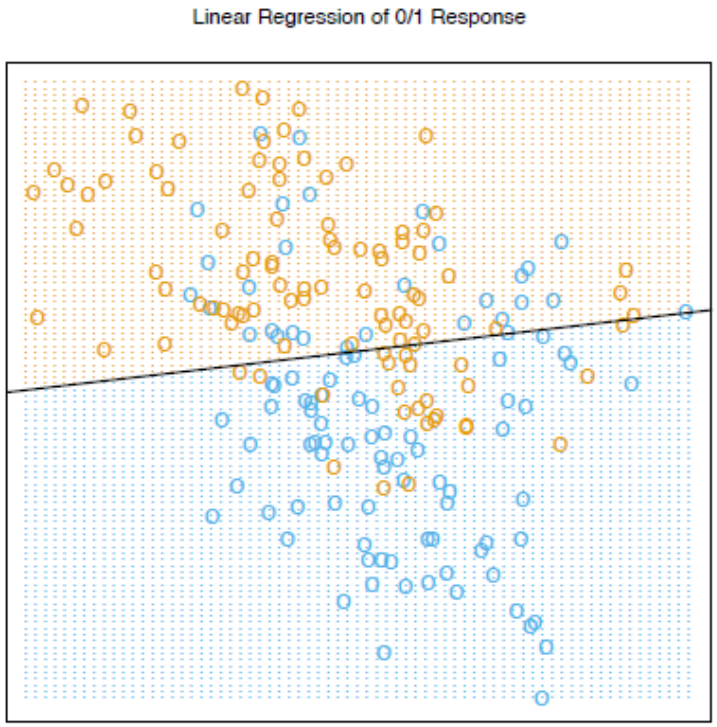
Linear Classification/
Logistic Regression/

Linear models for classification



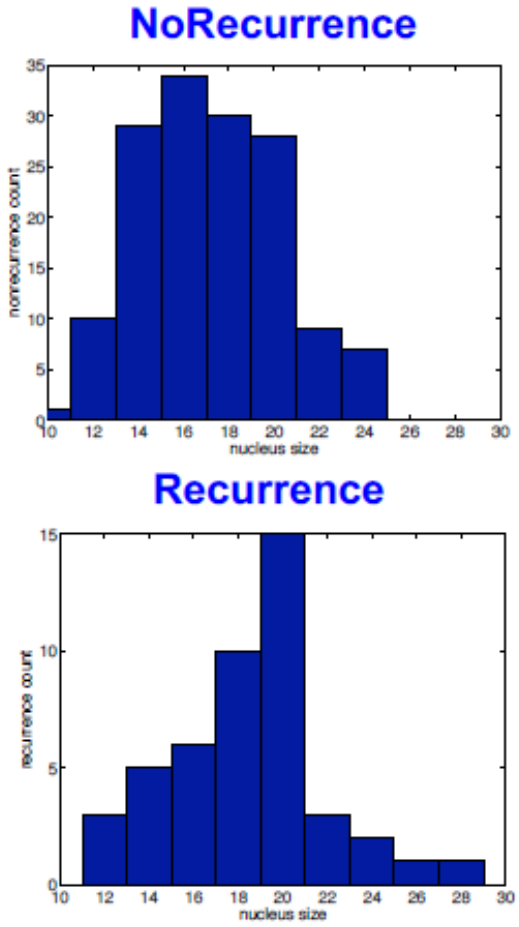
• The classes are coded as a binary variable (BLUE = 0, ORANGE = 1), and then fit by linear regression.

• The line is the decision boundary defined by $w^T x = 0.5$.

• The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.

Simple example

- Given “nucleus size”, predict cancer recurrence.
- Univariate input: X = nucleus size.
- Binary output: Y = {NoRecurrence = 0; Recurrence = 1}
- Try: Minimize the least-square error.



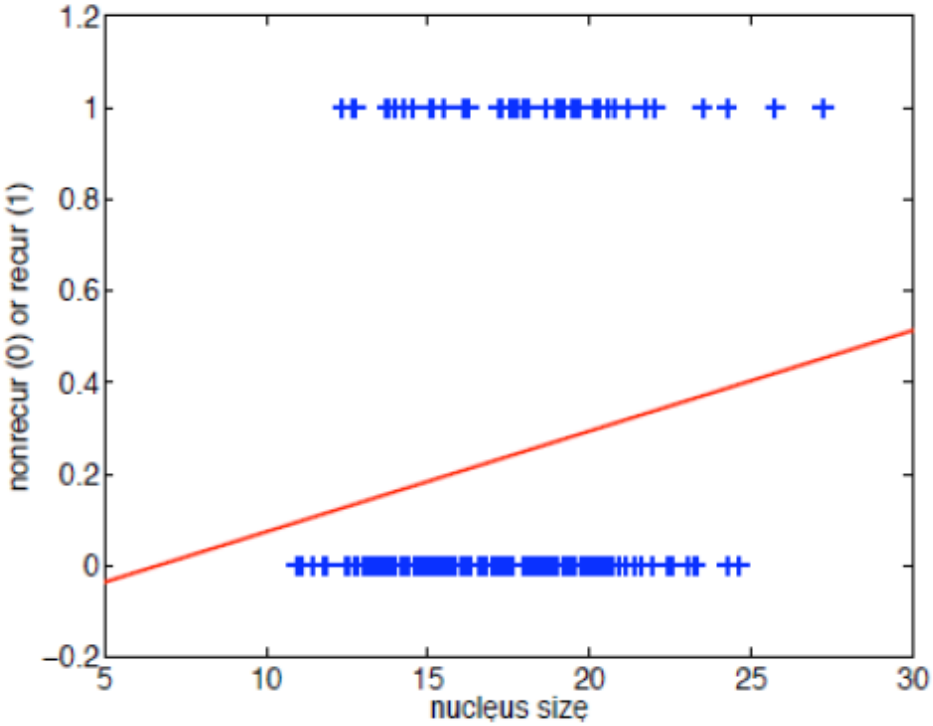
Predicting a class from linear regression

