Loss function of the Logistic Regression

How do we get the word weights?

What if we learn them from the data?

$$y = w0 + w_1\varphi_1(x^1) + w_2\varphi_2(x^2) + \ldots + w_D\varphi_D(x^D)$$

$\varphi_1(x)$	$\varphi_2(x)$	$\varphi_3(x)$	$\varphi_4(x)$	$\varphi_5(x)$	$\varphi_6(x)$	$\varphi_7(x)$	$\varphi_8(x)$	$\varphi_9(x)$
sushi	**was**	**great**	**the**	**food**	**awesome**	**but**	**service**	**terrible**
1	3	1	2	1	1	1	1	1

Word	**Weight**
sushi	w_1
was	w_2
great	w_3
the	w_4
food	w_5
awesome	w_6
but	w_7
service	w_8
terrible	w_9

In linear regression we learnt the weights for each feature.

Can we do something similar here?

Connecting Score & Probability

$$Score(x_i) = w^T \varphi(x^{(i)})$$

$-\infty$ $\quad \hat{y}_i = -1 \quad$ 0 $\quad \hat{y}_i = +1 \quad$ ∞

Very sure
$\hat{y}_i = -1$

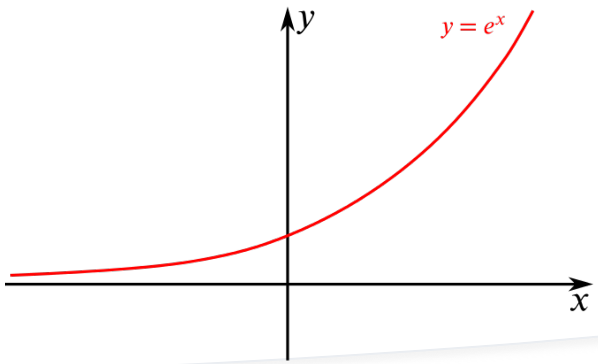
Not sure if
$\hat{y}_i = -1 \; or \; \hat{y}_i = +1$

Very sure
$\hat{y}_i = +1$

$\hat{P}(y_i = +1|x_i) = 0$ $\quad \hat{P}(y_i = +1|x_i) = 0.5$ $\quad \hat{P}(y_i = +1|x_i) = 1$

0 $\quad \hat{P}(y = +1\,|x)$ $\quad 1$

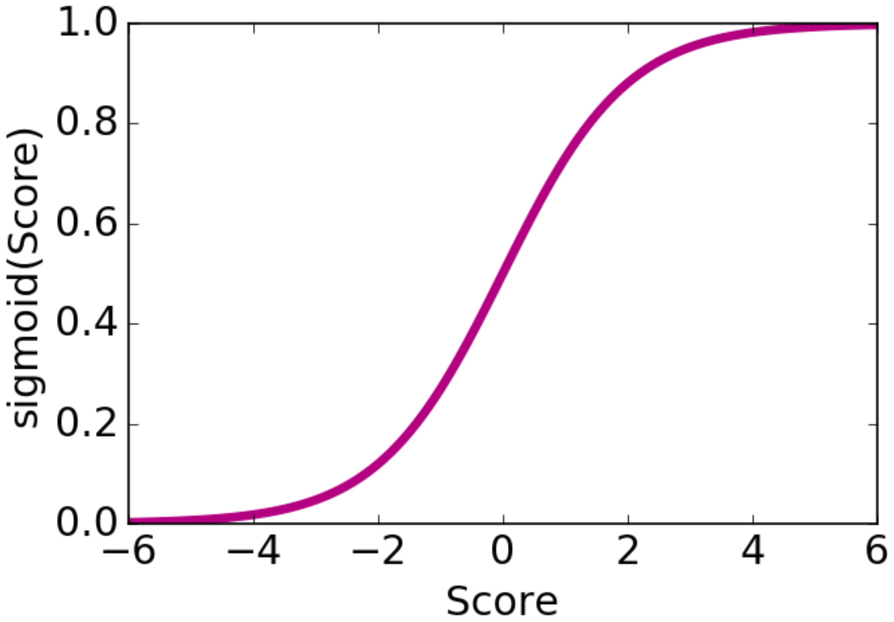
Logistic Function



Want: a function that takes numbers arbitrarily large/small and maps them between 0 and 1.

