Overview

Recall module is roughly split into four parts:

1. Random events: counting, events, axioms of probability, Bayes, independence
2. Random variables: discrete RVs, mean and variance, correlation, conditional expectation
3. Inequalities and laws of large numbers: Markov, Chebyshev bounds, sample mean, weak law of large numbers, central limit theorem, confidence intervals, bootstrapping
4. Statistical models: logistic regression, continuous random variables, least squares

Overview

- Classification
- Logistic Regression with Two Classes
- Gradient Descent
- Logistic Regression with Multiple Classes
- Probabilistic Interpretation

Classification with Two Classes

Examples:

- Email: spam or not ?
- Online transactions: fraudulent or not ?
- Tumor: malignant or benign ?

In general, the setup is that we have some observed data and using that we want to predict the class.

- Observed data is the input variable or features e.g. text of email, location, nationality, size of tumour
- Our prediction is the output variable or target variable. In classification with two classes it only takes values -1 or 1 and is often referred to as the **label**[1].

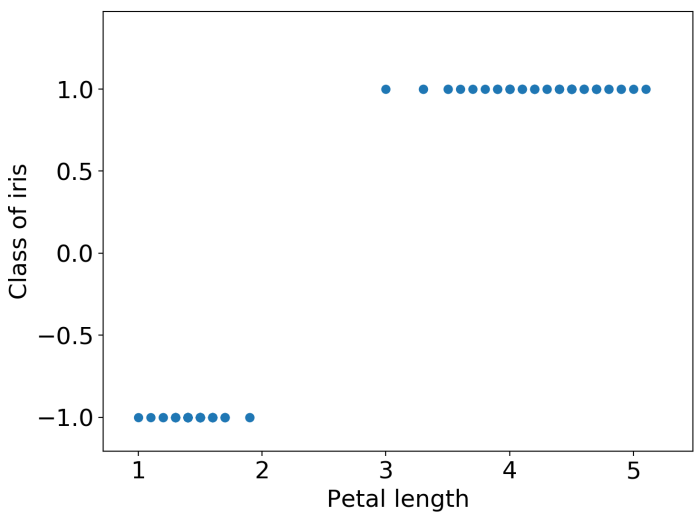
We also have some examples which have already been labelled e.g. emails which have been manually labelled as spam or not. This is our **training data**.

[1] Note could also use values 0 and 1 rather than -1 and 1, leave this as an exercise

Easy Example

Our task is predict variety of iris (either Iris setosa or Iris versicolor).

- Label Iris setosa as -1 and Iris versicolor as $+1$.
- Our observation is is the petal length.
- We have the following training data:



Some Notation

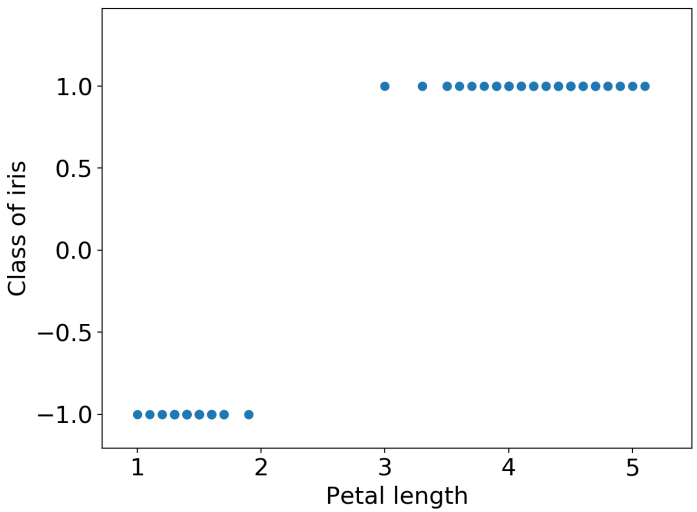
Training data:

Petal Length (x)	Label (y)
1.0	-1
3.0	$+1$
3.4	$+1$
$\vdots$	$\vdots$

- m=number of training examples
- x="input" variable/features (just one here)
- y="output" variable/"target" variable

- $(x^{(i)}, y^{(i)})$ the ith training example
- $x^{(1)} = 1.0$, $y^{(1)} = -1$, $x^{(2)} = 3.0$, $y^{(2)} = +1$
- $x^{(i)}$ can be a vector (an array) with multiple values when there are multiple features. We write $x_j^{(i)}$ for the j'th element of $x^{(i)}$

Easy Example[2]