- How to get a binary output?
  1. Threshold the output:

     { $y <= t$ for NoRecurrence,

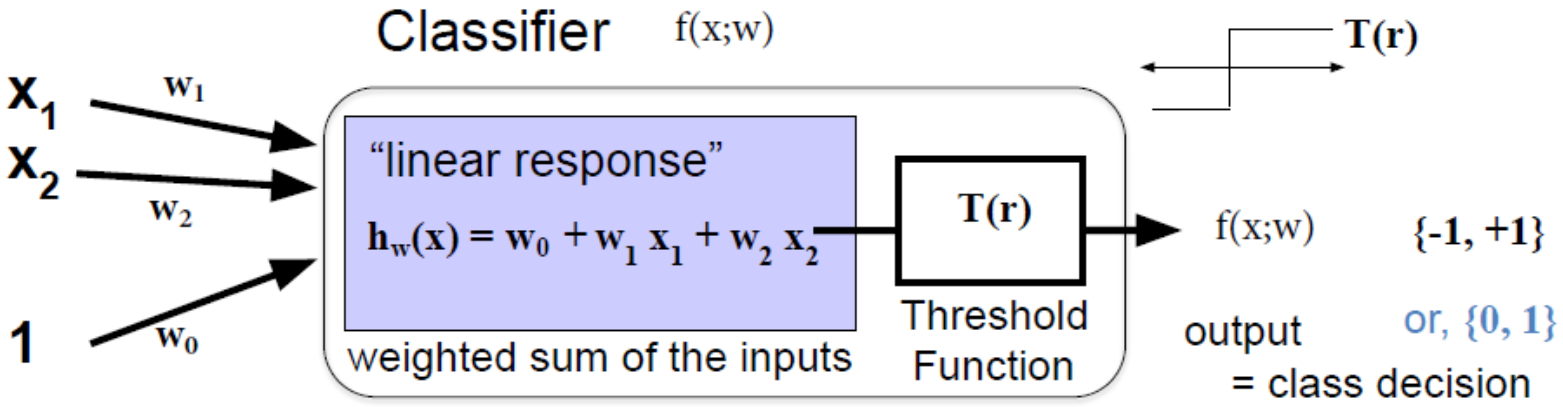
     $y > t$ for Recurrence}
  2. Interpret output as probability:

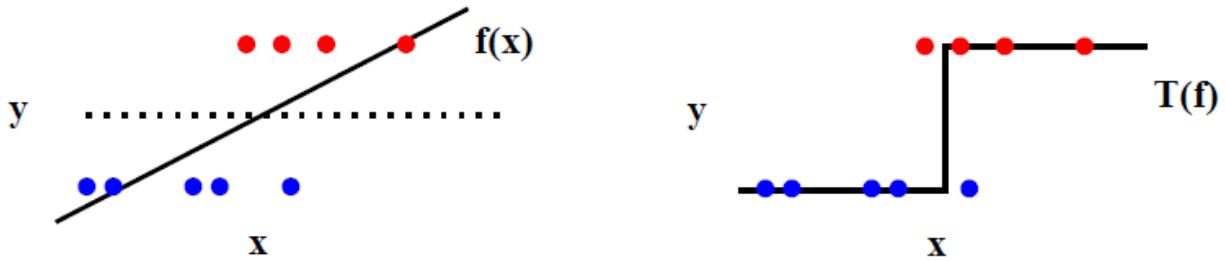
     $y = Pr$ (Recurrence)