$$sigmoid(Score(x)) = \frac{1}{1 + e^{-Score(x)}}$$

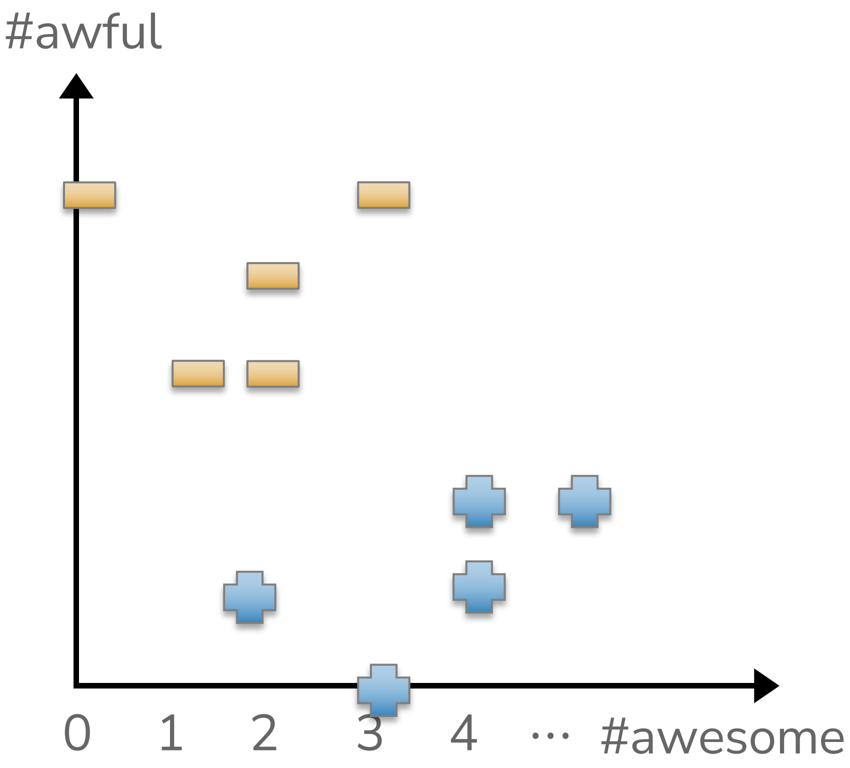
$Score(x)$	$sigmoid(Score(x))$
$-\infty$	$\frac{1}{1+e^{-(-\infty)}} = 0$
-2	
0	$\frac{1}{1+e^{-(0)}} = 0.5$
2	
∞	$\frac{1}{1+e^{-(\infty)}} = 1$



Learn $\hat{w}$

Now that we have our new model, we will talk about how to choose $\hat{w}$ to be the "best fit".

The choice of w affects how likely seeing our dataset is



$$\ell(w) = \prod_{i}^{n} P(y^{(i)} | x^{(i)}, w)$$

$$P\left(y^{(i)} = +1 \middle| x^{(i)}, w\right) = \frac{1}{1 + e^{-w^T h(x^{(i)})}}$$

$$P\left(y^{(i)} = 0 \middle| x^{(i)}, w\right) = \frac{e^{-w^T h(x^{(i)})}}{1 + e^{-w^T h(x^{(i)})}}$$

General Joint Likelihood, L(w)

- From the joint probability of all labels, we have

$L(w) = P(y_1, y_2, \ldots, y_n \mid x_1, x_2, \ldots, x_n; w)$

y_i are the labels

x_i are the features

- The log-likelihood is then

$\ell(w) = \log P(y_1, y_2, \ldots, y_n \mid x_1, x_2, \ldots, x_n; w)$

Independence Assumption (i.i.d.)

- Assume now that each (x_i, y_i) pair is independent and drawn from the same distribution.

- Then the joint probability becomes

$$P(y_1, \ldots, y_n \mid x_1, \ldots, x_n; w) = \prod_i P(y_i \mid x_i; w)$$

Formula for each individual conditional probability and application to a Bernoulli variable

- σ is the Sigmoid function.

  $P(y_i = 1 \mid x_i; w) = \sigma(w^T x_i)$

  $P(y_i = 0 \mid x_i; w) = 1 - \sigma(w^T x_i)$

$$\sigma(w^T x_i) = \frac{1}{1 + e^{-w^T x_i}}$$

- For Bernoulli variable y_i, the values that it can take are {0,1}. With this, we can rewrite two expressions above in one condensed form as

  $P(y_i \mid x_i; w) = [\sigma(w^T x_i)]^{y^i} [1 - \sigma(w^T x_i)]^{1-y^i}$

Applying the Bernoulli distribution

- So we have $P(y_i \mid x_i; w) = [\sigma(w^T x_i)]^{y^i} [1 - \sigma(w^T x_i)]^{1-y^i}$

- And now for all the labels
  $L(w) = \prod_i P(y_i \mid x_i; w) = \prod_i [\sigma(w^T x_i)]^{y^i} [1 - \sigma(w^T x_i)]^{1-y^i}$

- Take the log and substitute into the log-likelihood:
  $\log(L(w)) = \ell(w) = \Sigma_i \log([\sigma(w^T x_i)]^{y^i} [1 - \sigma(w^T x_i)]^{1-y^i})$

- Using log properties ($\log(ab) = \log a + \log b$, $\log(a^b) = b \log a$):
  $\ell(w) = \Sigma_i [y_i \log \sigma(w^T x_i) + (1 - y_i) \log(1 - \sigma(w^T x_i))]$

Loss Function

taking the log does not change the location of the maximum.