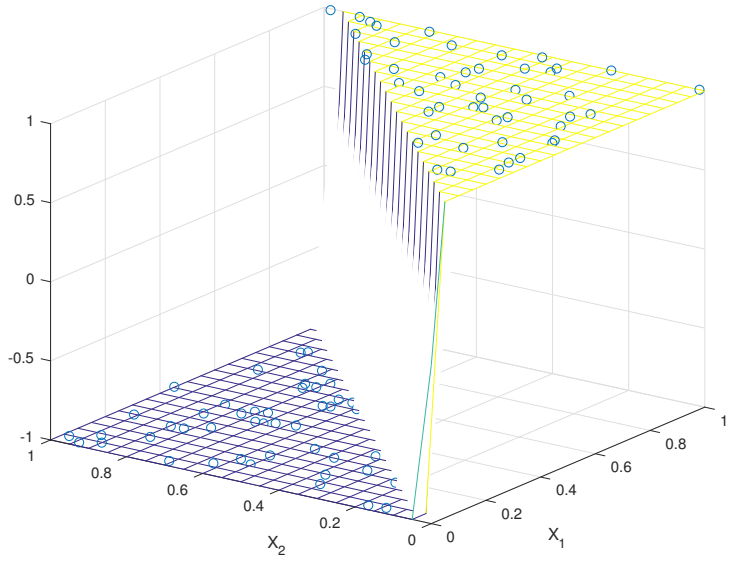


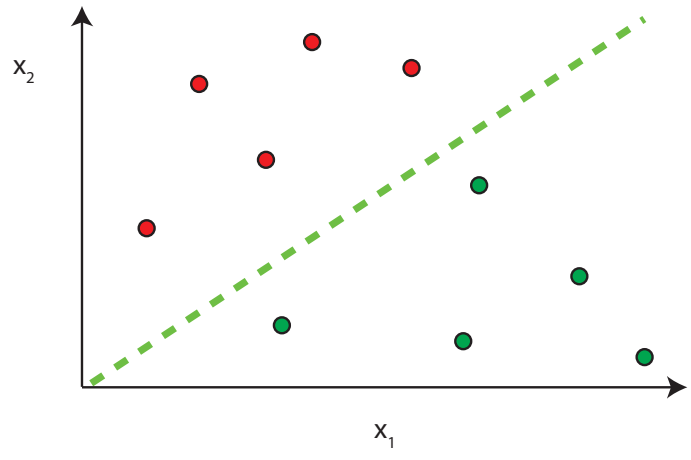
- Obvious solution: if petal length less than about 2.5 then predict -1 else predict $+1$
- i.e. $y = sign(x - 2.5)$ where $sign(z) = -1$ when $z < 0$ and $sign(z) = +1$ when $z \geq 0$
- Note that $x - 2.5 = 0$ defines a **decision boundary**. When $x - 2.5 < 0$ we predict -1, when $x - 2.5 > 0$ we predict $+1$

[2] https://archive.ics.uci.edu/ml/datasets/Iris

Slightly Harder Example

- Now have two features x_1 and x_2 e.g. x_1 might be tumour size and x_2 patient age.





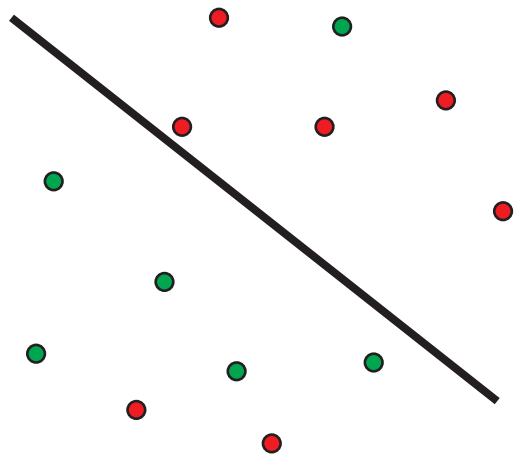
- When $x_1 \geq x_2$ then we predict $+1$, else predict -1
- i.e. $y = sign(x_1 - x_2)$
- The decision boundary is $x_1 - x_2 = 0$. When $x_1 - x_2 < 0$ we predict -1, when $x_1 - x_2 > 0$ we predict $+1$

Automating This Process

- Define a **model** or **hypothesis** $y = h_\theta(x)$ with **parameters** $\theta_0, \theta_1, \ldots$.
- E.g.
  - $h_\theta(x) = sign(\theta_0 + \theta_1 x)$, selecting $\theta_0 = -2.5$, $\theta_1 = 1$ then $y = sign(x - 2.5)$
  - $h_\theta(x) = sign(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$, selecting $\theta_0 = 0$, $\theta_1 = 1$, $\theta_2 = -1$, then $y = sign(x_1 - x_2)$
- More generally, when have n features then use $h_\theta(x) = sign(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n)$.
- The decision boundary $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n = 0$ is a line so this is a **linear** model.
- The value of the parameters is unknown. So we use the training data to select parameter values that give a model that makes accurate predictions, i.e. we **train** the model (or **learn** the parameter values) using the training data.

Linear Separability

- For the linear model used in logistic regression the decision boundary is a line. If the data can be cleanly separated by a line (all the -1 points on one side of the line, all the $+1$ points on the other side) then the data is said to be **linearly separable**
- Not all data is linearly separable e.g.