Perceptron

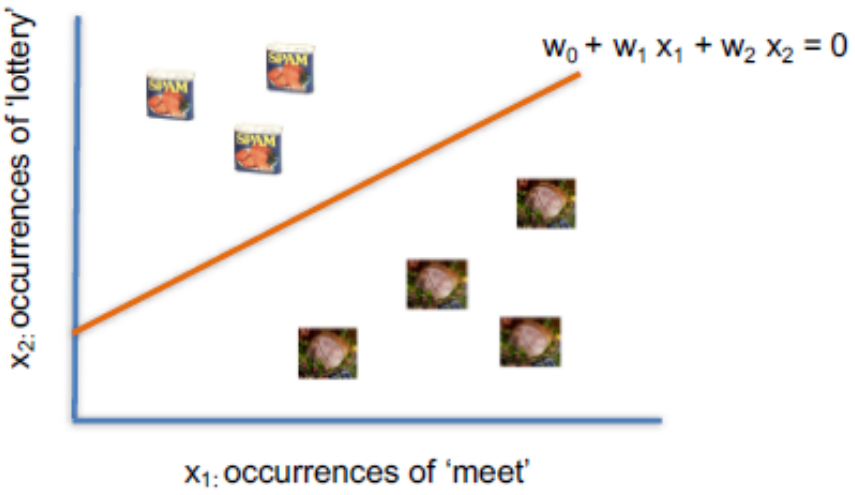
- Perceptron takes weighted linear combination of input features and pass it through a thresholding function which outputs 1 or 0.
- The sign of $w^T x$ tells us which side of the plane $w^T x=0$, the point x lies on.
- Thus by taking threshold as 0, perceptron classifies data based on which side of the plane the new point lies on.





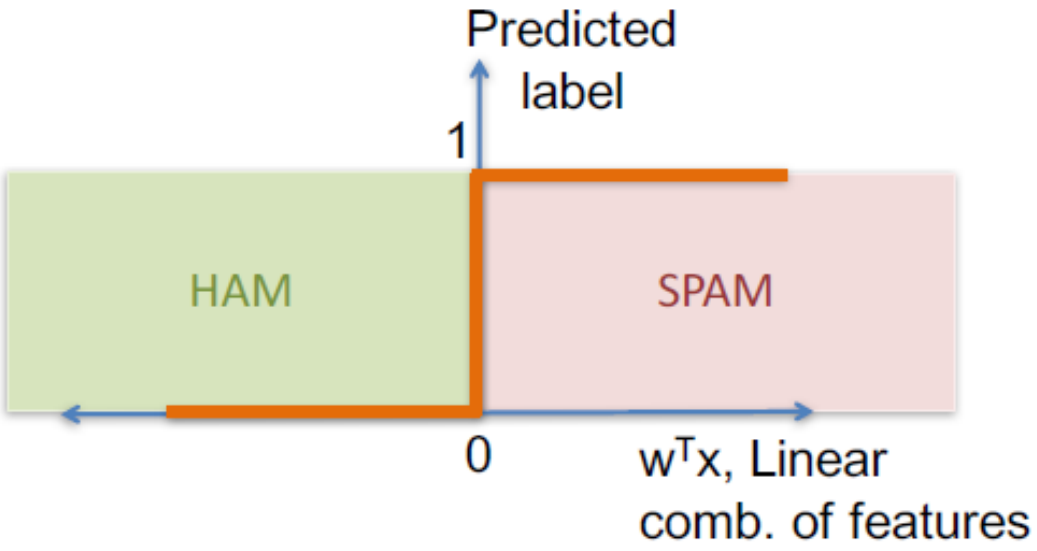
Example

- $x_1 = \#$ of times 'meet' appears in an email
- $x_2 = \#$ of times 'lottery' appears in an email
- Define feature vector $\mathbf{x} = [1, x_1, x_2]$
- Learn the decision boundary $w_0 + w_1 x_1 + w_2 x_2 = 0$ such that
  - If $\mathbf{w}^\top \mathbf{x} \geq 0$ declare $y = 1$ (spam)
  - If $\mathbf{w}^\top \mathbf{x} < 0$ declare $y = 0$ (ham)



Visualizing a Linear Classfier

- $x_1 =$ # of times 'lottery' appears in an email
- $x_2 =$ # of times 'meet' appears in an email
- Define feature vector $\mathbf{x} = [1, x_1, x_2]$
- Learn the decision boundary $w_0 + w_1 x_1 + w_2 x_2 = 0$ such that
  - If $\mathbf{w}^\top \mathbf{x} \geq 0$ declare $y = 1$ (spam)
  - If $\mathbf{w}^\top \mathbf{x} < 0$ declare $y = 0$ (ham)



$y = 1$ for spam, $y = 0$ for ham

Suppose you see the following email:

```
CONGRATULATIONS!!  Your email address have won you the
lottery sum of US$2,500,000.00 USD to claim your prize,
   contact your office agent (Athur walter) via email
    claims2155@yahoo.com.hk or call +44 704 575 1113
```

Keywords are [lottery, prize, office, email]
The given weight vector is $\mathbf{w} = [0.3, 0.3, -0.1, -0.04]^\top$

Will we predict that the email is spam or ham?