Find the w that maximizes the likelihood

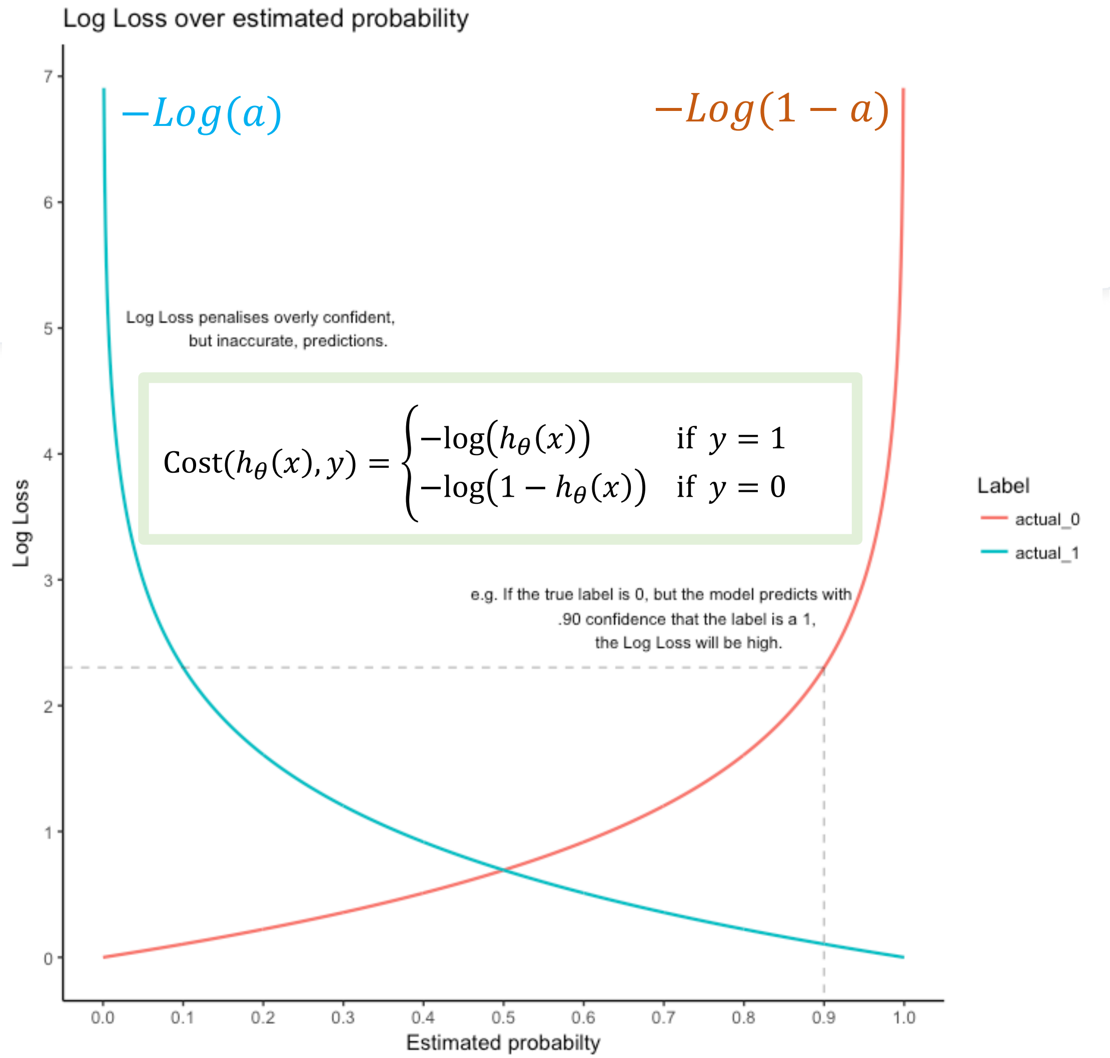
$$\hat{w} = \underset{w}{\operatorname{argmax}}\, \ell(w) = \underset{w}{\operatorname{argmax}} \prod_{i=1}^{n} P(y_i|x_i, w)$$

Generally, we maximize the log-likelihood which looks like

$$\hat{w} = \underset{w}{\operatorname{argmax}}\, \ell(w) = \underset{w}{\operatorname{argmax}}\, \log(\ell(w)) = \underset{w}{\operatorname{argmax}} \sum_{i=1}^{n} \log(P(y_i|x_i, w))$$

Also commonly written by separating out positive/negative terms

$$LogLoss = -\frac{1}{n}\sum_{i=0}^{n} \underbrace{[y_i log(\hat{y}_i)}_{For\ Positive\ terms} + \underbrace{(1-y_i)\log(1-\hat{y}_i)]}_{For\ Negative\ terms}$$