Some More Notation

- We add a dummy feature $x_0 = 1$, just to streamline notation
- Feature vector $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$ (we will always use column vectors)
- The transpose of a vector $x^T = [x_0, x_1, x_2, \dots, x_n]$
- Parameter vector $\theta^T = [\theta_0, \theta_1, \dots, \theta_n]$
- Product $\theta^T x = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n$
- So we can write our linear model as $h_\theta(x) = sign(\theta^T x)$

Training The Model

- We use the training data to select parameter values that give a model that makes accurate predictions, but how do we measure accuracy?
- We might consider the **0-1 loss function**:

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}(h_\theta(x^{(i)}) \neq y^{(i)})$$

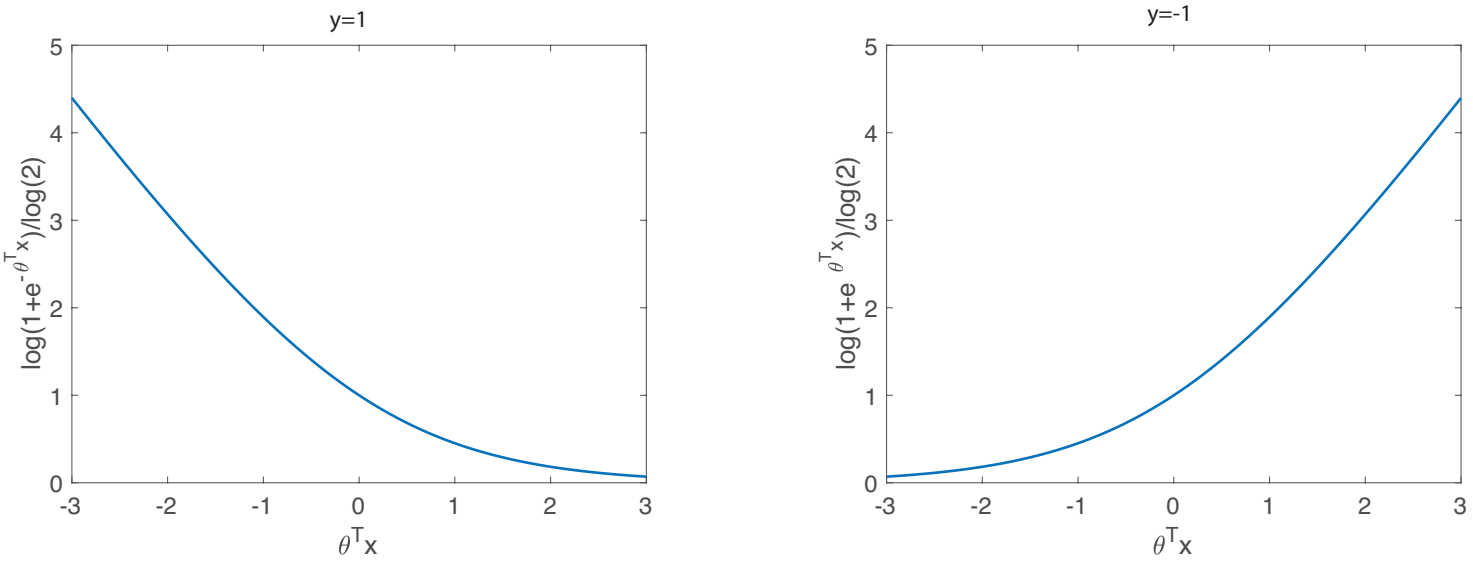
  where indicator function $\mathbb{I} = 1$ if $h_\theta(x^{(i)}) \neq y^{(i)}$ and $\mathbb{I} = 0$ otherwise. But hard to work with.
- For logistic regression we use:

$$\frac{1}{m}\sum_{i=1}^{m}\log(1 + e^{-y^{(i)}\theta^T x^{(i)}})/log(2)$$

  noting that $y = -1$ or $y = 1$. Scaling by $\log(2)$ is optional, but makes the loss 1 when $y^{(i)}\theta^T x^{(i)} = 0$.

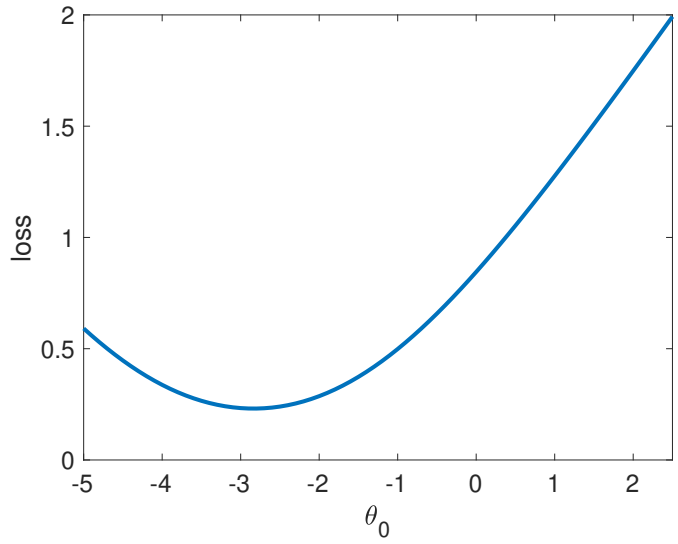
Logistic Regression Cost Function

Loss function: $\log(1 + e^{-y\theta^T x})/log(2)$



- So a small penalty when $\theta^T x \gg 0$ and $y = 1$, and when $\theta^T x \ll 0$ and $y = -1$.
- Minimising this thus gives preference to θ values that push $\theta^T x$ well away from the decision boundary $\theta^T x = 0$.