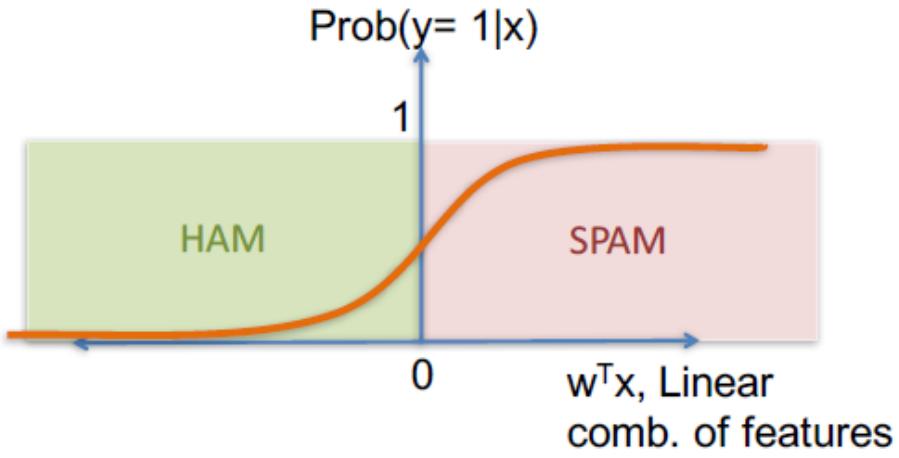
$$\mathbf{x} = [1, 1, 1, 2]^\top$$

$$\mathbf{w}^\top \mathbf{x} = 0.3 * 1 + 0.3 * 1 - 0.1 * 1 - 0.04 * 2 = 0.42 > 0$$

so we predict spam!

Intuition : Logistic Regression

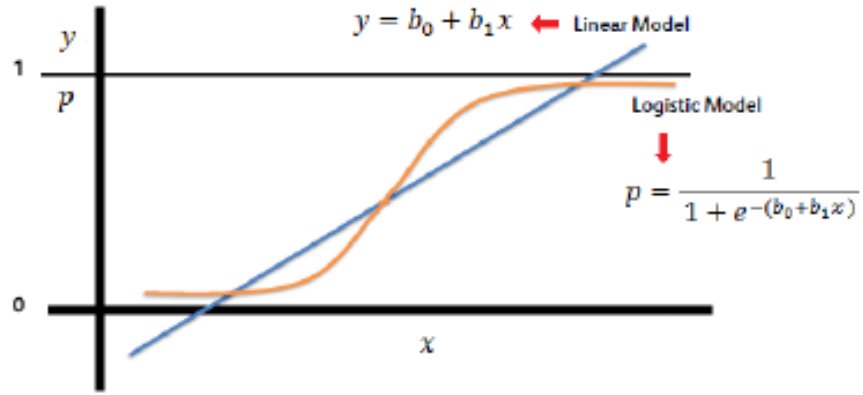
- Suppose we want to output the probability of an email being spam/ham instead of just 0 or 1
- This gives information about the confidence in the decision
- Use a function $\sigma(\mathbf{w}^\top \mathbf{x})$ that maps $\mathbf{w}^\top \mathbf{x}$ to a value between 0 and 1



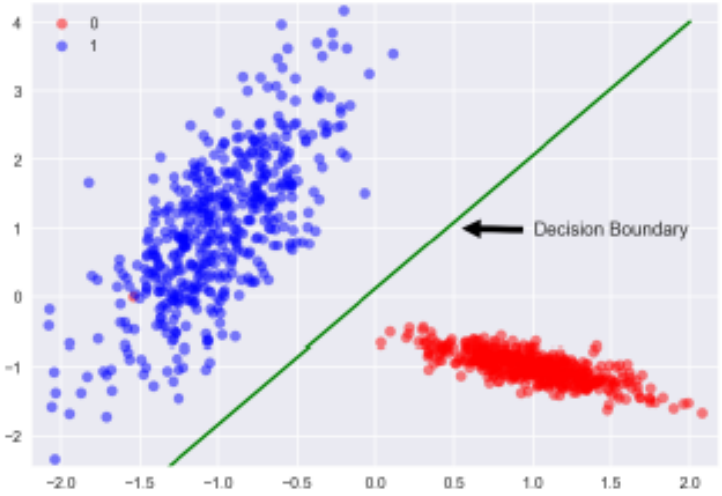
Probability that predicted label is 1 (spam)

Key Problem: Finding optimal weights **w** that accurately predict this probability for a new email

Sigmoid function returns values in [0,1]



Decision boundary is linear: linear classifier



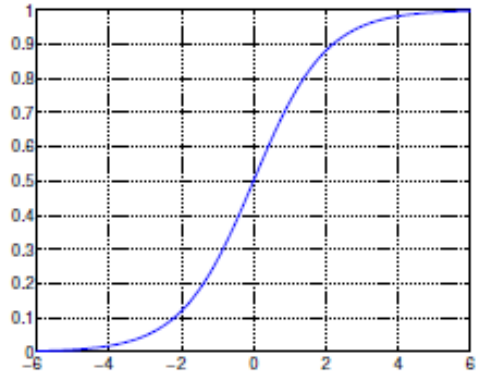
Why the Sigmoid Function?

