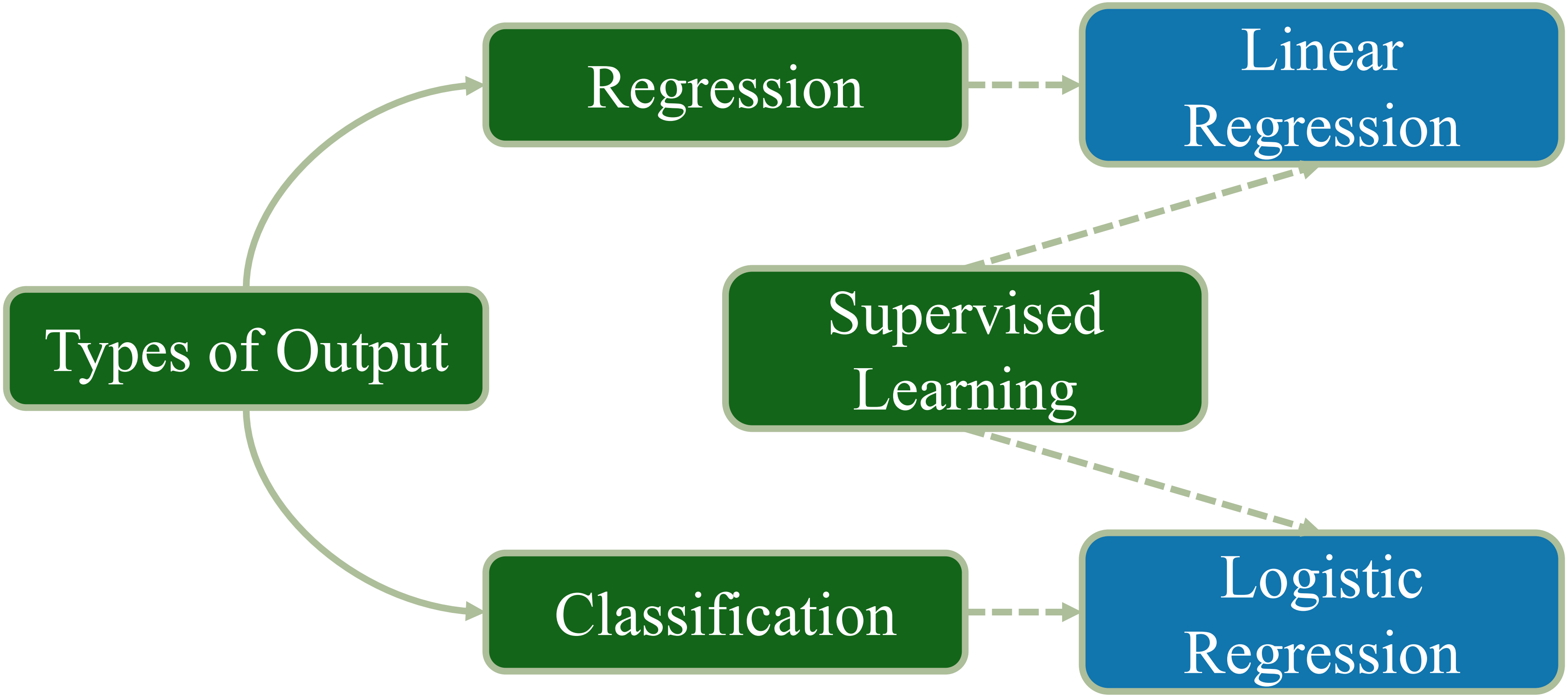
ITCS 6156/8156 Fall 2023
Machine Learning

Logistic Regression

Instructor: Hongfei Xue
Email: hongfei.xue@charlotte.edu
Class Meeting: Mon & Wed, 4:00 PM – 5:15 PM, CHHS 376