Connecting Log-Likelihood (MLE) to Cross-Entropy (ML)

- Log-likelihood under Bernoulli model:
  $\ell(w) = \Sigma_i [\, y_i \log \sigma(w^T x_i) + (1-y_i) \log(1-\sigma(w^T x_i)) \,]$

- Negative log-likelihood (the ML loss):
  $\mathcal{L}(w) = -\ell(w) = -\Sigma_i [\, \underbrace{y_i}_{p} \underbrace{\log \sigma(w^T x_i)}_{q} + \overbrace{(1-y_i)}^{1-p} \underbrace{\log(1-\sigma(w^T x_i))}_{1-q} \,]$

- Notice that the Negative log-likelihood is the cross-entropy with p and q:

Now, here we're comparing two distributions: p, q.

model

original

H(p, q) for one sample: $-[\, y_i \log \sigma(w^T x_i) + (1-y_i) \log(1-\sigma(w^T x_i)) \,]$

Therefore, minimizing cross-entropy is equivalent to minimizing $-\ell(w)$ (which is equivalent to maximizing $\ell(w)$.

$$H(p,q) = \boxed{-\frac{1}{n}} \sum_i \left[y_i \log \sigma(w^T x_i) + (1-y_i) \log(1-\sigma(w^T x_i)) \right]$$

The total cross-entropy

added for computing the mean

Finding the weights w

The optimal weights w are the ones which (as always) minimize the average error/objective function:

$$\min_w \mathcal{L} = \min_w \frac{1}{n}\sum_{i=1}^{n}\left[p_i(e=0)\log\left(\frac{1}{\frac{1}{1+e^{-x_i^T w}}}\right) + p_i(e=1)\log\left(\frac{1}{1-\frac{1}{1+e^{-x_i^T w}}}\right)\right]$$

Since this is a relatively complicated equation, we must find the minimum using **Gradient Descent**. Remember, if we want to find the optimal weight vector $w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$, we iteratively update:

Remember this

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}_{t+1} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}_{t} - \eta \frac{d\mathcal{L}}{dw}$$

- Once you find the optimal w using gradient descent, we can make future predictions using the sigmoid
- This is logistic regression, one of the most basic machine learning algorithms

Gradient of the Logistic Regression Objective

In order to conduct Gradient Descent, we obviously need the gradient, $\frac{d\mathcal{L}}{dw}$, we'll start by simplifying $\mathcal{L}$ as much as possible:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\left[p_i(e=0)\log\left(\frac{1}{\frac{1}{1+e^{-x_i^T w}}}\right) + p_i(e=1)\log\left(\frac{1}{1-\frac{1}{1+e^{-x_i^T w}}}\right)\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left[p_i(e=0)\log\left(1+e^{-x_i^T w}\right) + p_i(e=1)\log\left(\frac{1+e^{-x_i^T w}}{e^{-x_i^T w}}\right)\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left[(p_i(e=0)+p_i(e=1))\log\left(1+e^{-x_i^T w}\right) - p_i(e=1)\log\left(e^{-x_i^T w}\right)\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left[\log\left(1+e^{-x_i^T w}\right) + (1-p_i(e=0))x_i^T w\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left[\log\left(1+e^{-x_i^T w}\right) + \log\left(e^{x_i^T w}\right) - p_i(e=0)x_i^T w\right]$$

Gradient of the Logistic Regression Objective

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\left[\log\left(e^{x_i^T w}+1\right) - p_i(e=0)x_i^T w\right]$$

Now we are ready to take the derivative:

$$\frac{d\mathcal{L}}{dw} = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{x_i e^{x_i^T w}}{1+e^{x_i^T w}} - p_i(e=0)x_i\right]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left[\frac{e^{-x_i^T w}}{e^{-x_i^T w}}\frac{e^{x_i^T w}}{1+e^{x_i^T w}} - p_i(e=0)\right]x_i$$

remember this

$$= \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{e^{-x_i^T w}+1} - p_i(e=0)\right]x_i$$

