APL 405: Machine Learning in 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 $\mathbf{x}$ can be numerical, catergorical, or mixed

- Assume that an input variable is categorical and takes only two classes, say $\mathbf{A}$ and $\mathbf{B}$

- We can represent such an input variable x using 1 and 0

$$x = \begin{cases} 0, & \text{if } \mathbf{A} \\ 1, & \text{if } \mathbf{B} \end{cases}$$

- 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 $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$, 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 → learn relationships between some input variables $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_p]^T$ and a categorical output y
- The goal in classification is to take an input vector $\mathbf{x}$ and to assign it to one of M discrete classes $1,2 \ldots, M$

- From a statistical perspective, classification amounts to predicting the conditional class probabilities

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

- $p(y = m|\mathbf{x})$ 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 $p(y \mid \mathbf{x})$?

- Consider the simplest case of binary classification $(M = 2)$ and $y = -1 \text{ or } 1$
- In this binary classification case

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

- By the laws of probability,

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

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

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

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

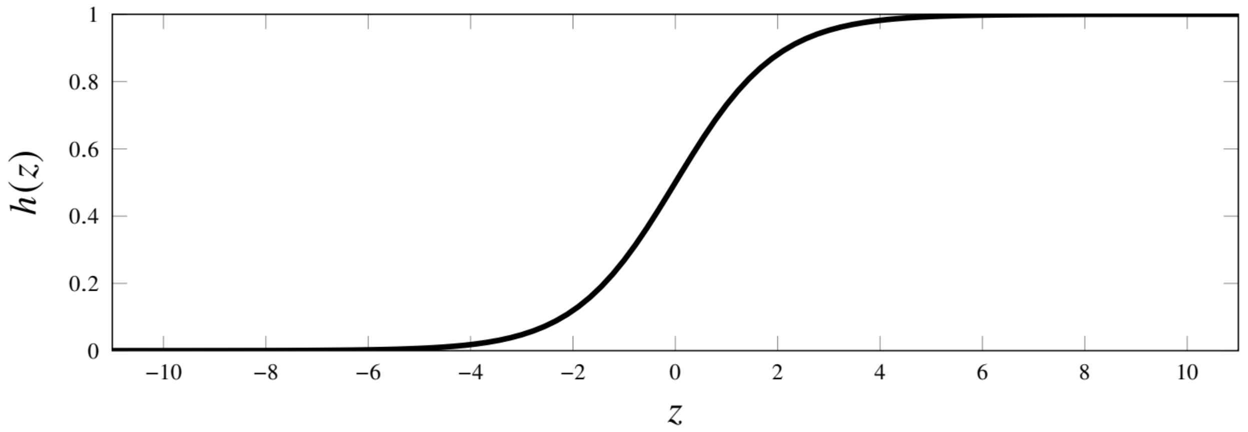
Since $\boldsymbol{g}(\mathbf{x})$ models a probability vector, each element $g_m(\mathbf{x}) \geq 0$ and $\sum_{m=1}^{M} g_m(\mathbf{x}) = 1$

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 Logistic Regression: we start with the linear regression model which, without the noise term ϵ
  - Define logit, $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p = \mathbf{x}^T \boldsymbol{\theta}$
  - 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,* $h(z) = \frac{e^z}{1+e^z}$



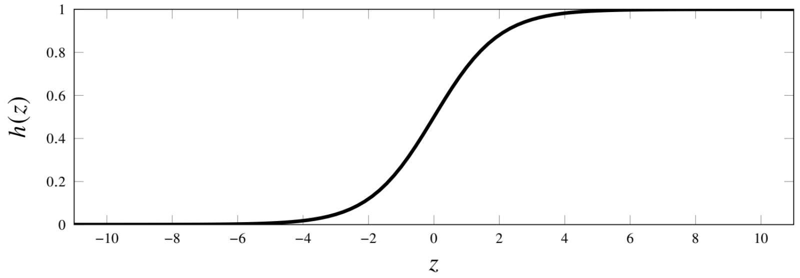
Logistic Regression

- Idea of Logistic Regression: we start with the linear regression model which, without the noise term
  - Define logit, $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p = \mathbf{x}^T \boldsymbol{\theta}$
  - 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