Some content in the slides is based on Dr. Razvan's and Dr. Andrew's lectures

Logistic Regression

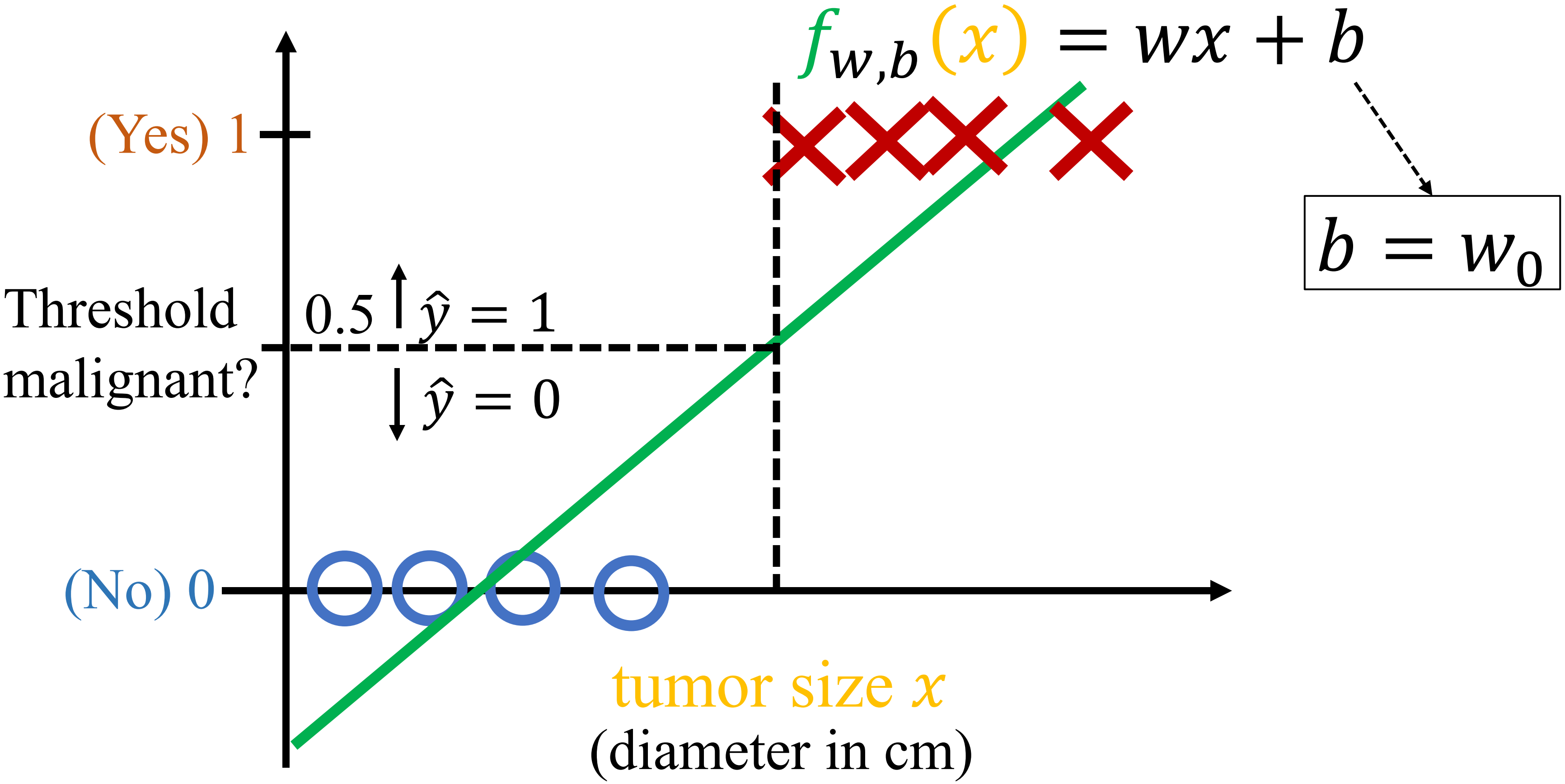


Classification

Question	Answer "y"
Is this email spam?	no yes
Is the transaction fraudulent?	no yes
Is the tumor malignant?	no yes

- **binary classification:**
  - "y" can only be one of two values:
    - false: 0: "negative class" = "absence"
    - true: 1: "positive class" = "presence"