Iris Example



- Model: $y = sign(\theta_0 + x)$, with unknown parameter θ_0
- Loss function: $\frac{1}{m}\sum_{i=1}^{m} \log(1 + e^{-y^{(i)}\theta^T x^{(i)}})$
- Plot loss function as vary value of θ_0, observe there is a minimum around -2.8 (close to the -2.5 we estimated by eye)

Logistic Regression Summary

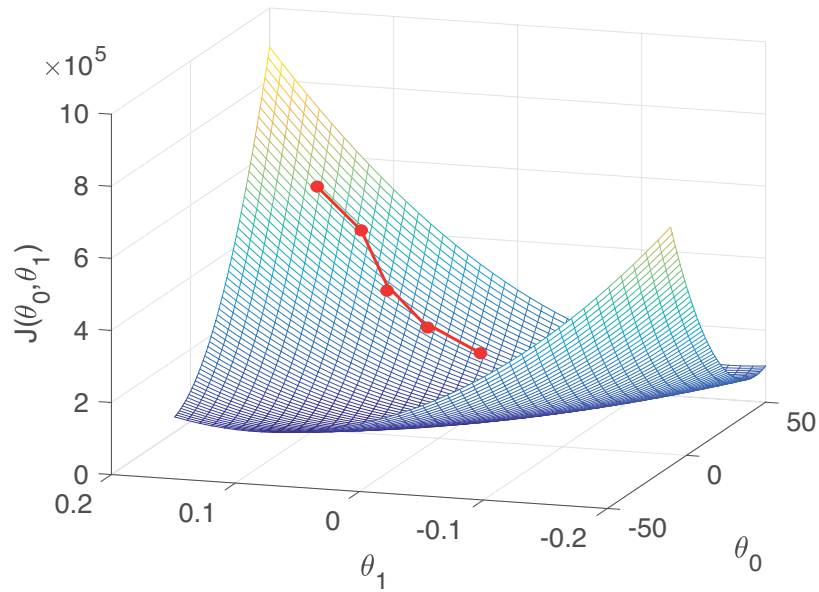
- Hypothesis: $h_\theta(x) = sign(\theta^T x)$
- Parameters: θ
- Training data: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$
- Cost Function: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \log(1 + e^{-y^{(i)} \theta^T x^{(i)}})$
- Learn Parameters: Select θ that minimises $J(\theta)$

Minimising The Cost Function

Need to select θ that minimises $J(\theta)$. Brute force search over the values of θ is inefficient, can we be smarter ?

- Start with some θ
- Repeat:

  Update θ to new value which makes $J(\theta)$ smaller

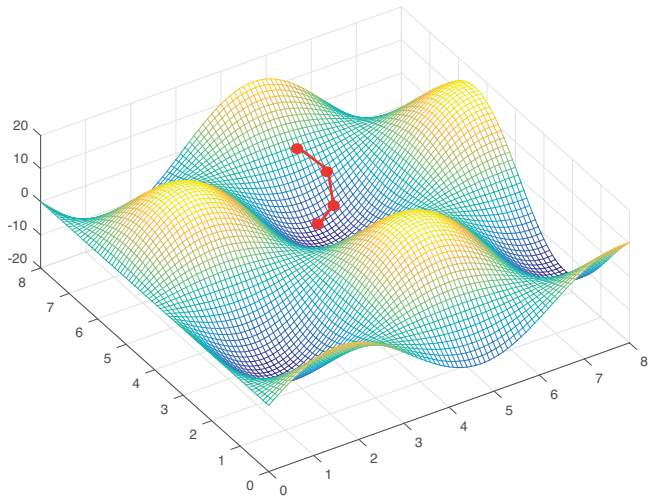


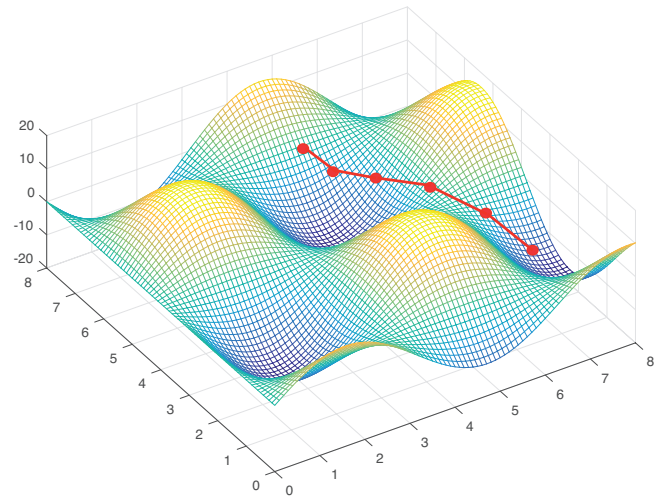
- When curve is "bowl shaped" or convex (as it always is for logistic regression) then this must eventually find the minimum.

Minimising The Cost Function

- Start with some θ
- Repeat:

  Update θ to new value which makes $J(\theta)$ smaller
- When curve has several minima then we can't be sure which we will converge to.
- Might converge to a local minimum, not the global minimum