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



- 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

- Logistic Regression: Essentially linear regression appended with logistic function
  - Logit, $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p = \mathbf{x}^T \boldsymbol{\theta}$
  - $p(y = 1|\mathbf{x}; \boldsymbol{\theta}) = g(\mathbf{x}; \boldsymbol{\theta}) = \frac{e^{\mathbf{x}^T\boldsymbol{\theta}}}{1+e^{\mathbf{x}^T\boldsymbol{\theta}}}$, $\quad p(y = -1|\mathbf{x}; \boldsymbol{\theta}) = 1 - g(\mathbf{x}; \boldsymbol{\theta}) = \frac{e^{-\mathbf{x}^T\boldsymbol{\theta}}}{1+e^{-\mathbf{x}^T\boldsymbol{\theta}}}$

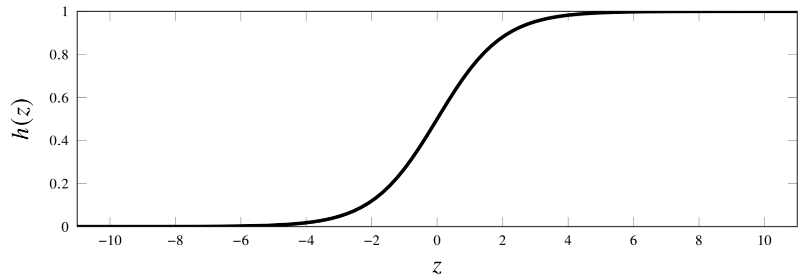
- Logistic regression is a method for classification, not regression (despite its misleading name)!

- 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 $\boldsymbol{\theta}$ 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 $\boldsymbol{\theta}$ from training data

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{y}|\mathbf{X};\boldsymbol{\theta})$$

- 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

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

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

$$-\ln p(y_i|\mathbf{x}_i;\boldsymbol{\theta}) = \begin{cases} -\ln g(\mathbf{x}_i;\boldsymbol{\theta}) & \text{if } y_i = 1 \\ -\ln\big(1 - g(\mathbf{x}_i;\boldsymbol{\theta})\big) & \text{if } y_i = -1 \end{cases}$$

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

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

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

$$-\ln p(y_i|\mathbf{x}_i;\boldsymbol{\theta}) = \begin{cases} -\ln g(\mathbf{x}_i;\boldsymbol{\theta}) & \text{if } y_i = 1 \\ -\ln\big(1 - g(\mathbf{x}_i;\boldsymbol{\theta})\big) & \text{if } y_i = -1 \end{cases}$$

Binary Cross-entropy loss function, $L\big(y_i, g(\mathbf{x}_i;\boldsymbol{\theta})\big)$

- 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)

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

Training Logistic Regression model with Maximum Likelihood

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

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

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

- 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 logistic regression is given by:

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \underbrace{\ln\left(1 + e^{-y_i \mathbf{x}_i{}^T \boldsymbol{\theta}}\right)}_{\text{Logistic loss function, } L(y_i, \mathbf{x}_i; \boldsymbol{\theta})}$$

- The logistic loss $L(y_i, \mathbf{x}_i; \boldsymbol{\theta})$ above is a special case of the cross-entropy loss

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

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

- 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 Regression

- Logistic regression predicts class probabilities for a test input $\mathbf{x}_*$
  - by first learning $\boldsymbol{\theta}$ 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* ← the class having the highest probability