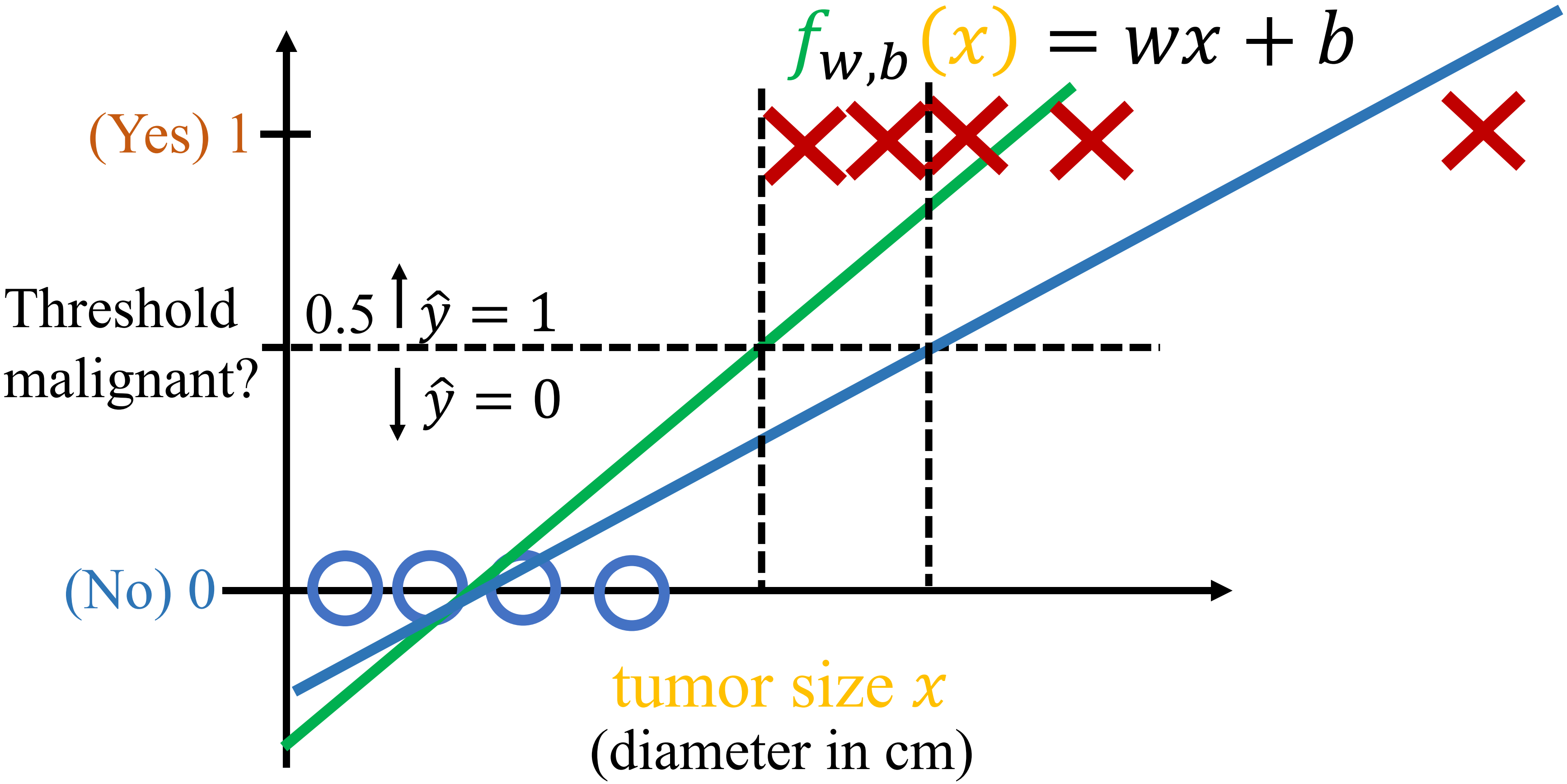
Linear Regression Approach



if $f_{w,b}(x) < 0.5 \rightarrow \hat{y} = 0$

if $f_{w,b}(x) \geq 0.5 \rightarrow \hat{y} = 1$

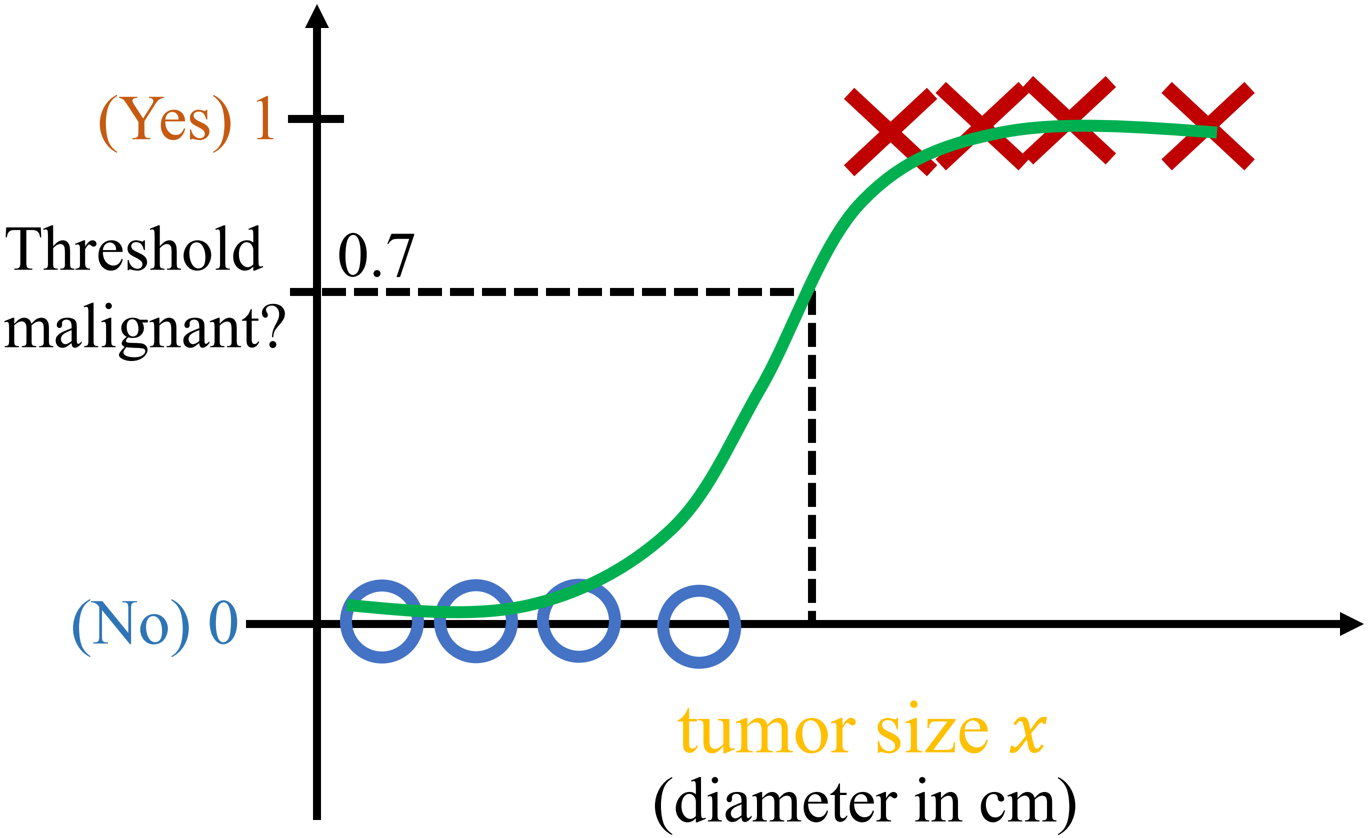
Linear Regression Approach



if $f_{w,b}(x) < 0.5 \rightarrow \hat{y} = 0$

if $f_{w,b}(x) \geq 0.5 \rightarrow \hat{y} = 1$

