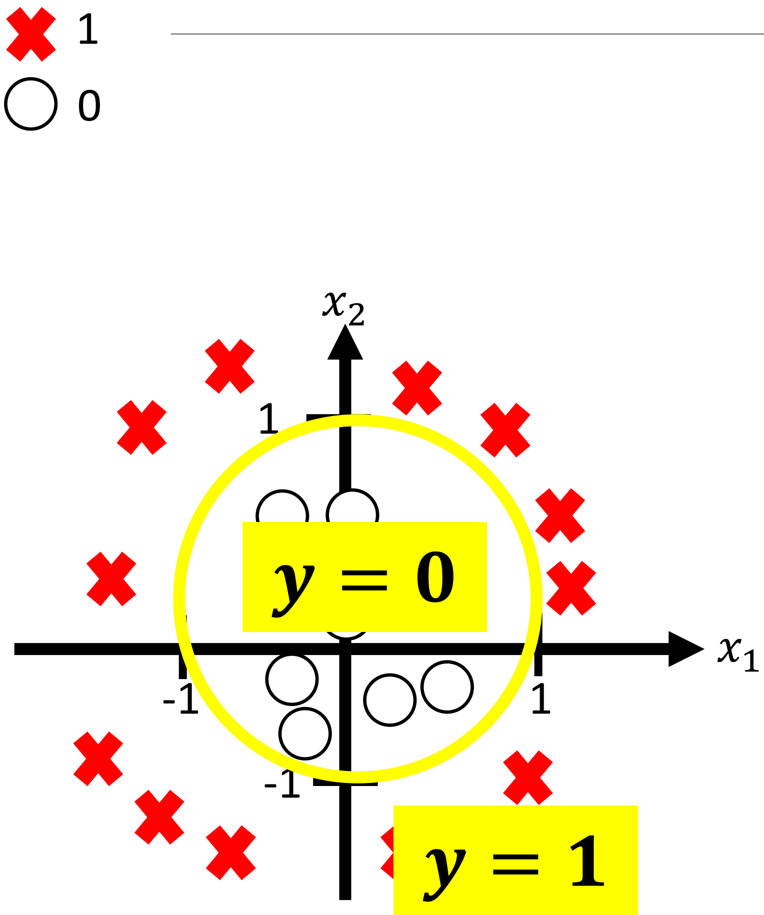
Review

DECISION BOUNDARIES

Non-linear Decision Boundary



Suppose we use polynomial features…

$$h(x) = (w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2)$$

Suppose we are able to train a model an get the following weights…

$$W = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Predict $y = 1$, if $-1 + x_1^2 + x_2^2 \geq 0$

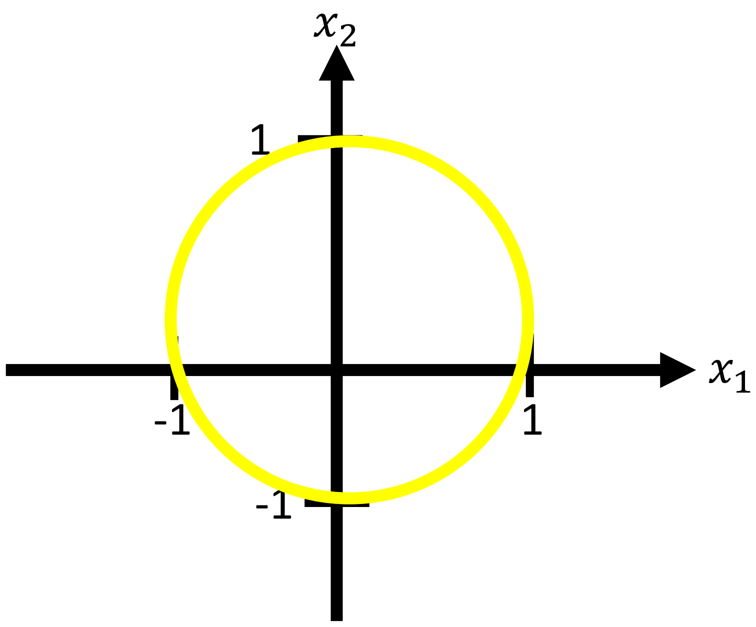
Predict $y = 1$, if $x_1^2 + x_2^2 \geq 1$

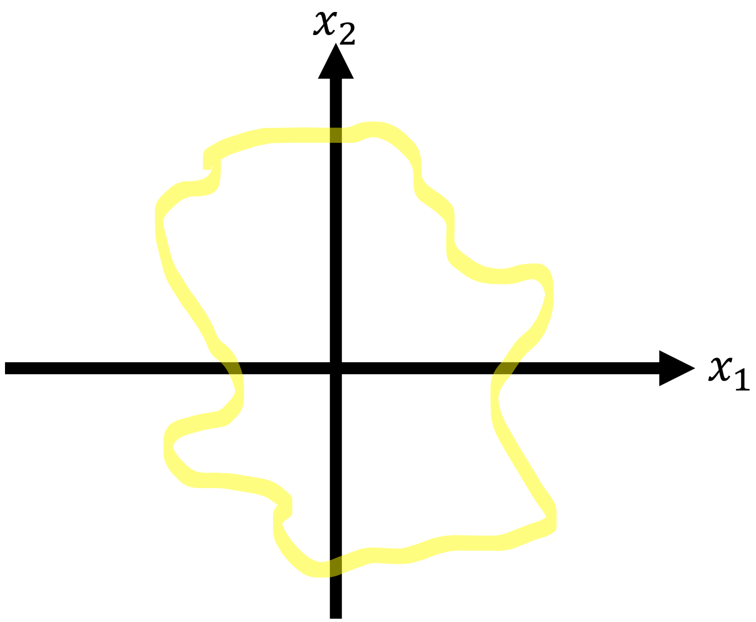
Equation of a circle

This means by controlling the weights, we can build complex decision boundaries!
Or simpler boundaries from complex features!

Non-linear Decision Boundary

$$h(x) = \left(w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_1^2x_2 + w_5x_2^2 + w_6x_2^2 + w_7x_1^3x_2 + \ \ldots\right)$$



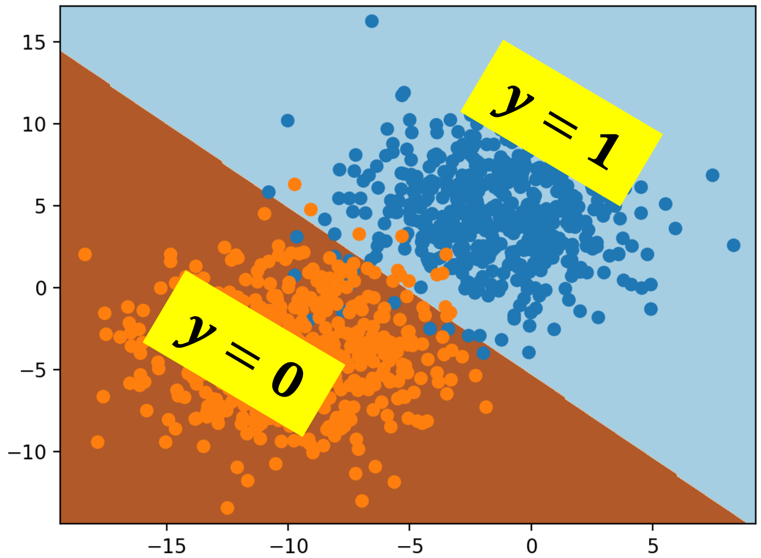


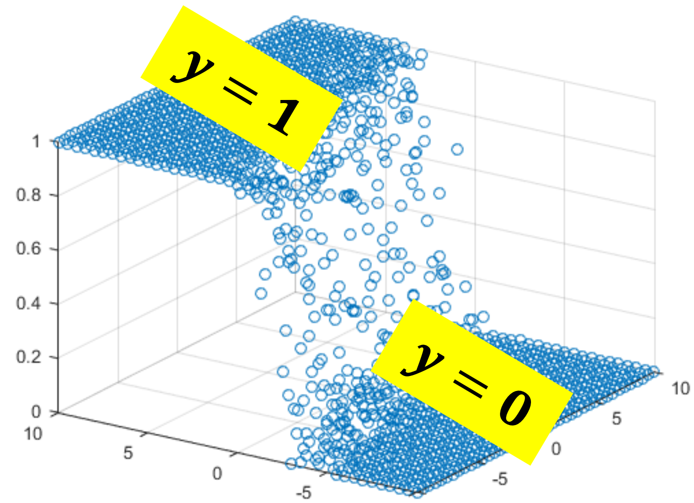
Verify on https://www.desmos.com/calculator

