows with M