Logistic Function

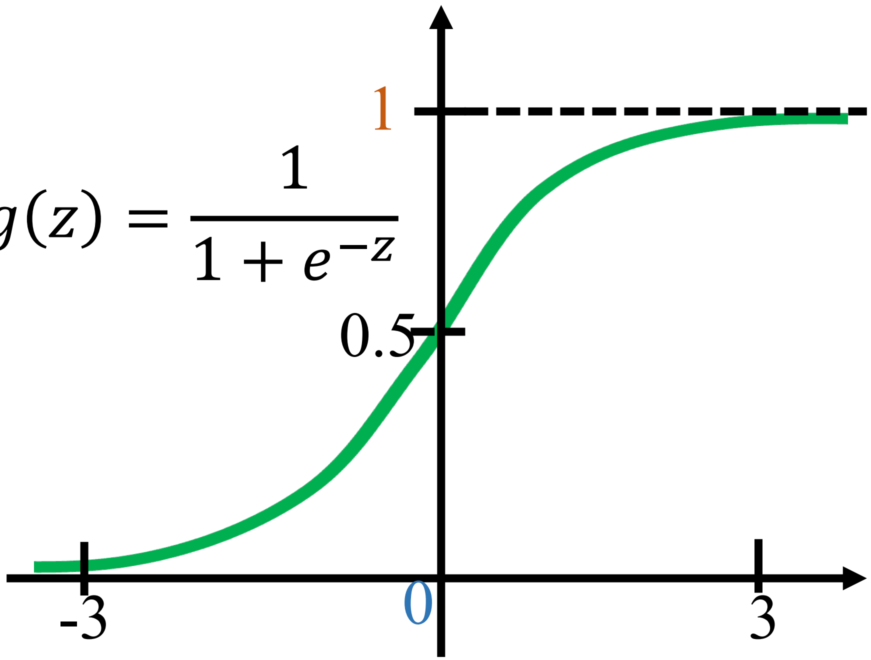


Probabilistic Discriminative Models: directly model the posterior class probabilities $p(C|\mathbf{x}; \boldsymbol{w}, b)$

Logistic Function

- Want outputs between 0 and 1

$$g(z) = \frac{1}{1+e^{-z}}$$



$$w \cdot x + b$$