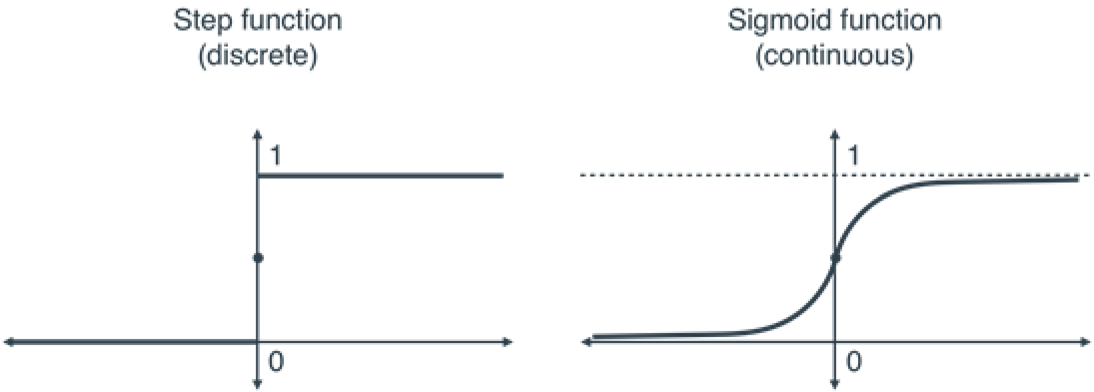
This means by controlling the weights, we can build complex decision boundaries!

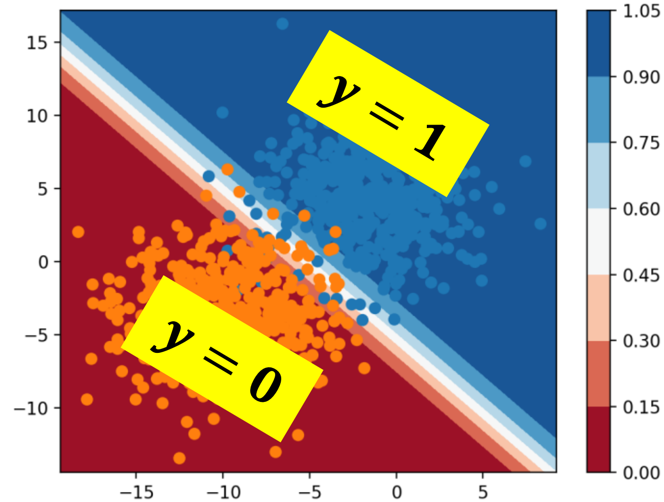
Recall that more complex boundaries can cause overfitting (i.e., high variance)

Hard VS Soft Boundaries Classifiers









https://livebook.manning.com/concept/machine-learning/sigmoid-function
https://machinelearningmastery.com/plot-a-decision-surface-for-machine-learning/
https://stackoverflow.com/questions/29360872/fitting-3d-sigmoid-to-data

Logistic Regression: Summary

Training Set: $\left\{\left(X^{(1)}, y^{(1)}\right), \left(X^{(2)}, y^{(2)}\right), \dots, \left(X^{(m)}, y^{(m)}\right)\right\}$

$\boldsymbol{m}$ **examples:** $\boldsymbol{x} \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix}$ $\boldsymbol{x_0 = \ 1, y \ \in \{0, 1\}}$

$$\boldsymbol{h(x) = \frac{1}{1 + e^{-W^T X}}}$$

What should be the cost function now?

Cost/Loss Function

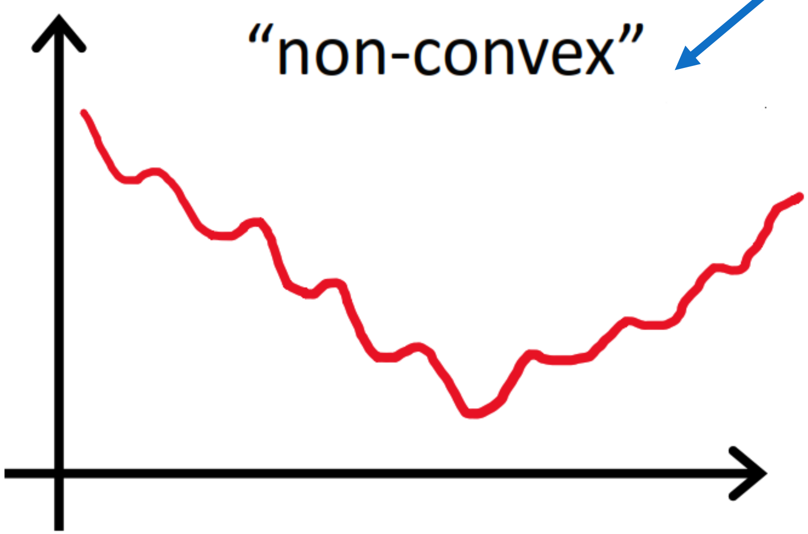
- Linear Regression:

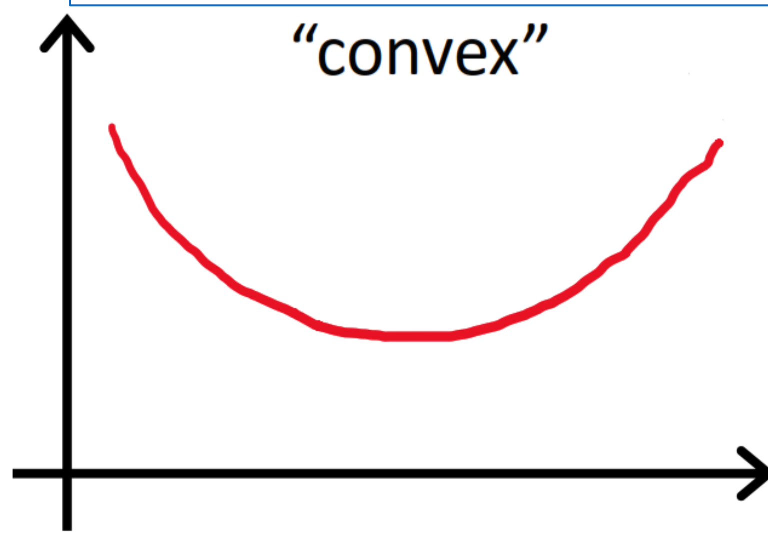
$$J(W) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h\left(x^{(i)}\right) - y^{(i)}\right)^2$$

- Logistic Regression

$$J(W) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(\frac{1}{1+e^{-w^T x^{(i)}}} - y^{(i)}\right)^2$$

It's plausible to have a cost function that is convex (to avoid many many local minimas)