- This is the derivative we will use in gradient descent

Decision Boundary

EXTRA THINGS NOT INCLUDED IN THE EXAM

The decision boundary is the set of x such that

$$\frac{1}{1+e^{-(w^T X)}} = 0.5$$

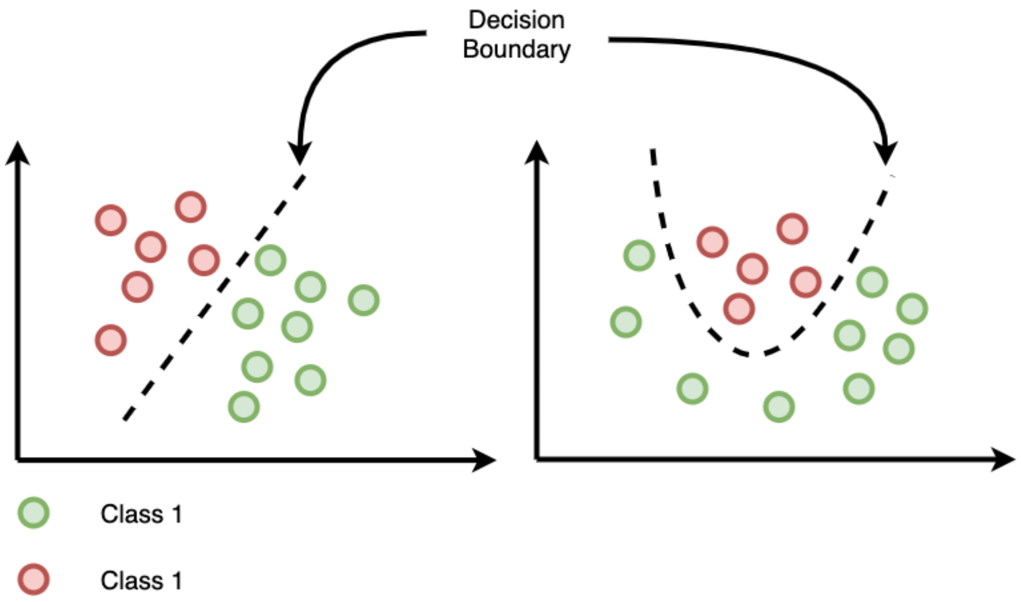
A little bit of algebra shows that this is equivalent to

$$1 = e^{-(w^T X)}$$

and, taking the natural log of both sides,

$$0 = -\sum_{i=0}^{D} w_i x_i$$

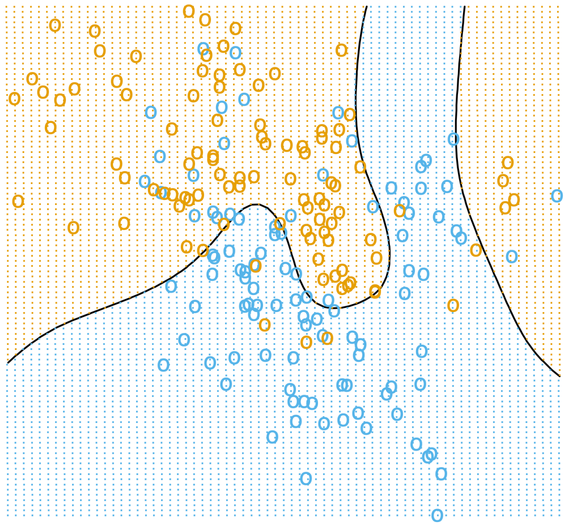
So, our decision boundary is linear!

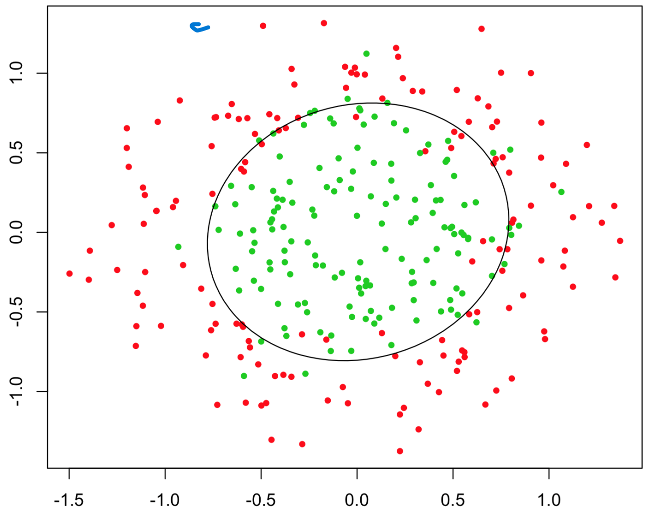


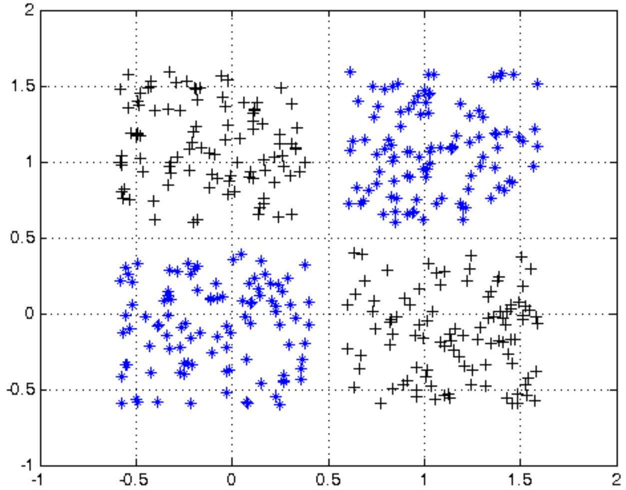
Complex Decision Boundaries?

What if we want to use a more complex decision boundary?