$$\downarrow$$

$$z$$

$$\downarrow$$

$$g(z) = \frac{1}{1+e^{-z}}$$

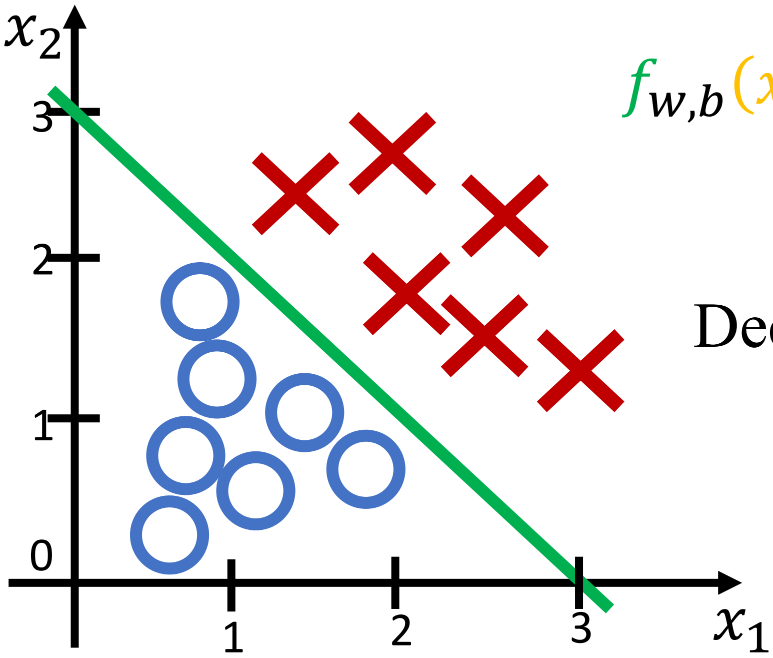
$$f_{w,b}(x) \;= g(w \cdot x + b\,)$$

$$= \frac{1}{1+e^{-(w \cdot x + b\,)}}$$

logistic regression

- sigmoid function
- logistic function
- outputs between 0 and 1 $g(z) = \frac{1}{1+e^{-z}}$, $0 < g(z) < 1$