Our Goal for the Cost Function

- Differentiable
- Convex

$$\boldsymbol{h(x)} = \frac{\mathbf{1}}{\mathbf{1}+\boldsymbol{e}^{-\boldsymbol{w}^T \boldsymbol{x}^{(i)}}}$$ **varies between 0 and 1**

- Punish **incorrect** answers with **high cost**
  - $y = 1, h(x) \rightarrow 0$
  - $y = 0, h(x) \rightarrow 1$

$\mathbf{log_{10}(0)} = ?$
$\mathbf{log_{10}(1)} = ?$

$\mathbf{log_{10}(0)} = -\infty$
$\mathbf{log_{10}(1)} = \mathbf{0}$

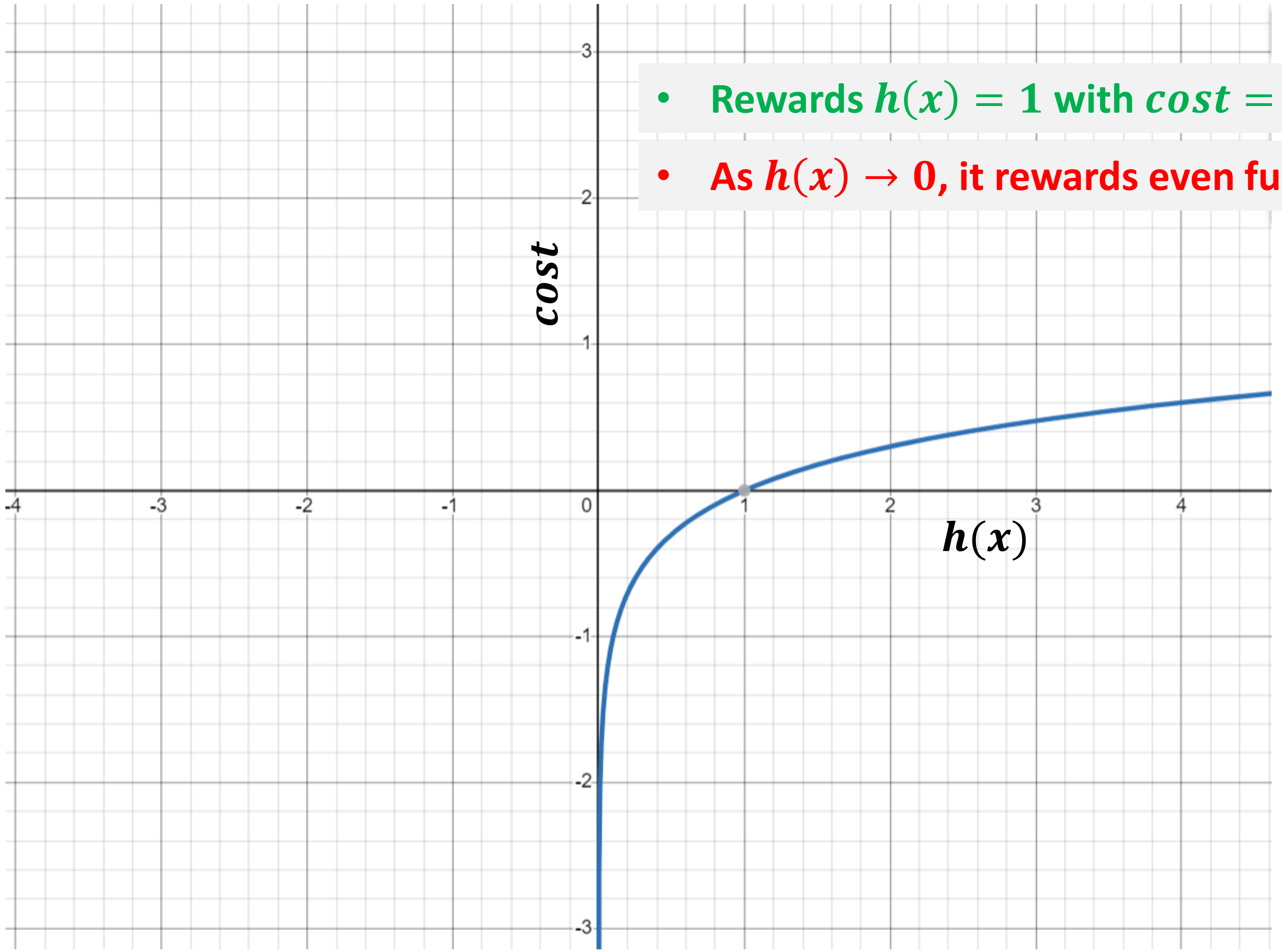
How can we use these two relations in designing our cost function?

- Reward **correct** answers with a **low cost**
  - $y = 1, h(x) \rightarrow 1$
  - $y = 0, h(x) \rightarrow 0$

Remember, now y is either 0 or 1, but prediction would be a real number between 0 and 1