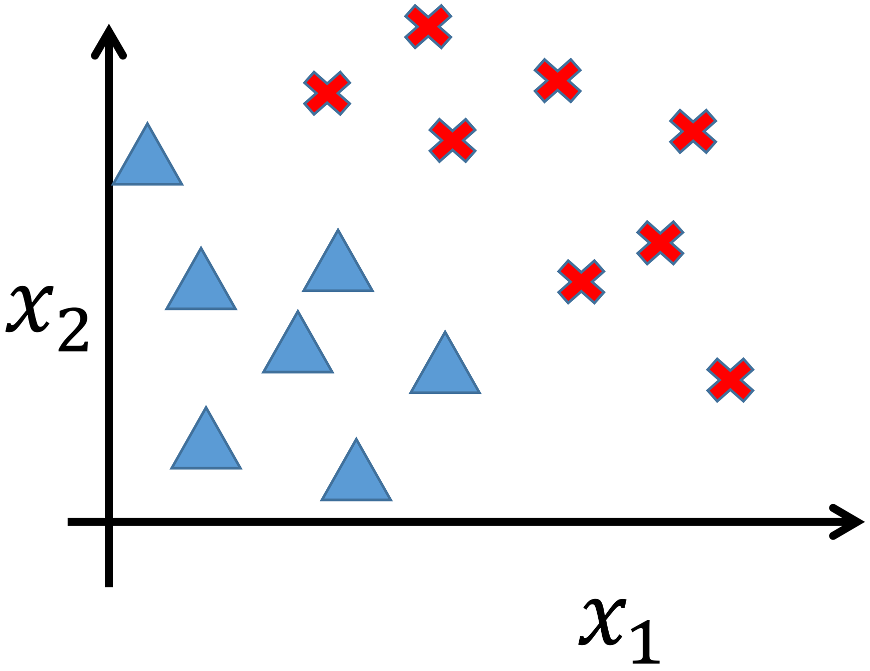
- Need more complex model/features!



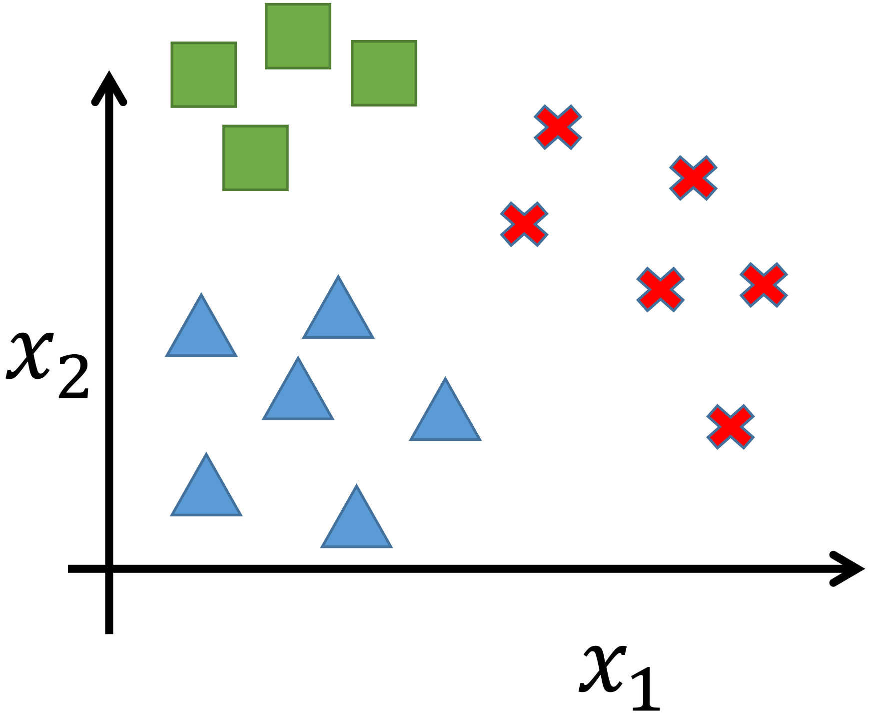




Binary classification

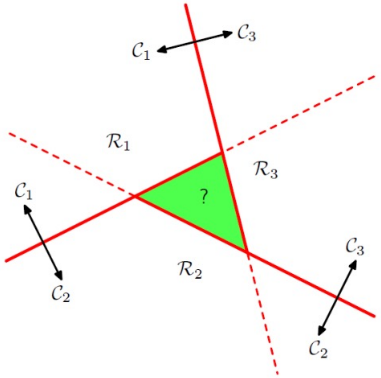


Multiclass classification



How do we extend Logistic Regression to Multiclass classification?