Decision Boundary



$$f_{w,b}(x) = g(z) = g(w_1x_1 + w_2x_2 + b)$$

Decision Boundary: $z = w \cdot x + b = 0$

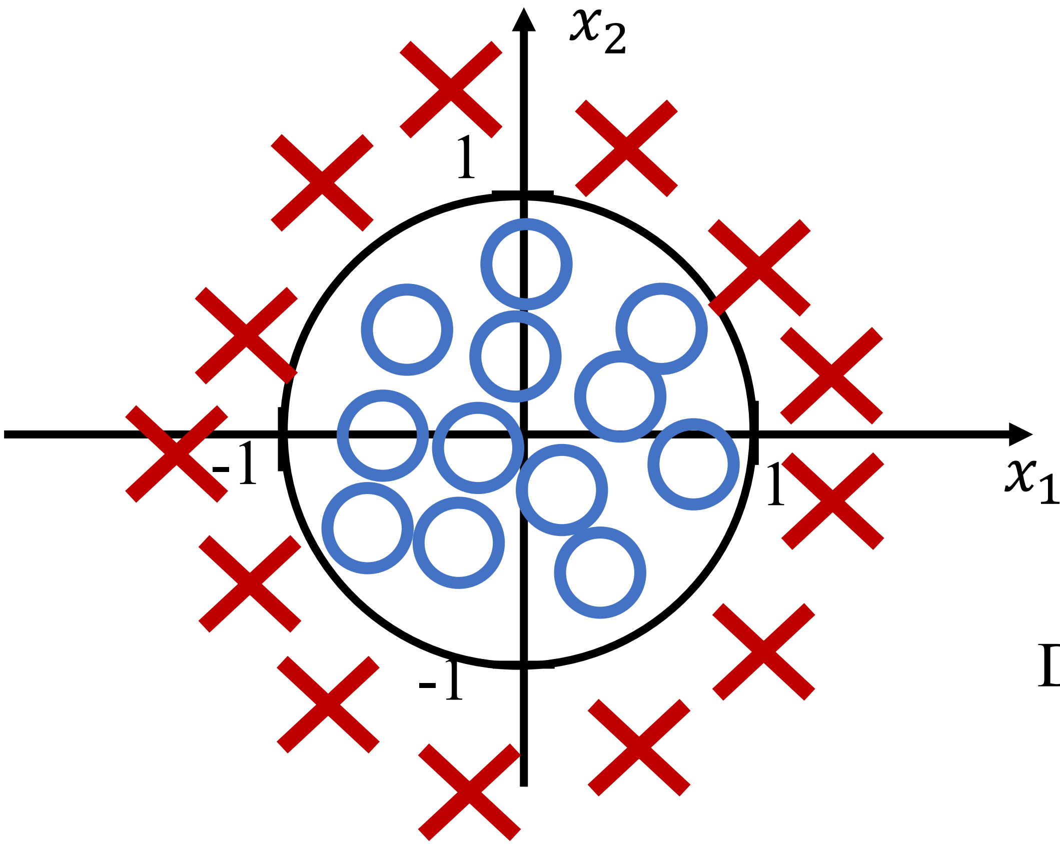
(set $w_1 = 1, w_2 = 1$)

$$z = x_1 + x_2 - 3 = 0$$

$$x_1 + x_2 = 3$$

Decision boundary is hyperplane $f(x) = 0.5 \;\rightarrow z = 0$

Non-linear Decision Boundary



$$z = w_1 x_1^2 + w_2 x_2^2 + b$$

Decision Boundary:

(set $w_1 = 1, w_2 = 1$)

$z = {x_1}^2 + {x_2}^2 - 1 = 0$

${x_1}^2 + {x_2}^2 = 1$

Loss Function

Training Set

tumor size(cm)	…	patient's age	malignant?
x_1		x_n	y
10		52	1
2		73	0
5		55	0
12		49	1
…		…	…

$i = 1,2,\cdots m$: number of training samples

$j = 1,2,\cdots n$: number of features

target y is 0 or 1

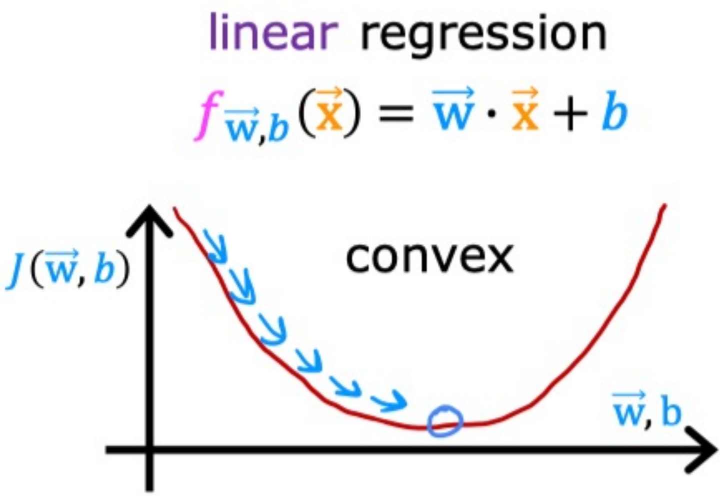
$$f_{w,b}(x) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