Verify on https://www.desmos.com/calculator

$\boldsymbol{y = 1}$ $\boxed{\boldsymbol{cost = log(h(x))}}$

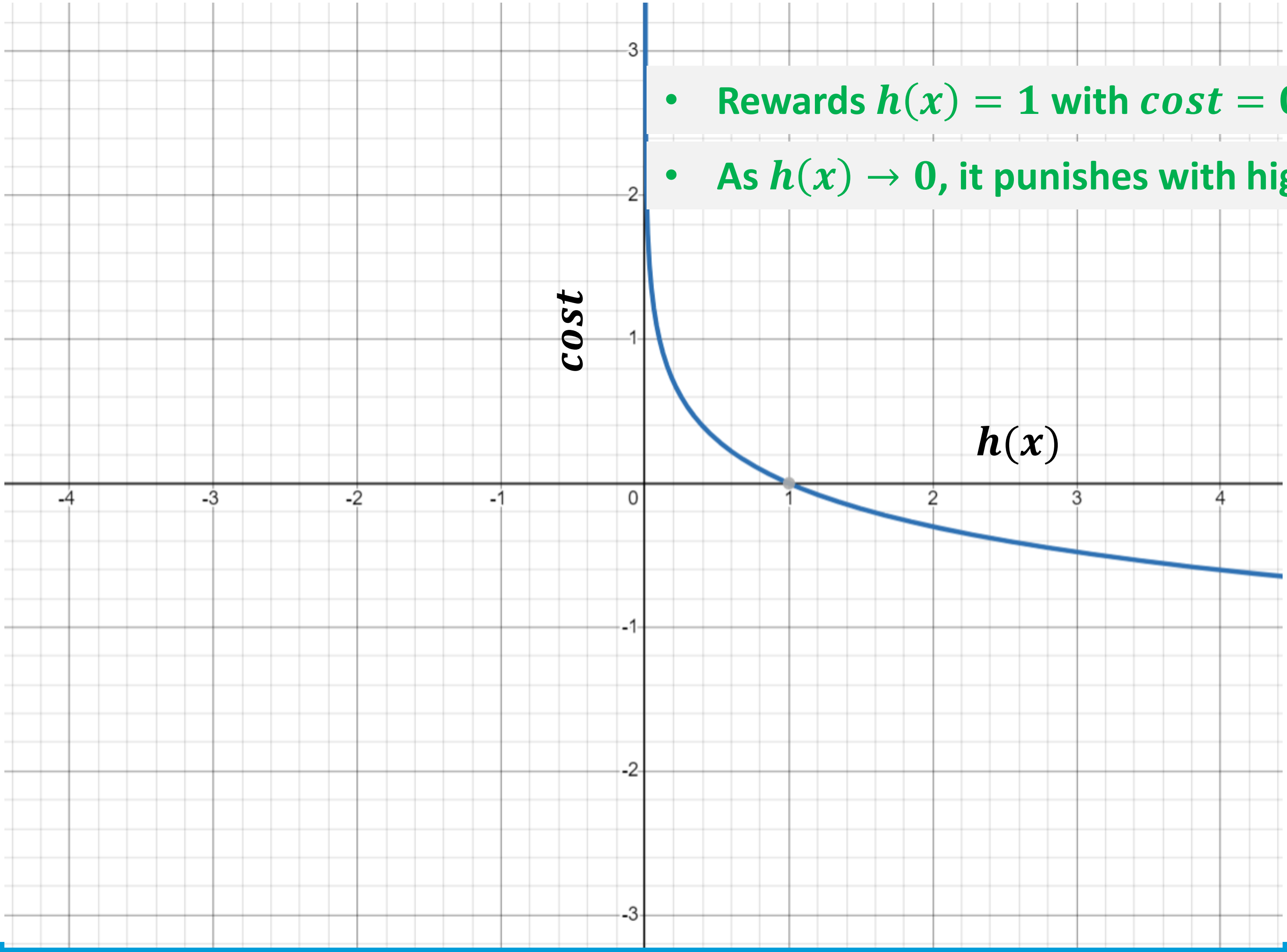


- Rewards $h(x) = 1$ with $cost = 0$
- As $h(x) \rightarrow 0$, it rewards even further!

Verify on https://www.desmos.com/calculator

$y = 1$

$$\boxed{cost = -log(h(x))}$$



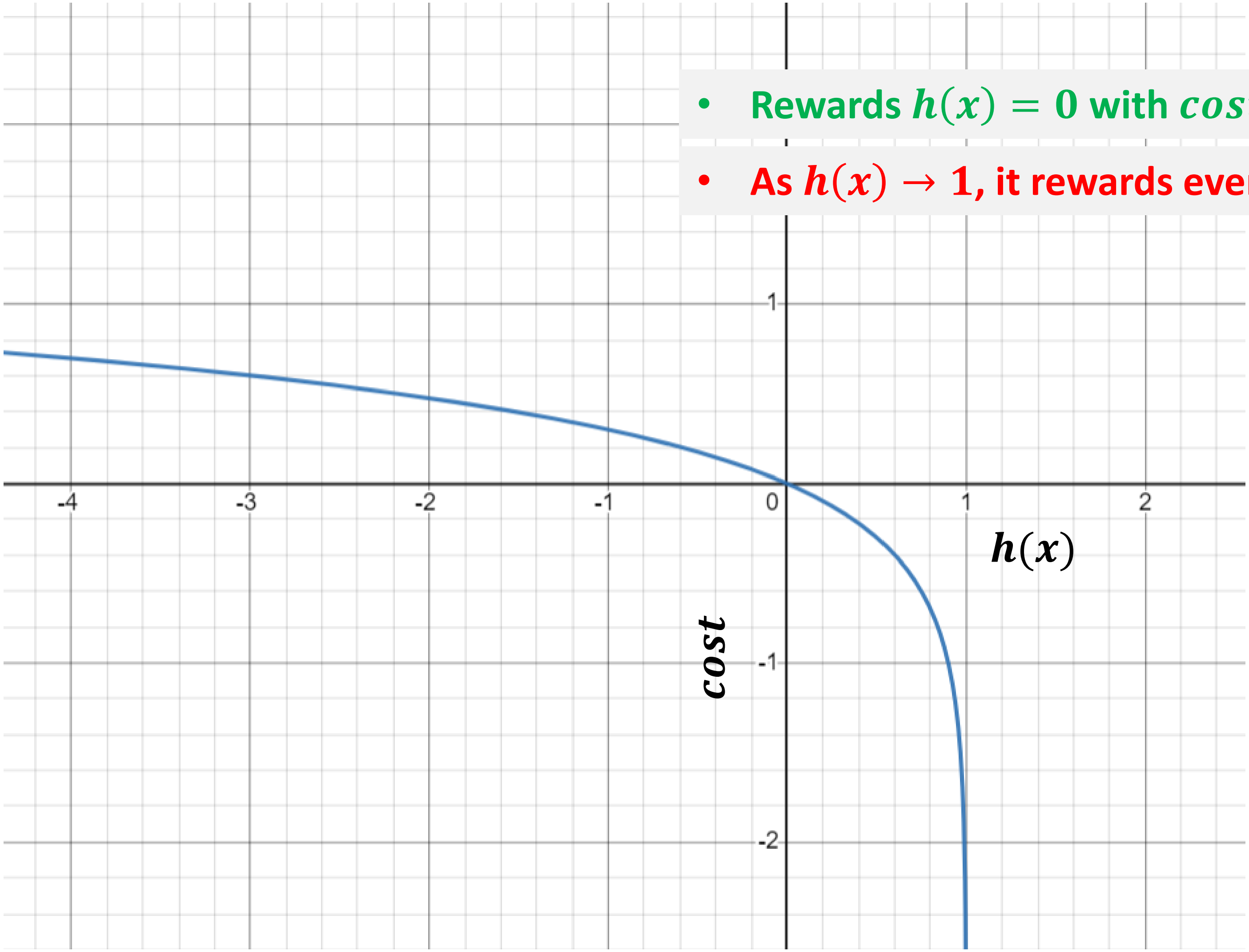
- Rewards $h(x) = 1$ with $cost = 0$
- As $h(x) \to 0$, it punishes with higher cost