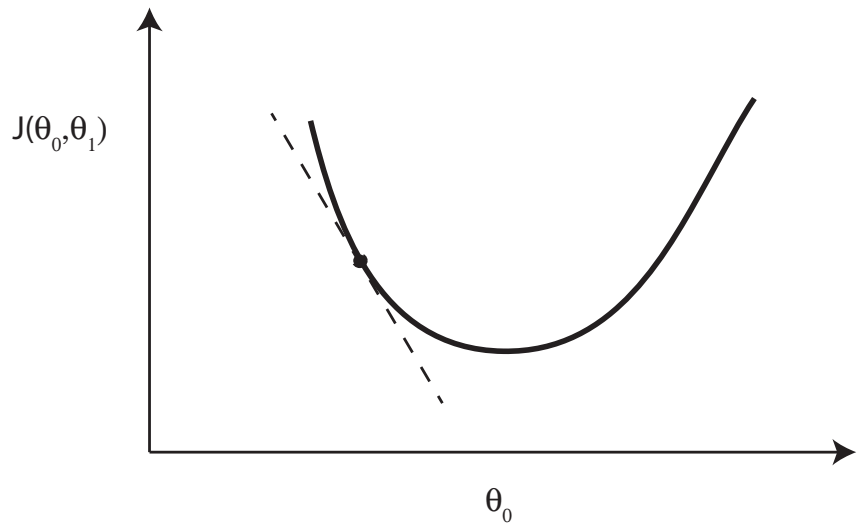
Gradient Descent

Repeat: Update θ_0 and θ_1 to new value which makes $J(\theta_0, \theta_1)$ smaller

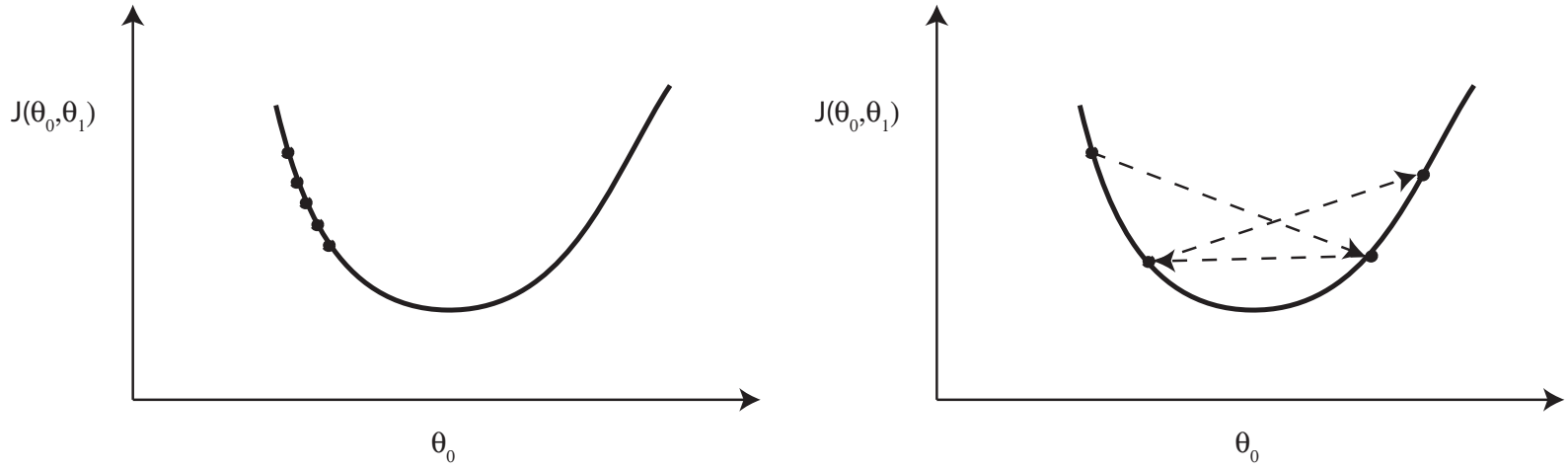
- One option: carry out local search of θ_0 and θ_1 to find one that decreases J.
- Another option: gradient descent:

$$\textit{for } i = 1 \textit{ to } n \{\textit{temp}_i := \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta)\}$$

$$\textit{for } i = 1 \textit{ to } n \{\theta_i := \textit{temp}_i\}$$



Gradient Descent



- Selecting step size α too small will mean it takes a long time to converge to minimum
- But selecting α too large can lead to us overshooting the minimum
- We need to adjust α so that algorithm converges in a reasonable time.

Gradient Descent

For $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \log(1 + e^{-y^{(i)}\theta^T x^{(i)}})$:

- $\frac{\partial}{\partial\theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m} -y^{(i)} x_j^{(i)} \frac{e^{-y^{(i)}\theta^T x^{(i)}}}{1+e^{-y^{(i)}\theta^T x^{(i)}}}$
- (Remember $\frac{d\log(x)}{dx} = \frac{1}{x}$, $\frac{d\exp(x)}{dx} = exp(x)$ and chain rule $\frac{df(z(x))}{dx} = \frac{df}{dz}\frac{dz}{dx}$)

So gradient descent algorithm is:

- Start with some θ
- Repeat:

  for j=0 to n $\{tempj := \theta_j + \frac{\alpha}{m}\sum_{i=1}^{m} y^{(i)} x_j^{(i)} \frac{e^{-y^{(i)}\theta^T x^{(i)}}}{1+e^{-y^{(i)}\theta^T x^{(i)}}}\}$

  for j=0 to n $\{\theta_j := tempj\}$
- $J(\theta)$ is convex, has a single minimum. Iteration moves downhill until it reaches the minimum

Example: Sentiment Analysis

Our task is to classify a movie review as either positive or negative. We're given training data[3] consisting of the text of the reviews plus a label as to whether the review is positive or negative (added manually). First thing we need to do is decide on what our features are. We can't use text directly, we need to map it to a numerical vector of values. **Bag of Words** approach:

- Delete all the little words (of, the, a etc), truncate word endings (e.g. liking, liked, likeable all get trimmed back to "lik")
- Form a list of all the unique words that are left, and number the words in the list 1,2,..., n
- Map the text of a review to a vector of length n by setting element i of the vector equal to the number of times the ith word in the dictionary appears in the review.