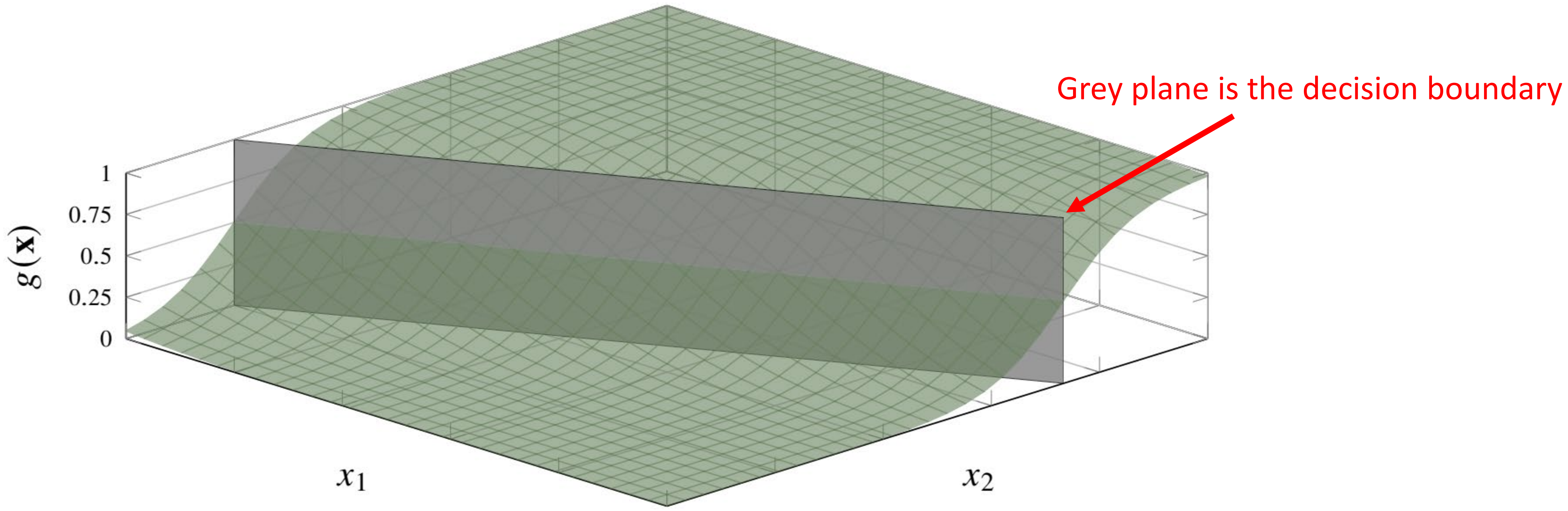
- For binary classification, we can express this as:

r = 0.5 minimizes the so-called misclassification rate

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

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

- 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

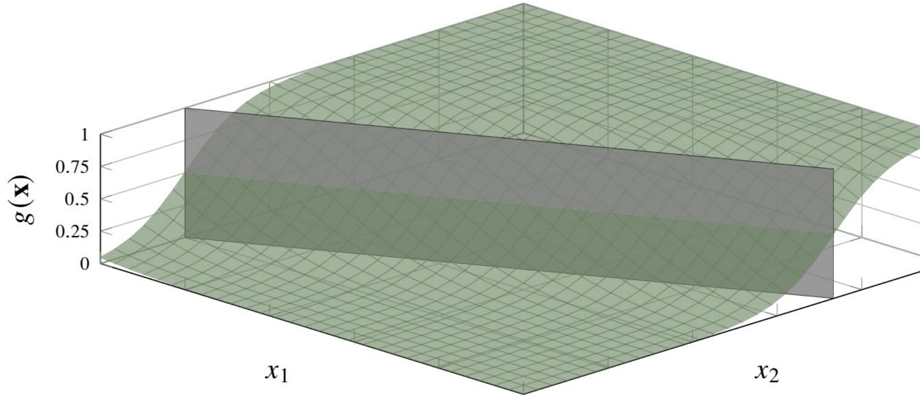
- The decision boundary for binary classification can be computed by solving the equation

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

- 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

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

- The equation $\mathbf{x}^T\boldsymbol{\theta} = 0$ parameterizes 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:

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

- Choosing r = 0.5 minimizes the so-called misclassification rate

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

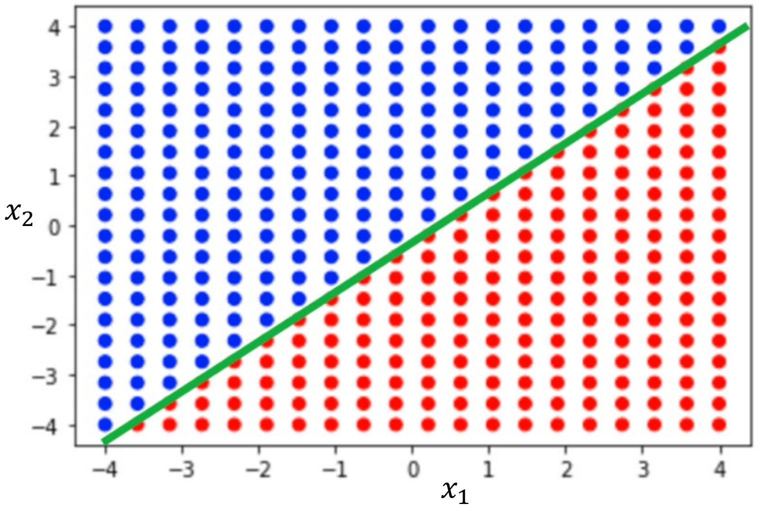
$\Longrightarrow$ The sign of the expression $\mathbf{x}^T\boldsymbol{\theta}$ determines if we are predicting the <u>positive</u> (1) or the <u>negative</u> (-1) class

- Compactly, one can write the test output prediction for a test input $\mathbf{x}_*$ from a logistic regression as

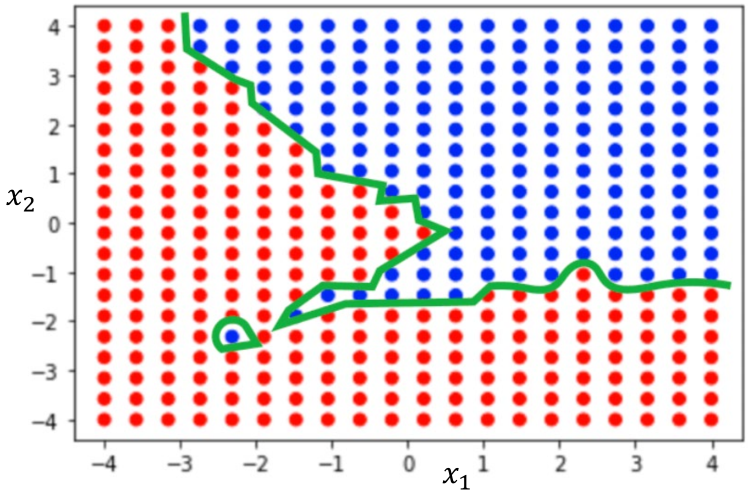
$$\hat{y}(\mathbf{x}_*) = \mathrm{sign}({\mathbf{x}_*}^T\boldsymbol{\theta})$$

Linear vs Non-linear classifiers

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



Linear classifier



Non-Linear classifier

- 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 $\boldsymbol{g}(\mathbf{x})$, where

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

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

- For this purpose, we define M different logits, $z_m = (\boldsymbol{\theta}^m)^T\mathbf{x}$, $m = 1,2, \dots , M$

- The use the *softmax* function (a vector-valued generalization of logistic function)

$$\text{softmax}(\boldsymbol{z}) \triangleq \frac{1}{\sum_{m=1}^{M} e^{z_m}} \begin{bmatrix} e^{z_1} \\ e^{z_2} \\ \vdots \\ e^{z_M} \end{bmatrix}$$

- $\boldsymbol{z}$ is an M-dimensional vector
- $\text{softmax}(\boldsymbol{z})$ 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

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

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

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

- This is the *multiclass* logistic regression model

- Note that this construction uses M parameter vectors $\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^M$ (one for each class)
  - Note the number of parameters to learn grows with M