Verify on https://www.desmos.com/calculator

$y = 0$

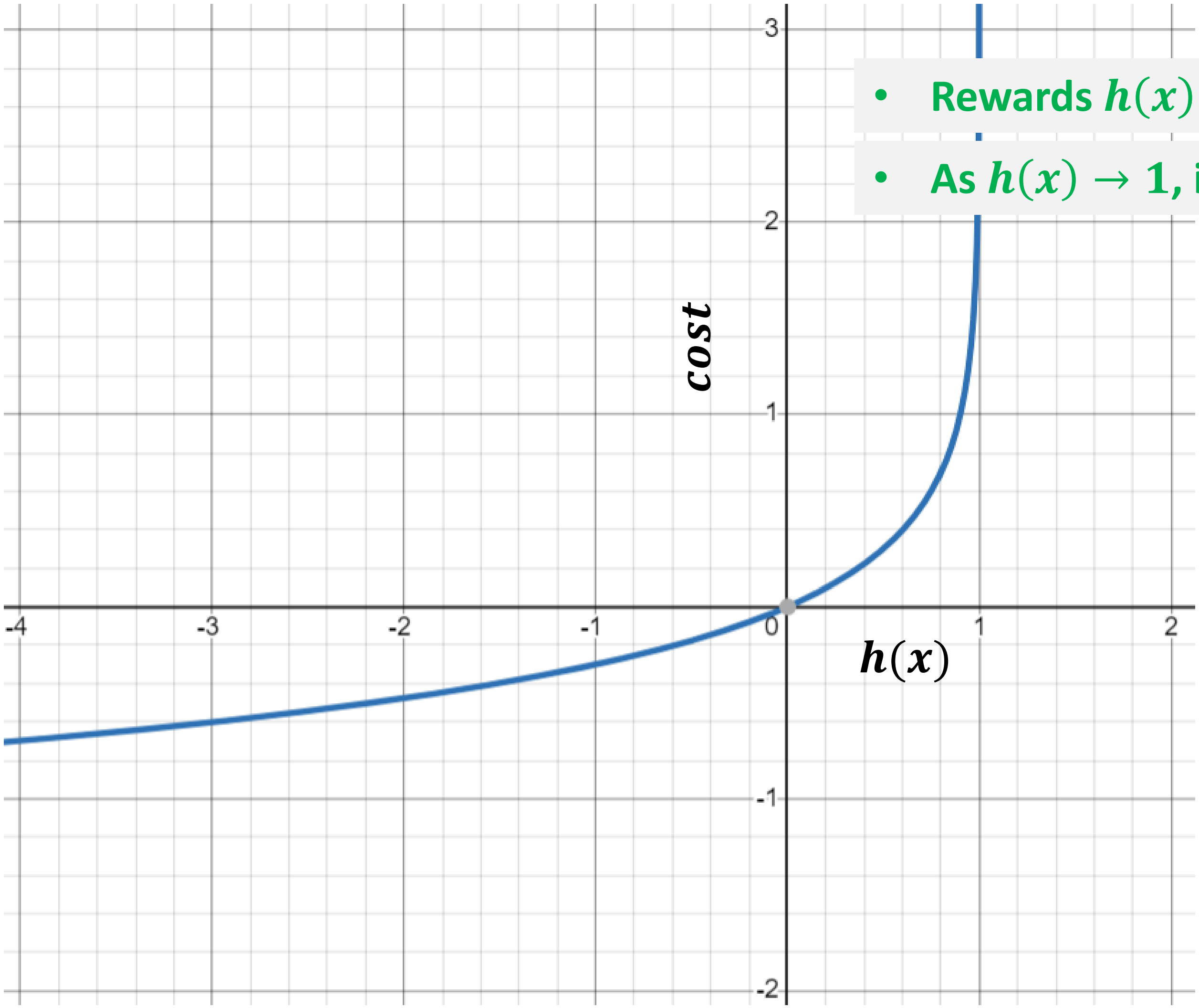
$$\boxed{cost = log(1 - h(x))}$$

- Rewards $h(x) = 0$ with $cost = 0$
- As $h(x) \rightarrow 1$, it rewards even further!



Verify on https://www.desmos.com/calculator

$y = 0$ $\boxed{cost = -log(1 - h(x))}$



- Rewards $h(x) = 0$ with $cost = 0$
- As $h(x) \rightarrow 1$, it punishes with higher cost

Logistic Regression Cost Function

$$\boxed{J(W) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h\left(x^{(i)}\right), y^{(i)}\right)}$$

$$\boxed{Cost(h(x), y) = \begin{cases} -log\left(h(x)\right) & \text{if } y = 1 \\ -log\left(1 - h(x)\right) & \text{if } y = 0 \end{cases}}$$

Can we combine these and create a switch that turns on the correct cost values for the values of y?

Note: $y = 0$ or $y = 1$ always! (Recall Bernoulli Distribution)

We can combine summations and products using interpolation:

If $y \in \{0, 1\}$: $\boxed{Cost = a^{y} b^{(1-y)}}$ **Chooses a if $y = 1$ and b if $y = 0$**

$\boxed{Cost = y \times a + (1 - y) \times b}$ **Chooses a if $y = 1$ and b if $y = 0$**

Logistic Regression Cost Function

$$\boldsymbol{J(W) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h\left(x^{(i)}\right), y^{(i)}\right)}$$

$$\boldsymbol{Cost(h(x), y) = \begin{cases} -log\left(h(x)\right) & \textbf{if } y = 1 \\ -log\left(1 - h(x)\right) & \textbf{if } y = 0 \end{cases}}$$

Can we combine these and create a switch that turns on the correct cost values for the values of y?

$$\boldsymbol{Cost(h(x), y) = -y\, log\left(h(x)\right) - (1 - y)\, log\left(1 - h(x)\right)}$$

$$\textbf{if } \boldsymbol{y = 1: cost(h(x), y) = -log(h(x)}$$

$$\textbf{if } \boldsymbol{y = 0: cost(h(x), y) = -log(1 - h(x)}$$

Logistic Regression Cost Function

$$\boldsymbol{Cost(h(x), y) = -y\, log\big(h(x)\big) - (1 - y)\, log\big(1 - h(x)\big)}$$

- We have created a loss function that prefers the correct class labels of the **training examples** to be more likely.
  - It **estimates** the model parameters to **maximize the likelihood** of the training set
  - This is called **maximum likelihood estimation (MLE)**
- We choose the parameters w_j that maximize the log probability of the true y labels in the training data, given the observations X.
- The resulting loss function is the negative log likelihood loss **(log loss)**, generally called **cross-entropy loss.**

LR Cost Function: Summary

$$\boxed{J(W) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h\left(x^{(i)}\right), y^{(i)}\right)}$$

$$\boxed{J(W) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\ log\left(h\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - h\left(x^{(i)}\right)\right)\right]}$$

To fit parameters W:

$$\boxed{\min_{W} J(W)}$$

To make a prediction for a new x:

Output: $h(x) = \frac{1}{1+e^{-W^T x}}$ **Gives** $P(y = 1|x; W)$

Note: This gives you probability of $y = 1$

How would gradient descent work now?

Gradient Descent

$$J(W) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\ log\left(h\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - h\left(x^{(i)}\right)\right)\right]$$

Repeat {

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(W)$$

} **(simultaneously update all w_j)**

After a very long math, you will end up having the following derivative!

$$\frac{\partial}{\partial w_j}\left(J(W)\right) = \frac{1}{m}\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

(see Jurafksy SLP3 for derivation)

Same as MSE from Linear Regression!

Batch Gradient Descent

$$J(W) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\ log\left(h\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - h\left(x^{(i)}\right)\right)\right]$$

Epsilon is the weights change threshold…

Repeat until $W_{t+1} - W_t \geq \epsilon$ **{**

$$w_j := w_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right) . x_j^{(i)}$$

}

(Simultaneously update all w_j)

Stochastic Gradient Descent

$$J(W) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\ log\left(h\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - h\left(x^{(i)}\right)\right)\right]$$

Repeat until $W_{t+1} - W_t \geq \epsilon$ **{**

Repeat for each training instance $\left(x^{(i)}, y^{(i)}\right)$**, in random order {**

$$w_j := w_j - \alpha\left(h\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

}

}

(Simultaneously update all w_j)

Mini Batch Gradient Descent

$$J(W) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\ log\left(h\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - h\left(x^{(i)}\right)\right)\right]$$

Repeat until $W_{t+1} - W_t \geq \epsilon$ **{**

Repeat for each randomly selected mini batch of size k **{**

$$w_j := w_j - \alpha \frac{1}{k} \sum_{i=1}^{k} \left(h\left(x^{(i)}\right) - y^{(i)}\right) . x_j^{(i)}$$

}

}

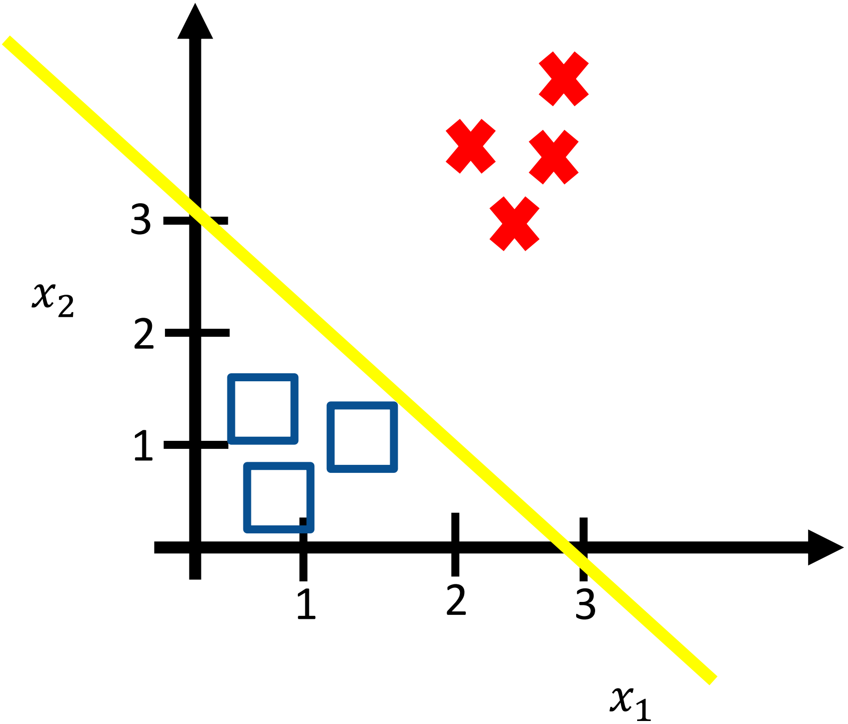
(Simultaneously update all w_j)