[3] Polarity dataset, https://www.cs.cornell.edu/people/pabo/movie-review-data/

Example: Sentiment Analysis

E.g.

- Suppose our dictionary had $n = 4$ words: 1) machine 2) learning 3) regression 4) logistic.
- And our text is "logistic regression is an important regression method in machine learning"
- We map this text to feature vector

$$\begin{matrix} 1 \\ 1 \\ 2 \\ 1 \end{matrix}$$

Now we can apply our logistic regression methods as usual. The model predicts the right sentiment about 80% of the time, not too bad!

Probabilistic Interpretation: Logistic Regression

- Label Y only takes values -1 or 1.
- Assume

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

  and recall $y = 1$ or $y = -1$ only.
- Assume each observation is independent.
- For the i'th training data point, $P(Y = y^{(i)}|\theta, x^{(i)}) = \frac{1}{1+e^{-y^{(i)}\theta^T x^{(i)}}}$
- The **likelihood** $P(d|\theta)$ of the observing all m training data points d is therefore:

$$P(d|\theta) = \prod_{i=1}^{m} \frac{1}{1 + e^{-y^{(i)}\theta^T x^{(i)}}}$$

  (we multiply because the observations are independent)

Probabilistic Interpretation: Logistic Regression

- The **likelihood** $P(d|\theta)$ of the training data d is :

$$P(d|\theta) = \prod_{i=1}^{m} \frac{1}{1 + e^{-y^{(i)}\theta^T x^{(i)}}}$$

- Taking logs: $\log P(d|\theta) = \sum_{i=1}^{m} \log \frac{1}{1+e^{-y^{(i)}\theta^T x^{(i)}}}$
- Maximising $\log P(d|\theta)$ is the same as minimising $-\log P(d|\theta)$
- And $-\log(1/z) = \log(z)$, so ...
- ... the maximum likelihood estimate of θ minimises:

$$-\sum_{i=1}^{m} \log \frac{1}{1 + e^{-y^{(i)}\theta^T x^{(i)}}} = \sum_{i=1}^{m} \log(1 + e^{-y^{(i)}\theta^T x^{(i)}})$$

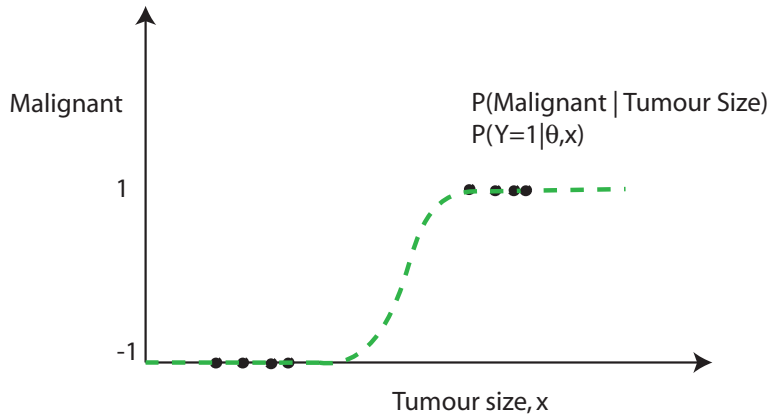
i.e. we've arrived at the logistic regression cost function

Probabilistic Interpretation: Logistic Regression

- The probabilistic formulation of logistic regression provides us with a new insight:

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

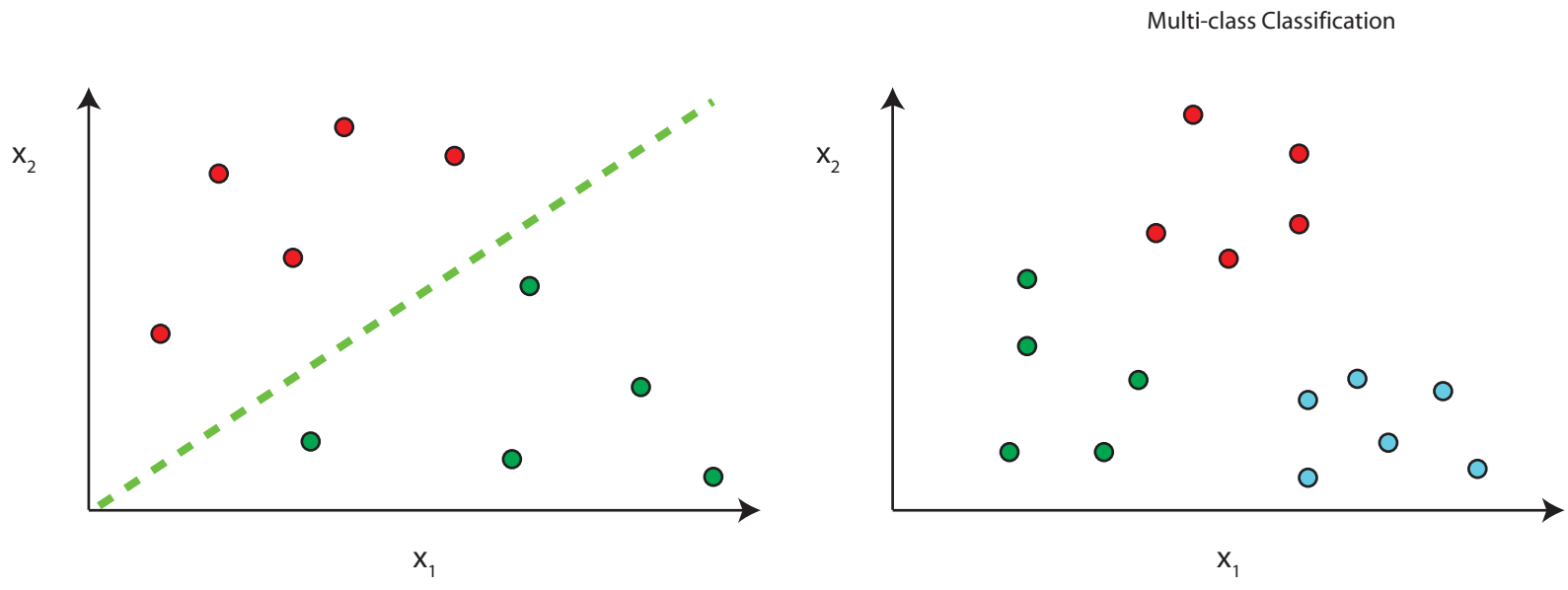
- So in addition to prediction $h_\theta(x) = sign(\theta^T x)$ we also have an estmate of our confidence in the prediction, namely $\frac{1}{1+e^{-y\theta^T x}}$.



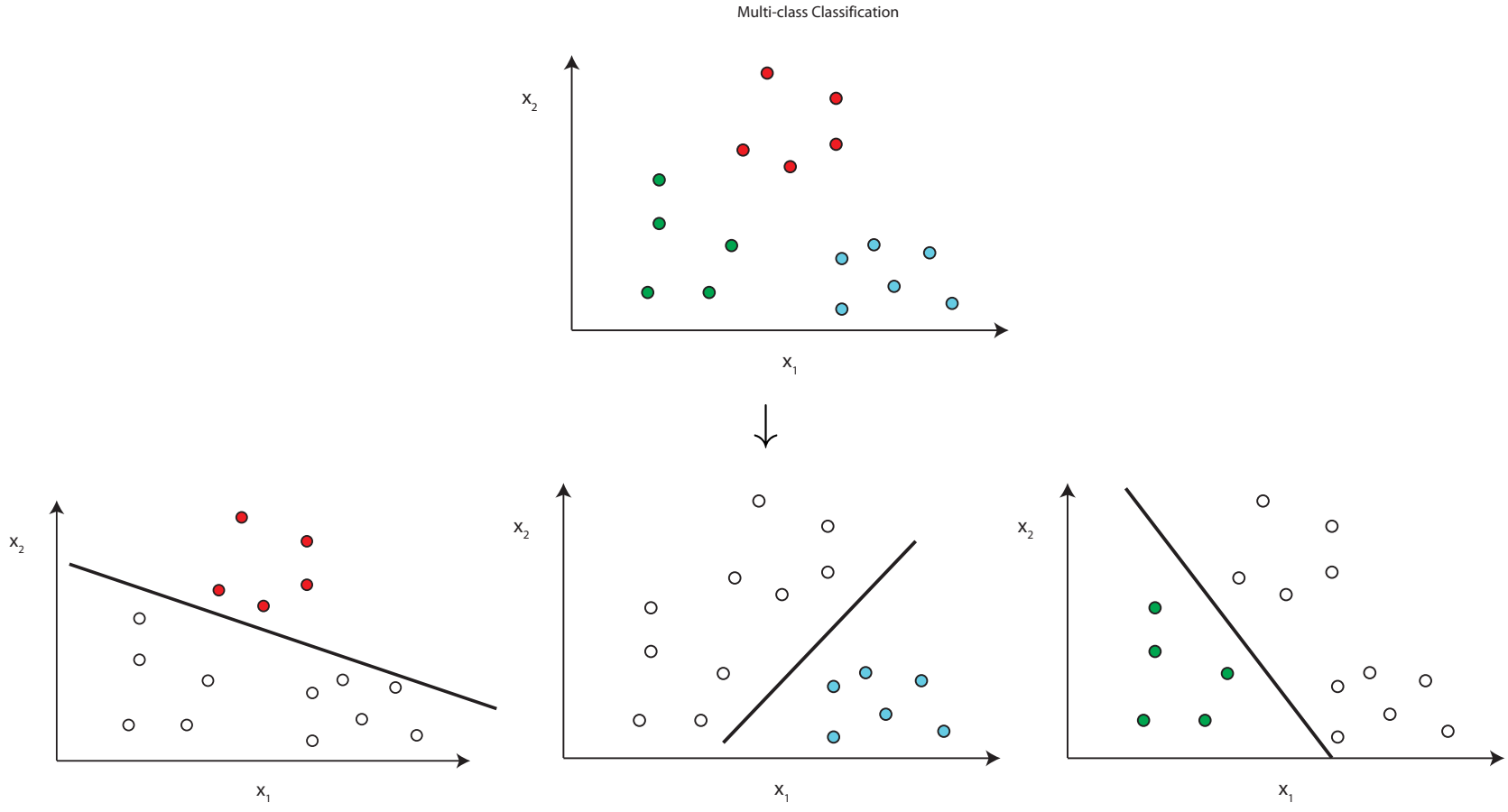
- When $\frac{1}{1+e^{-y\theta^T x}}$ is close to 1 or to 0, then we are confident in our prediction but when $\frac{1}{1+e^{-y\theta^T x}}$ is between 0 and 1 then we are less confident.