- Approach 1: one-versus-one
  - Computationally very expensive

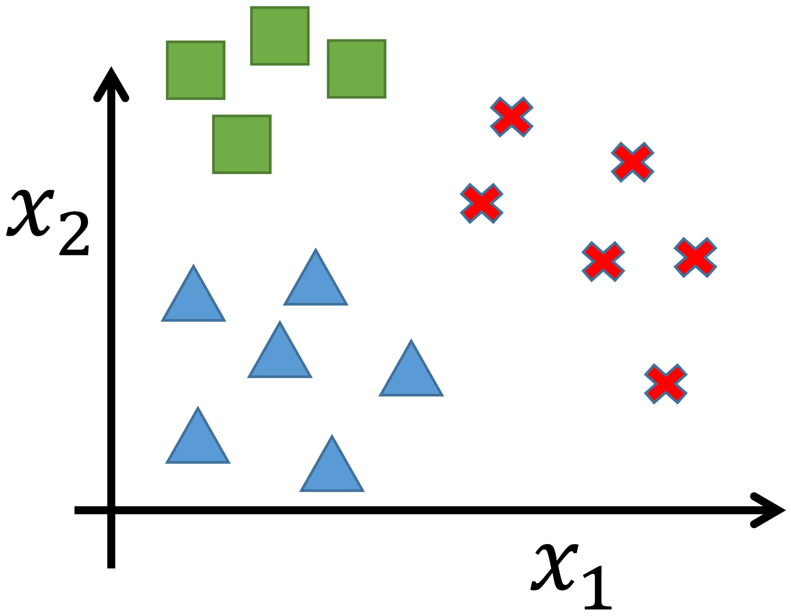


Find $(K-1)K/2$ classifiers $f_{(1,2)}, f_{(1,3)}, \ldots, f_{(K-1,K)}$

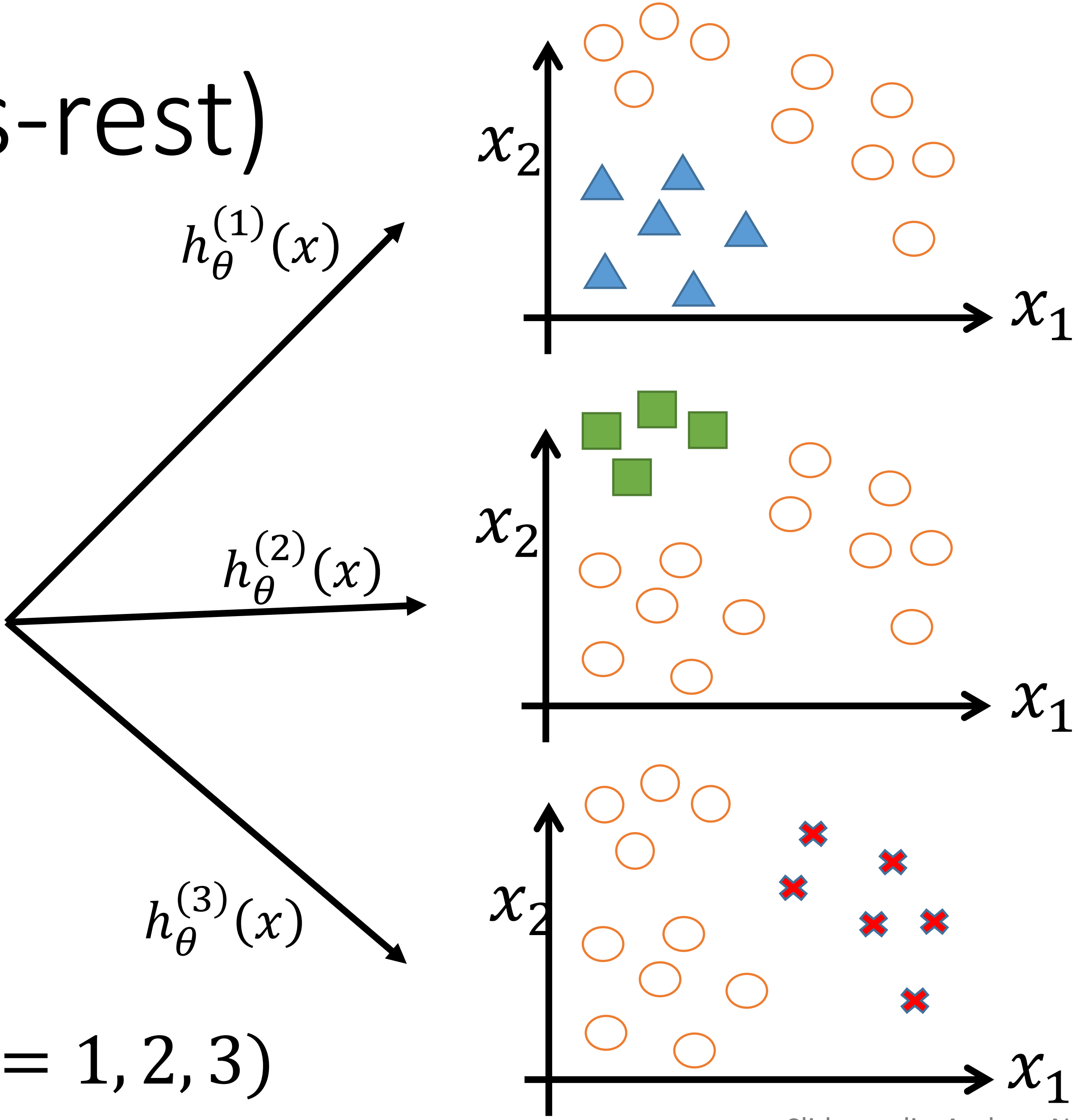
- $f_{(1,2)}$ classifies $1\ vs\ 2$
- $f_{(1,3)}$ classifies $1\ vs\ 3$
- …
- $f_{(K-1,K)}$ classifies $K-1\ vs\ K$