Multiclass Classification

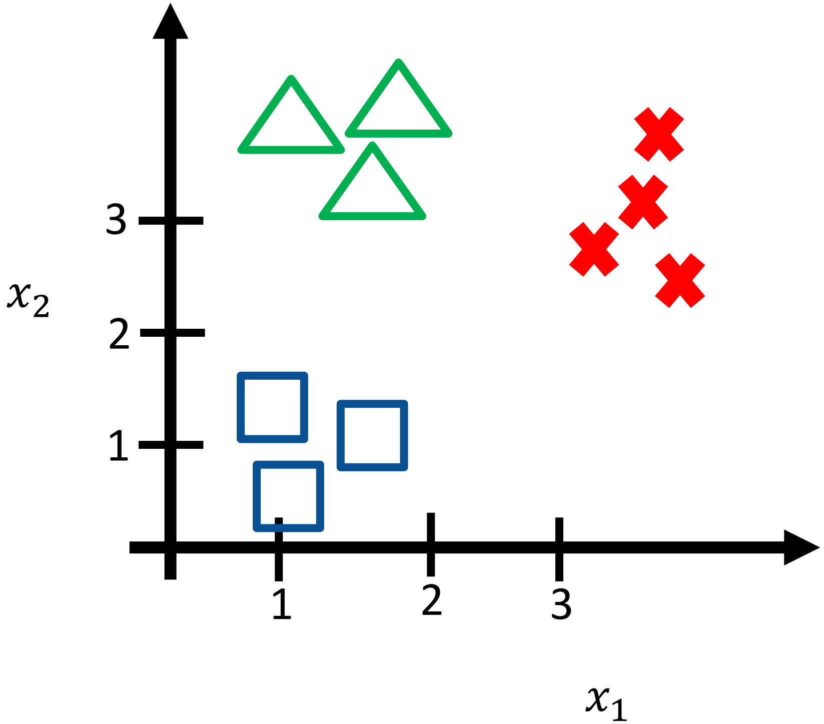
- **What if we have more than two classes?**

Multiclass (Multinomial) Classification

Binary Classification
(Instance belongs to one of two classes)



Multiclass Classification
(Instance belongs to one of many classes)



Note: Multiclass classification is different than multilabel classification

Multiclass Classification

- Logistic Regression inherently models binary classification and needs meta-strategies for multiclass classification

- **Multiclass classification**
  - Multiple class labels are present in the dataset

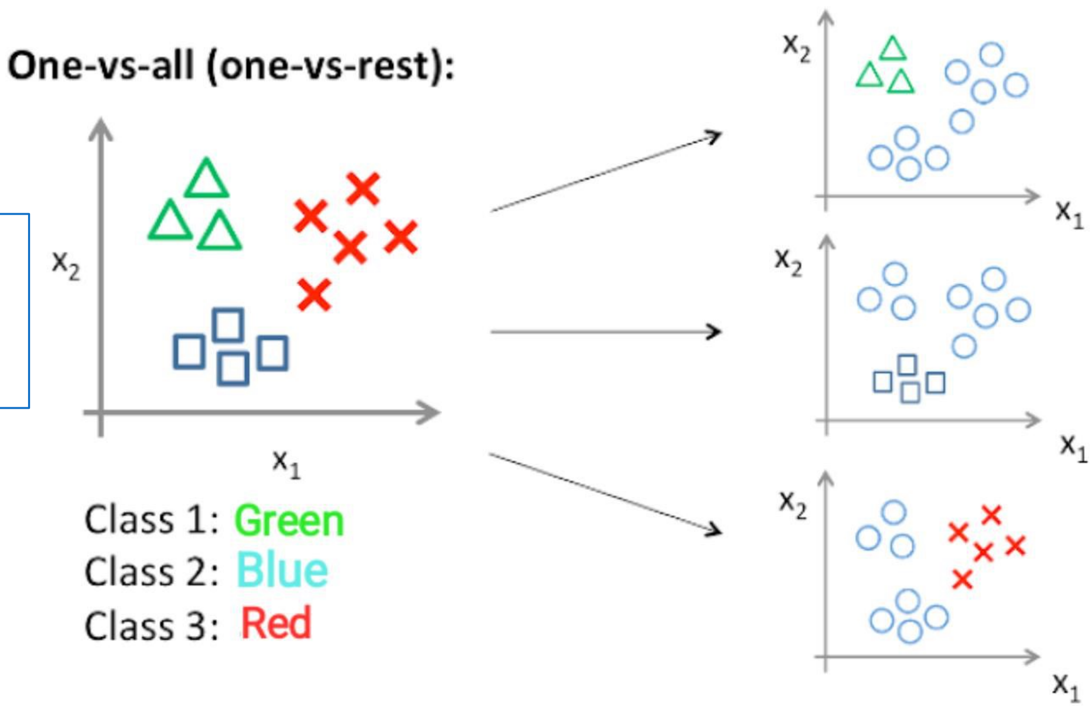
- **Binary classifiers for multiclass classification**
  - **One VS Rest (aka One Vs All)**
    - Splits a multiclass classification into one binary classification problem per class
    - n −class instances then create n binary classifiers
  - **One VS One**
    - Splits a multiclass classification into one binary classification problem per each pair of classes
    - n −class instances then have $\binom{n}{2} = \frac{n \times (n-1)}{2}$ binary classifier models

Credit: https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94daed32a87b

One VS All (One VS Rest)

- Train one classifier per class ($x \in class_i \; vs \; x \notin class_i$) for all n classes
- In the scoring phase, all the n classifiers predict the probability of each class and the class with highest probability is selected.

- Classifier 1:- [Green] VS [Red, Blue]
- Classifier 2:- [Blue] VS [Green, Red]
- Classifier 3: [Red] VS [Blue, Green]



Create Separate Datasets for Training

Main Dataset

Features			Classes
x1	x2	x3	G
x4	x5	x6	B
x7	x8	x9	R
x10	x11	x12	G
x13	x14	x15	B
x16	x17	x18	R

Class 1 :- Green Class 2 :- Blue Class 3 :- Red

Training Dataset 1
Class :- Green

Features			Green
x1	x2	x3	**+1**
x4	x5	x6	-1
x7	x8	x9	-1
x10	x11	x12	**+1**
x13	x14	x15	-1
x16	x17	x18	-1