What does it look like?

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

where

$$a = \boldsymbol{w}^\top \boldsymbol{x}$$



Sigmoid properties

- Bounded between 0 and 1 ← thus, interpretable as probability
- Monotonically increasing ← thus, usable to derive classification rules
  - $\sigma(a) \geq 0.5$, positive (classify as '1')
  - $\sigma(a) < 0.5$, negative (classify as '0')
- Nice computational properties ← as we will see soon

Suppose you see the following email:

```
CONGRATULATIONS!! Your email address have won you the
lottery sum of US$2,500,000.00 USD to claim your prize,
  contact your office agent (Athur walter) via email
   claims2155@yahoo.com.hk or call +44 704 575 1113
```

Keywords are [lottery, prize, office, email]
The given weight vector is $\mathbf{w} = [0.3, 0.3, -0.1, -0.04]^\top$

What is the probability that the email is spam?

$$\mathbf{x} = [1, 1, 1, 2]^\top$$

$$\mathbf{w}^\top \mathbf{x} = 0.3 * 1 + 0.3 * 1 - 0.1 * 1 - 0.04 * 2 = 0.42 > 0$$

$$\Pr(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x}) = \frac{1}{1 + e^{-0.42}} = 0.603$$

Logistic Regression Objective Function

- Can't just use squared loss as in linear regression:

$$J(\mathrm{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\mathrm{w}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

  - Using the logistic regression model

$$h_{\mathrm{w}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\mathrm{w}^{\mathsf{T}}\boldsymbol{x}}}$$

    results in a non-convex optimization

Deriving the Cost Function via Maximum Likelihood Estimation

- The intuition behind it is to find the parameters of a model that maximize the probability of observing the given data.

- Product of the individual likelihoods for each data point when the data points are assumed to be independent.

$$l(\boldsymbol{\theta}) = \prod_{i=1}^{n} p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

- Likelihood of data is given by:

- So, looking for the $\boldsymbol{\theta}$ that maximizes the likelihood

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

Deriving the Cost Function via Maximum Likelihood Estimation

- Can take the log without changing the solution:

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \log \prod_{i=1}^{n} p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

Deriving the Cost Function via Maximum Likelihood

Let us assume that

$$\begin{aligned} P(y = 1 \mid x; \theta) &= h_\theta(x) \\ P(y = 0 \mid x; \theta) &= 1 - h_\theta(x) \end{aligned}$$

Note that this can be written more compactly as

$$p(y \mid x; \theta) = (h_\theta(x))^y \, (1 - h_\theta(x))^{1-y}$$

Assuming that the m training examples were generated independently, we can then write down the likelihood of the parameters as

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X; \theta) \\ &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

Deriving the Cost Function via Maximum Likelihood

- Expand as follows:

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left[y^{(i)} \log p(y^{(i)} = 1 \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}) + \left(1 - y^{(i)}\right) \log \left(1 - p(y^{(i)} = 1 \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})\right) \right]$$

- Substitute in model, and take negative to yield

Logistic regression objective:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$J(\boldsymbol{\theta}) = -\sum_{i=1}^{n} \left[y^{(i)} \log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1 - y^{(i)}\right) \log \left(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right) \right]$$

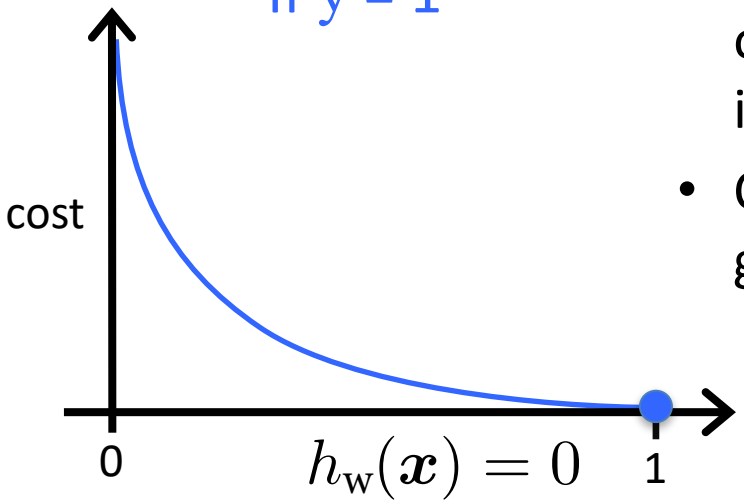
Intuition Behind the Objective

$$\text{cost}\,(h_{\mathrm{w}}(\boldsymbol{x}), y\,) = \begin{cases} -\log(h_{\mathrm{w}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1- \ h_{\mathrm{w}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

When actual label **y** is 1

- Cost = 0 if prediction is correct
- As $h_{\mathrm{w}}(\boldsymbol{x}) \to 0, \text{cost} \to \infty$