- Approach 2: one-versus-rest
- Approach 3: discriminant functions

One-vs-all (one-vs-rest)



Class 1: 
Class 2: 
Class 3: 



$$h_\theta^{(i)}(x) = P(y = i|x;\theta) \quad (i = 1,2,3)$$

Slide credit: Andrew Ng

One-vs-all

- Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$

- Given a new input x, pick the class i that maximizes

$$\max_{\mathrm{i}} h_\theta^{(i)}(x)$$

A way to squash $a = (a_1, a_2, \dots, a_i, \dots)$ into probability vector p

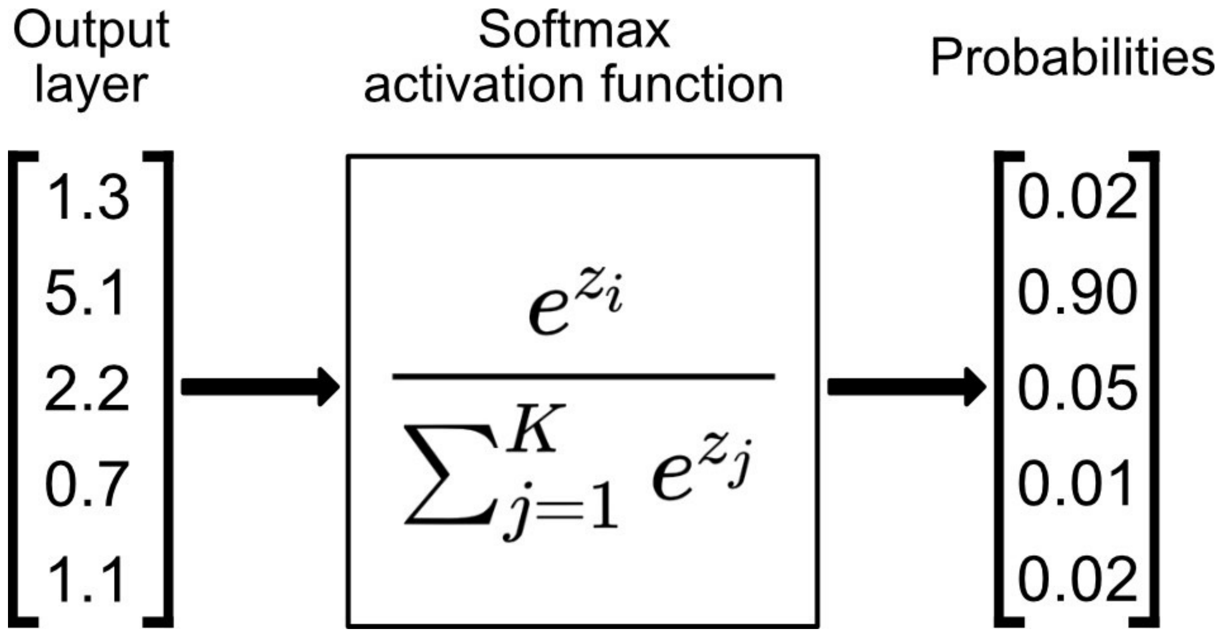
$$\mathrm{softmax}(a) = \left(\frac{\exp(a_1)}{\sum_j \exp(a_j)}, \frac{\exp(a_2)}{\sum_j \exp(a_j)}, \dots, \frac{\exp(a_i)}{\sum_j \exp(a_j)}, \dots \right)$$

SoftMax

A way to squash $a = (a_1, a_2, \dots, a_i, \dots)$ into probability vector p

$$\mathrm{softmax}(a) = \left(\frac{\exp(a_1)}{\sum_j \exp(a_j)}, \frac{\exp(a_2)}{\sum_j \exp(a_j)}, \dots, \frac{\exp(a_i)}{\sum_j \exp(a_j)}, \dots \right)$$

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$



SoftMax

Read more on the difference between Softmax and Sigmoid (6 min):

https://dataaspirant.com/difference-between-softmax-function-and-sigmoid-function/