Training Dataset 2
Class :- Blue

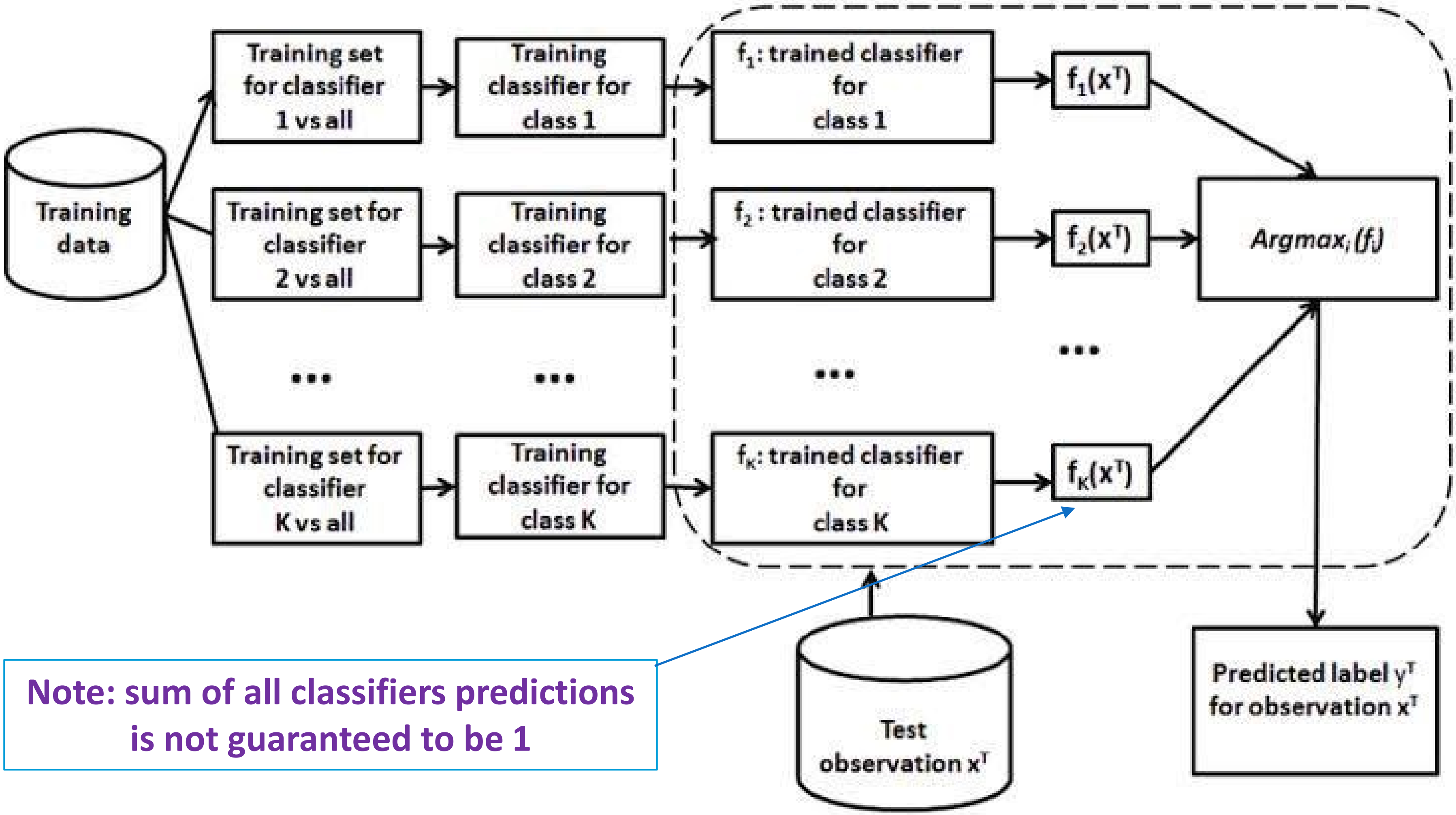
Features			Blue
x1	x2	x3	-1
x4	x5	x6	**+1**
x7	x8	x9	-1
x10	x11	x12	-1
x13	x14	x15	**+1**
x16	x17	x18	-1

Training Dataset 3
Class :- Red

Features			Red
x1	x2	x3	-1
x4	x5	x6	-1
x7	x8	x9	**+1**
x10	x11	x12	-1
x13	x14	x15	-1
x16	x17	x18	**+1**

Credit: https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94daed32a87b

One VS All Pipeline

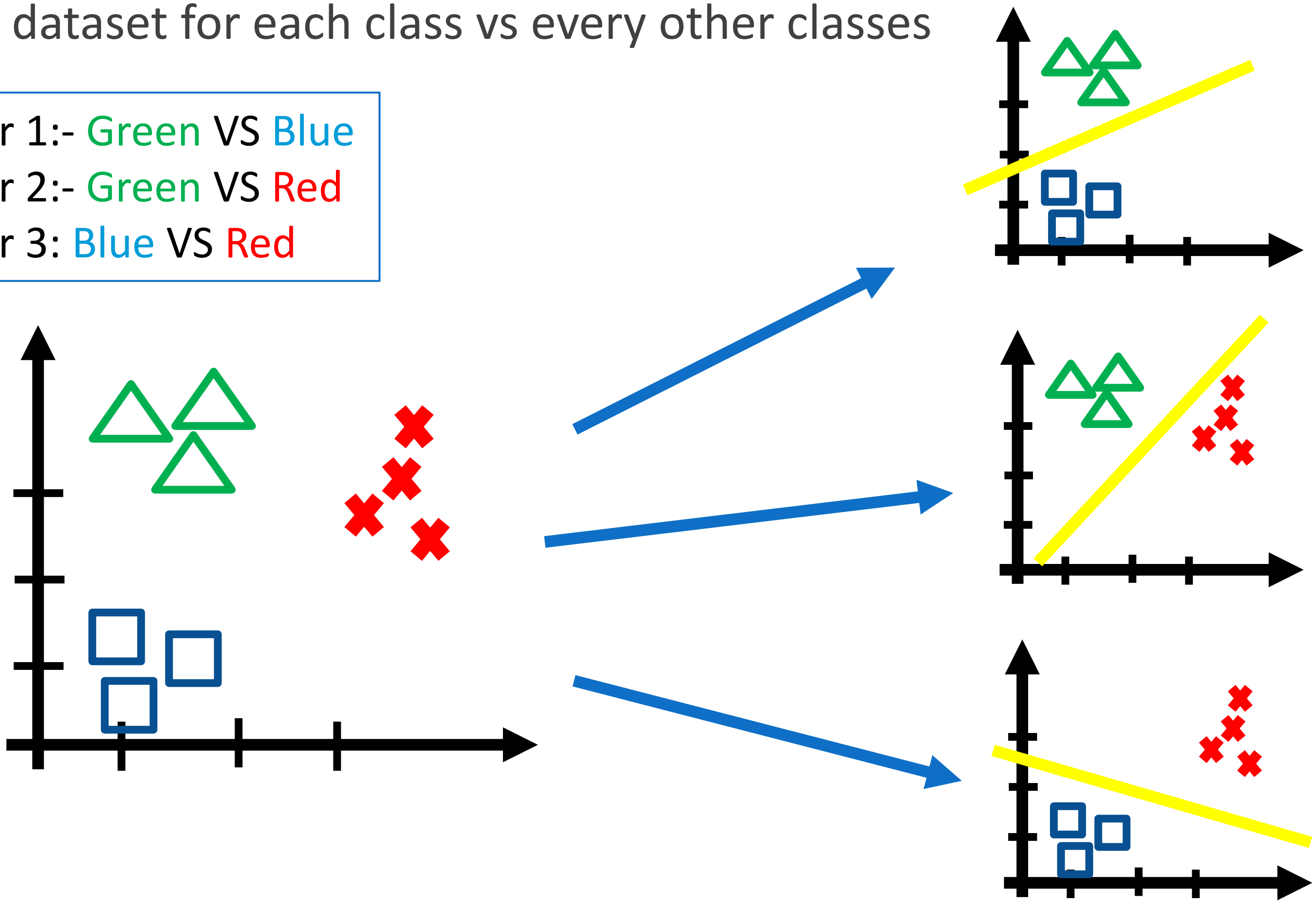


Note: sum of all classifiers predictions is not guaranteed to be 1

One VS One

- For n classes, we must generate $\frac{n\times(n-1)}{2}$ binary classifier models
- Using this classification approach, we split the primary dataset into one dataset for each class vs every other classes

- Classifier 1:- Green VS Blue
- Classifier 2:- Green VS Red
- Classifier 3: Blue VS Red



One VS One

- Each binary classifier predicts one class label and the model with the most predictions or votes is predicted by the one-vs-one strategy

- If the binary classification models predict a numerical class membership, such as probability
  - then the $argmax$ of the sum of the scores (class with the largest sum score) is predicted as the class label.