If y = 1



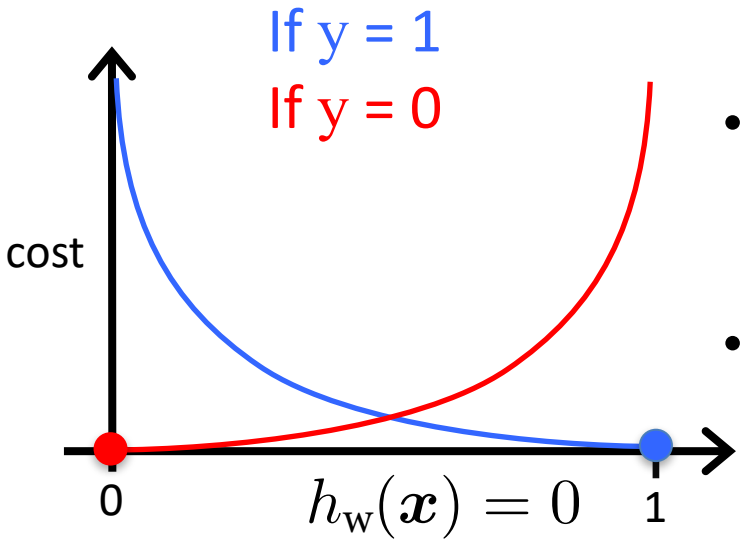
- meaning the predicted probability of a positive outcome is very low, the cost for that example increases significantly.
- Captures intuition that larger mistakes should get larger penalties
  - e.g., predict $h_{\mathrm{w}}(\boldsymbol{x}) = 0$, but y = 1

Intuition Behind the Objective

$$\text{cost}\left(h_{\mathrm{w}}(\boldsymbol{x}), y\right) = \begin{cases} -\log(h_{\mathrm{w}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\mathrm{w}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

If y = 0

- Cost = 0 if prediction is correct
- As $(1 - h_{\mathrm{w}}(\boldsymbol{x})) \to 0, \text{cost} \to \infty$



- meaning the predicted probability of a negative outcome is very low), the cost for that example increases significantly.
- Captures intuition that larger mistakes should get larger penalties

Gradient Descent for Logistic Regression

$$J_{\text{reg}}(\mathrm{w}) = -\sum_{i=1}^{n}\left[y^{(i)} \log h_{\mathrm{w}}(\boldsymbol{x}^{(i)}) + \left(1 - y^{(i)}\right) \log\left(1 - h_{\mathrm{w}}(\boldsymbol{x}^{(i)})\right)\right]$$

Want $\min_{\mathrm{w}} J(\mathrm{w})$

- Initialize w
- Repeat until convergence

$$\mathrm{w}_j = \mathrm{w}_j - \alpha \frac{\partial}{\partial \mathrm{w}j} J(\mathrm{w})$$

simultaneous update
for j = 0 ... d

Use the natural logarithm (ln = $\log_e$) to cancel with the exp() in $h_{\mathrm{w}}(\boldsymbol{x})$

Gradient Descent for Logistic Regression

$$J_{\text{reg}}(\mathrm{w}) = -\sum_{i=1}^{n}\left[y^{(i)}\log h_{\mathrm{w}}(\boldsymbol{x}^{(i)}) + \left(1-y^{(i)}\right)\log\left(1-h_{\mathrm{w}}(\boldsymbol{x}^{(i)})\right)\right]$$

Want $\min_{\mathrm{w}} J(\mathrm{w})$

- Initialize w
- Repeat until convergence (simultaneous update for j = 0 ... d)

$$\mathrm{w}_0 = \mathrm{w}_0 - \alpha\sum_{i=1}^{n}\left(h_{\mathrm{w}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)$$

$$\mathrm{w}_j = \mathrm{w}_j - \alpha\left[\sum_{i=1}^{n}\left(h_{\mathrm{w}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)}\right]$$

Gradient Descent for Logistic Regression

- Initialize w
- Repeat until convergence (simultaneous update for j = 0 ... d)

$$\mathrm{w}_0 \leftarrow \mathrm{w}_0 - \alpha \sum_{i=1}^{n} \left(h_{\mathrm{w}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)$$

$$\mathrm{w}_j \leftarrow \mathrm{w}_j - \alpha \left[\sum_{i=1}^{n} \left(h_{\mathrm{w}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)} \right]$$

This looks IDENTICAL to linear regression!!!