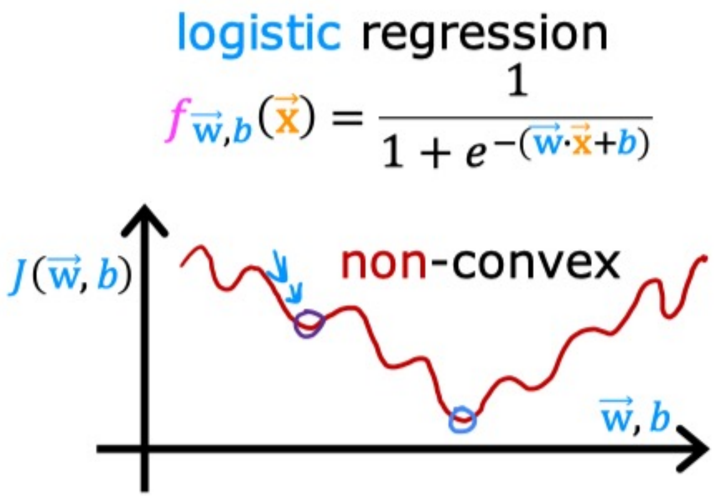
How to choose $w = [w_1, w_2, w_3, \cdots w_n]$ and b?

Loss Function

- Squared Error Cost:

$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)^2$$





- Differentiable => can use gradient descent
- Non-convex => not guaranteed to find the global optimum

Loss Function

- Logistic Loss Function:

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)), & if\ y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & if\ y^{(i)} = 0 \end{cases}$$

$if\ y^{(i)} = 1,$ As $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 1$, then loss $\to 0$
As $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 0$, then loss $\to \infty$

$if\ y^{(i)} = 0,$ As $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 1$, then loss $\to \infty$
As $f_{\vec{w},b}\left(\vec{x}^{(i)}\right) \to 0$, then loss $\to 0$

Simplified Loss Function

- Logistic Loss Function:

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = \begin{cases} -\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right), & if\ y^{(i)} = 1 \\ -\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) & if\ y^{(i)} = 0 \end{cases}$$

- Simplified Logistic Loss Function (Convex):

$$L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right) = -y^{(i)}\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) - \left(1 - y^{(i)}\right)\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)$$

- Overall:

$$\begin{aligned} J(\vec{w}, b) &= \frac{1}{m}\sum_{i=1}^{m}\left[L\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right), y^{(i)}\right)\right] \\ &= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)\right] \end{aligned}$$

Can be derived from Maximum Likelihood

Gradient Descent

- Overall Loss (Cost):

$$J(\vec{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)\right]$$

- Gradient Decent:

Repeat {

$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}\left(J(\vec{w}, b)\right),$$

where $\frac{\partial}{\partial w_j}\left(J(\vec{w}, b)\right) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$

$$b = b - \alpha\frac{\partial}{\partial b}\left(J(\vec{w}, b)\right),$$

where $\frac{\partial}{\partial b}\left(J(\vec{w}, b)\right) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)$

} simultaneous updates

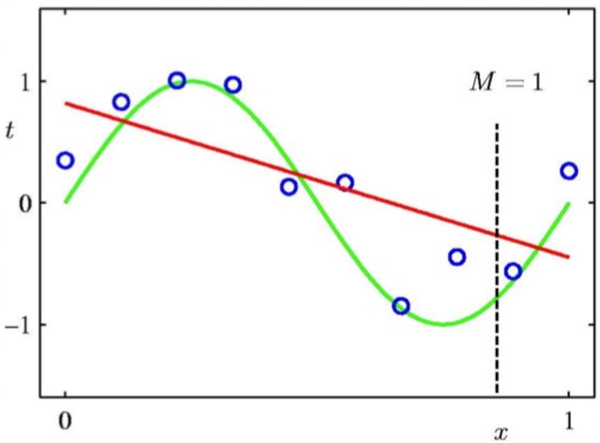
Compared with Linear Regression:

$$\frac{\partial E}{\partial w_j} = \sum_i (y_i - \mathbf{w}^\top \mathbf{x}_i)(-x_{ij})$$