- Classifier 1:- Green VS Blue
- Classifier 2:- Green VS Red
- Classifier 3: Blue VS Red

- $Score(r) = P_2(r) + P_3(r)$
- $Score(b) = P_1(b) + P_3(b)$
- $Score(g) = P_1(g) + P_2(g)$

Take $argmax$ of scores for final prediction.

oVr vs oVo

- oVr trains fewer classifiers and is faster overall

- oVo is less prone to imbalance in datasets

$C1$	$C2$	$C3$
10	10	10

oVo

$C1$	$C2$
10	10

$C2$	$C3$
10	10

$C1$	$C3$
10	10

oVr

$C1$	$rest$
10	20

$C2$	$rest$
10	20

$C3$	$rest$
10	20

Evaluating Performance of Multiclass Classification

- **How to evaluate the performance of classifiers in case of multiclass classification?**

Recall: Binary Classification Evaluation

Actual Labels/Ground Truth

Predicted Labels/Predictions

	Spam (1)	Not Spam (0)
Spam (1)	**TP = 8**	**FP = 30**
Not Spam (0)	**FN = 2**	**TN = 960**

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$F1 = \frac{2 \times (Precision \times Recall)}{(Precision + Recall)}$$

Confusion Matrix for 3-class Classification

		gold labels urgent	normal	spam	
system output	urgent	8	10	1	**precision**u= $\frac{8}{8+10+1}$
	normal	5	60	50	**precision**n= $\frac{60}{5+60+50}$
	spam	3	30	200	**precision**s= $\frac{200}{3+30+200}$
		recallu = $\frac{8}{8+5+3}$	**recall**n = $\frac{60}{10+60+30}$	**recall**s = $\frac{200}{1+50+200}$	

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$F1 = \frac{2 \times (Precision \times Recall)}{(Precision + Recall)}$$

What's the Accuracy?

Now we have three values of Precision and Recall!! How to get single value?

Combining P/R from 3 Classes to Get One Metric

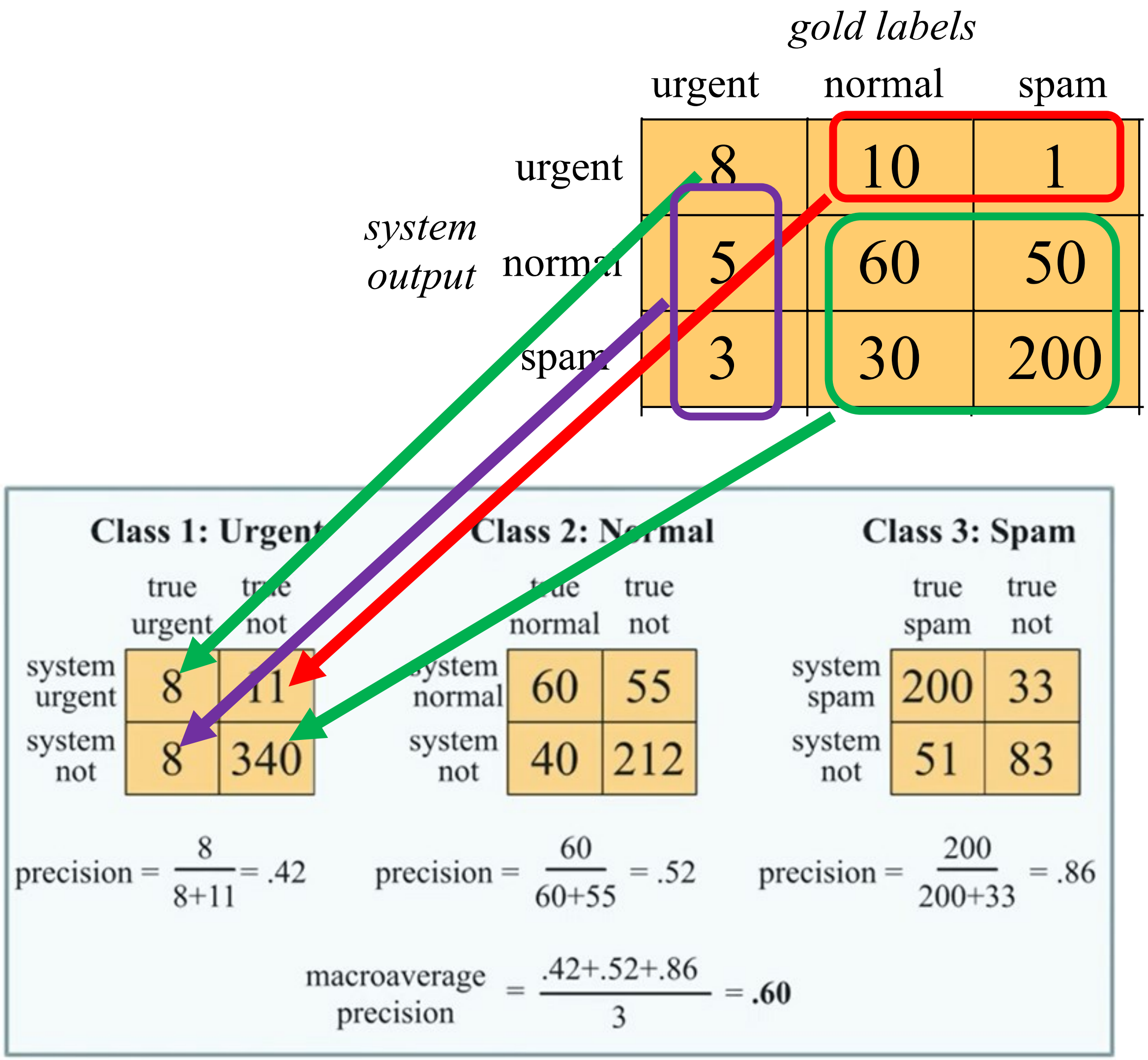
Macro-averaging

- Computer the performance (P/R) for each class, and then take average

Micro-averaging

- Collect decisions (TP, TN, FP, FN) for all classes into one confusion matrix
- Compute Precision and Recall from this new matrix

Macro-Averaging



Micro-Averaging

		gold labels urgent	normal	spam
system output	urgent	8	10	1
	normal	5	60	50
	spam	3	30	200

Similarly, you can calculate Macro and Micro Recall and then F1.

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

Evaluation

- A micro-average is dominated by the more frequent class (in this case, spam)
  - As the counts are pooled
- The macro-average better reflects the statistics of the smaller classes
  - Is more appropriate when performance on all the classes is equally important

Assignment 3: Task 2

- Train logistic regression classifier for “me” and “not me” labels in your images dataset
  - Compare the performance with decision tree in terms of F1-score, on the test split
  - Make sure you are using the same train and test splits that you used in training the decision tree
- Train multinomial logistic regression classifier for your “expressions” labels
  - Computer the macro averaging precision, recall, and f1-score
- In both models, make sure you are using “log loss” and appropriate “penalty” (i.e., regularization).
- Now, use the trained logistic regression model for your webcam script and also show the probability along with the class label on webcam stream.
- Build a GUI using “Gradio” that takes an image as input, and predicts the class and displays the probability.

Quiz 3

- Thursday 27th April
- Lectures 15, 16

Book Reading

- Murphy – Chapter 8
- Jurafsky – Chapter 5, Chapter 4