- Ignoring the $1/\mathrm{n}$ constant
- However, the form of the model is very different:

$$h_{\mathrm{w}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\mathrm{w}^{\mathsf{T}} \boldsymbol{x}}}$$

Gradient Descent for Logistic Regression

$$J_{\text{reg}}(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\left[y^{(i)}\log h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) + \left(1-y^{(i)}\right)\log\left(1-h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})\right)\right] + \lambda\|\boldsymbol{\theta}_{[1:d]}\|_2^2$$

Want $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

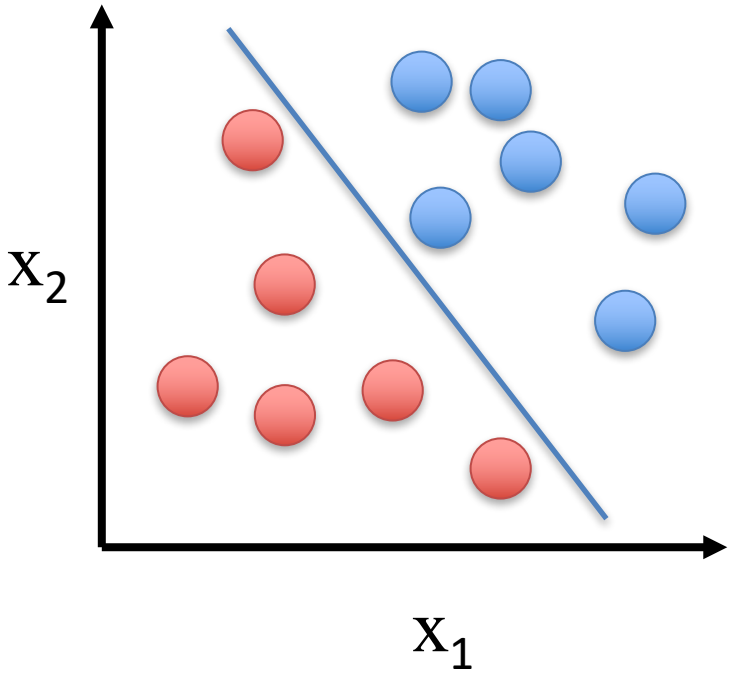
- Initialize $\boldsymbol{\theta}$
- Repeat until convergence (simultaneous update for j = 0 ... d)

$$\theta_0 \quad \theta_0 - \alpha\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)$$

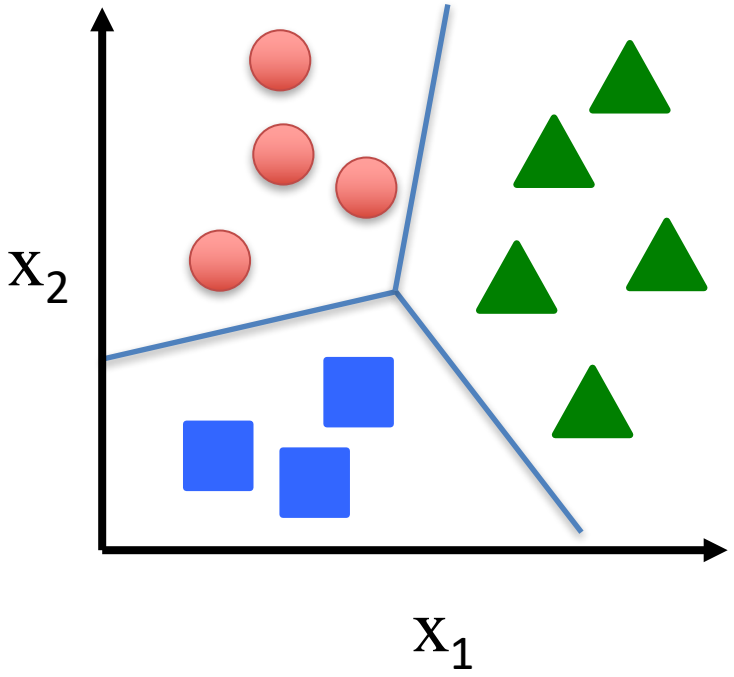
$$\theta_j \quad \theta_j - \alpha\left[\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)} - \frac{\lambda}{n}\theta_j\right]$$

Multi-Class Classification

Binary classification:



Multi-class classification:

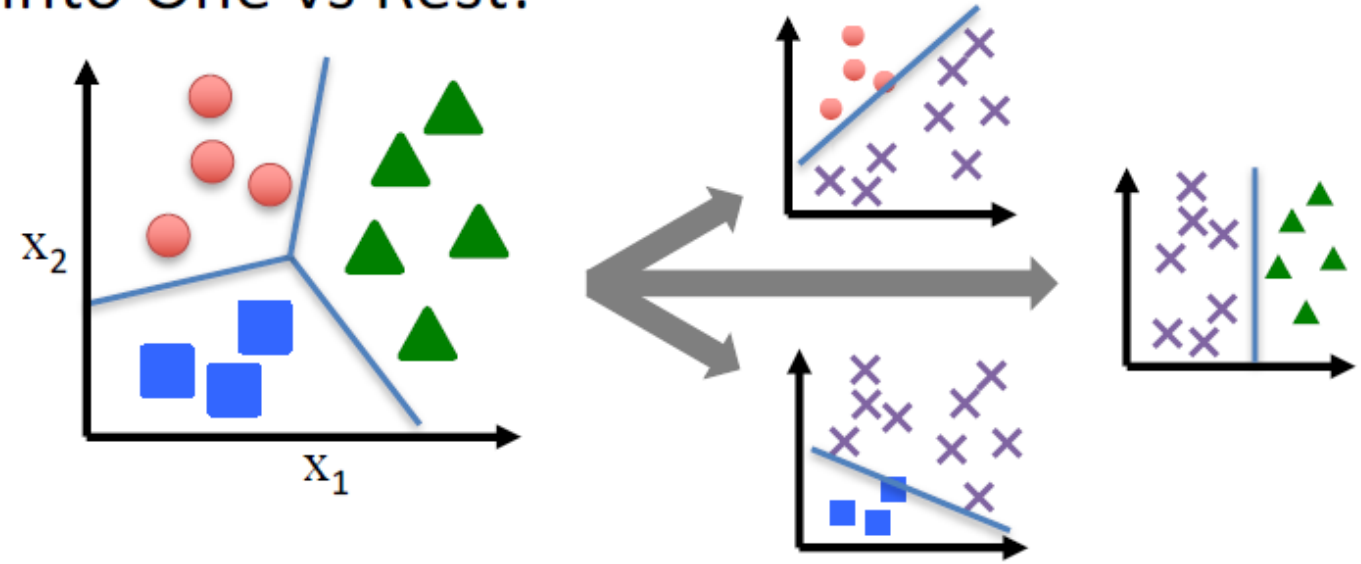


Disease diagnosis: healthy / cold / flu / pneumonia

Object classification: desk / chair / monitor / bookcase

Multi-Class Logistic Regression

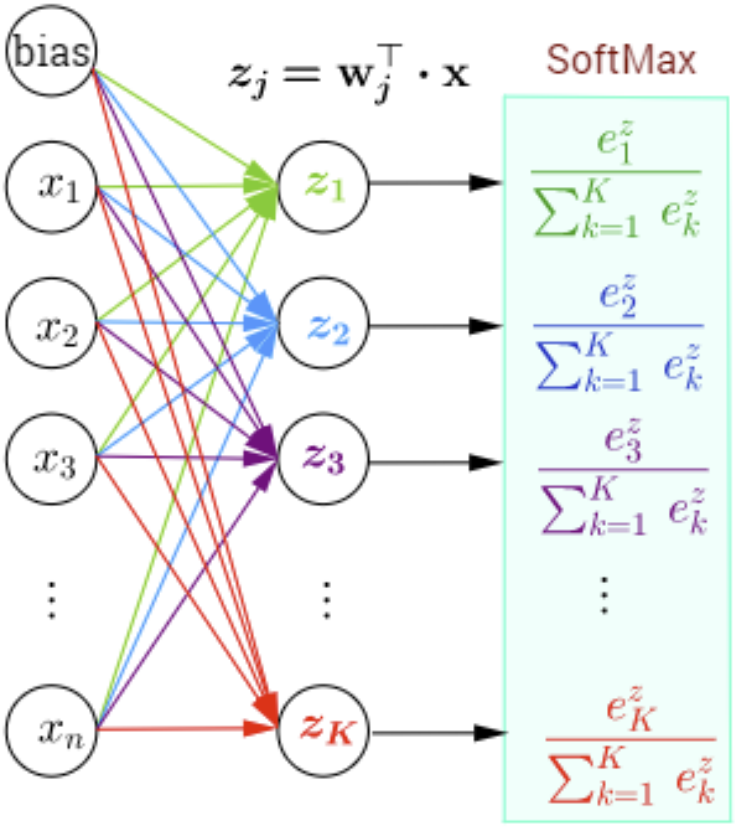
Split into One vs Rest:



- Train a logistic regression classifier for each class i to predict the probability that $y = i$ with

$$h_c(\boldsymbol{x}) = \frac{\exp(\boldsymbol{\theta}_c^\top \boldsymbol{x})}{\sum_{c=1}^{C} \exp(\boldsymbol{\theta}_c^\top \boldsymbol{x})}$$

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$



Multi-Class Logistic Regression

- For 2 classes:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x})} = \frac{\exp(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x})}{1+\exp(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x})}$$

- For C classes {1, ..., C}:

$$p(y = c \mid \boldsymbol{x}; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_C) = \frac{\exp(\boldsymbol{\theta}_c^{\mathsf{T}}\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{\theta}_c^{\mathsf{T}}\boldsymbol{x})}$$

  - Called the **softmax** function

Implementing Multi-Class Logistic Regression

- Use $h_c(\boldsymbol{x}) = \dfrac{\exp(\boldsymbol{\theta}_c^\top \boldsymbol{x})}{\sum_{c=1}^{C} \exp(\boldsymbol{\theta}_c^\top \boldsymbol{x})}$ as the model for class c

- Gradient descent simultaneously updates all parameters for all models
  - Same derivative as before, just with the above $\mathrm{h_c(x)}$

- Predict class label as the most probable label

$$\max_c h_c(\boldsymbol{x})$$