What If We Have More Than Two Classes ?

- Examples:
  - Email folder tagging: work, friends, family, hobby
  - Weather, sunny, cloudy, rain, snow
  - Given where I live in Dublin, predict which political party I'll vote for.
- Now y="output" variable/"target" variable takes values 0,1,2,.... E.g. $y = 0$ if sunny, $y = 1$ if cloudy, $y = 2$ if rain etc.



What If We Have More Than Two Classes ?

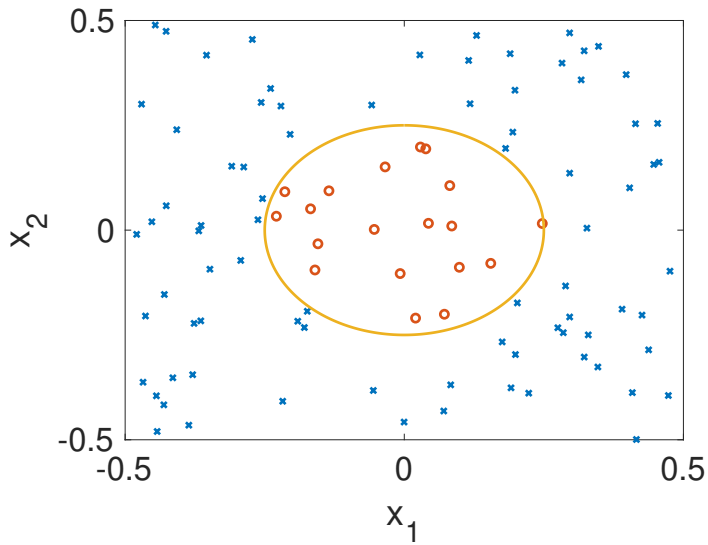


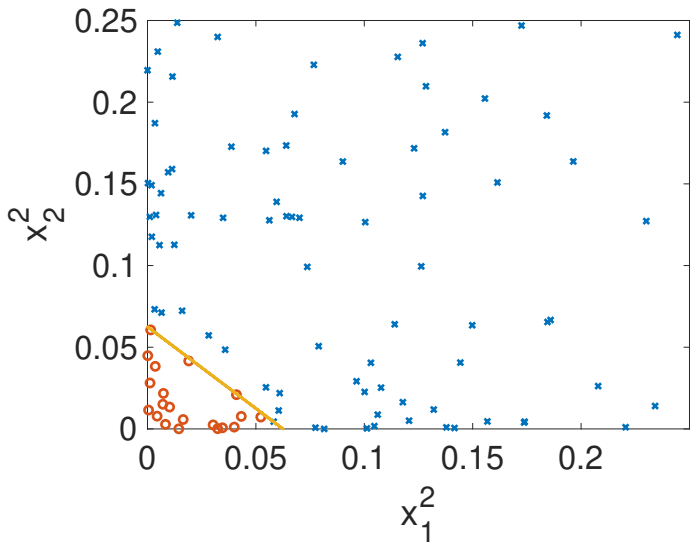
- Train a classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$. Training data: re-label data as $y = -1$ when $y \neq i$ and as $y = 1$ when $y = i$, so we're back to a binary classification task.

Separating by Nonlinear Curves: Choosing Features

Example:

- We have training with two features x_1, x_2 and ± 1 labelled outputs.
- Suppose when we plot our training it is not linearly separable but rather has a circle separating the two classes
- Recall that the equation for a circle of radius R is $x_1^2 + x_2^2 = R^R$ ...
- ... so change our features to be x_1^2, x_2^2
- Using this new feature vector our hypothesis can be rewritten as $h_\theta = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2$, so can directly apply all the ideas we've just discussed.





Confidence Intervals: Model Parameters

We would like to get a rough idea of how sensitive our model (and its predictions) are to the training data. We can use bootstrapping for this. For example:

- Construct a new set of training data by sampling with replacement from the original training data
- Train model on this new data i.e. select the parameter values which minimise the cost function for this data.
- Repeat the above two steps to obtain a set of parameter values. Use that to estimate the distribution of the parameter values, and so the confidence interval i.e. how much we expect the parameter values to vary

Confidence Intervals: Prediction Accuracy

We can also use a variation on this to estimate a confidence interval for the model predictions. For example:

- Randomly split the training data into a training set and a test set e.g. 80% used for training and 20% for testing
- Train model on the training data i.e. select the parameter values which minimise the cost function for this data.
- Evaluate the accuracy of the model predictions using the test data e.g calculate the mean prediction accuracy.
- Repeat the above three steps to obtain a set of prediction errors. Use that to estimate the distribution and so the confidence interval of the prediction error.