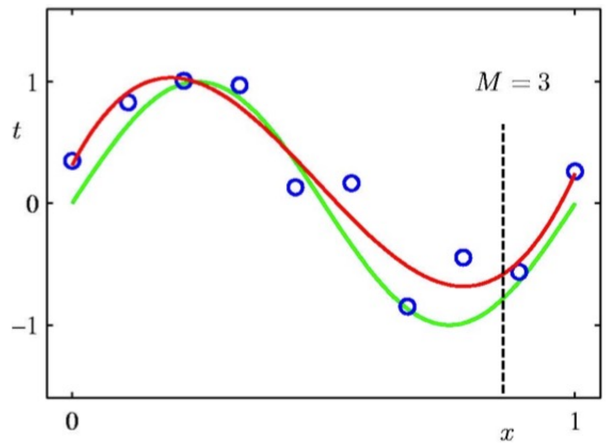
Bias & Variance

- Bias and Variance are two fundamental concepts in machine learning that pertain to the errors associated with predictive models.
- Bias: The differences between actual or expected values and the predicted values are known as bias error or error due to bias. Bias is a systematic error that occurs due to wrong assumptions in the machine learning process.
  - Low Bias: In this case, the model will closely match the training dataset.
  - High Bias: If a model has high bias, this means it can't capture the patterns in the data, no matter how much you train it. The model is too simplistic. This scenario is often referred to as underfitting.
- Variance: Variance is the amount by which the performance of a predictive model changes when it is trained on different subsets of the training data. More specifically, variance is the variability of the model that how much it is sensitive to another subset of the training dataset (i.e. how much it can adjust on the new subset of the training dataset).
  - Low Variance: Low variance means that the model is less sensitive to changes in the training data and can produce consistent estimates of the target function with different subsets of data from the same distribution.
  - High Variance: High variance means that the model is very sensitive to changes in the training data and can result in significant changes in the estimate of the target function when trained on different subsets of data from the same distribution.

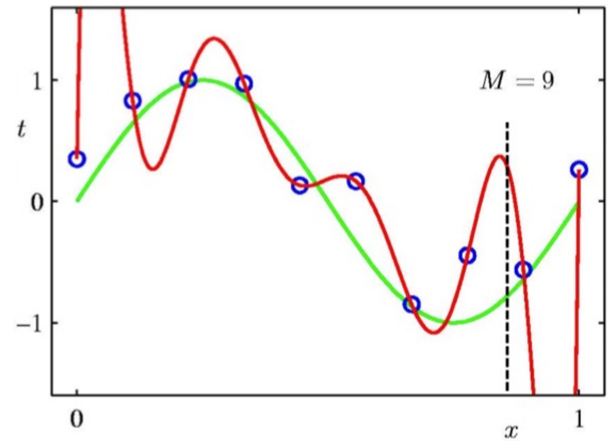
Polynomial Regression Examples



M=1



M=3



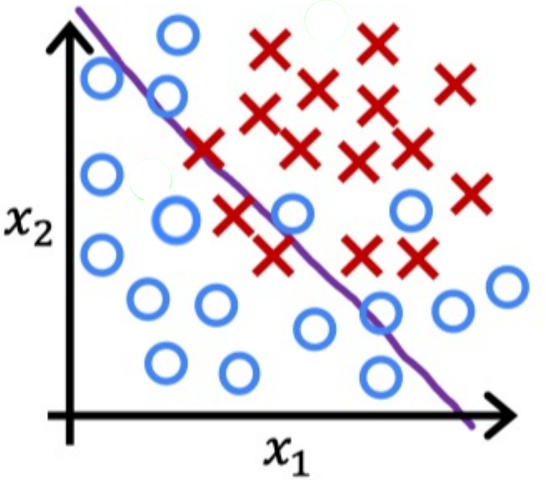
M=9

- **Underfitting**
- Does not fit the training set well
- Cannot fit the test set as well
- High bias

- **Just right**
- Fits training set pretty well
- Fits test set well
- Generalization

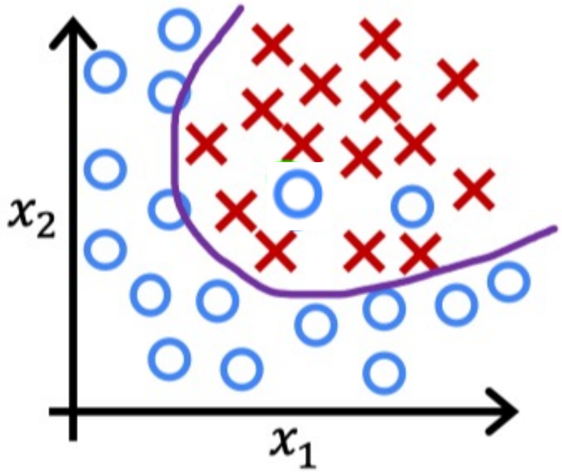
- **Overfitting**
- Fit the training set extremely well
- Cannot fit the test set as well
- High variance

Classification Examples



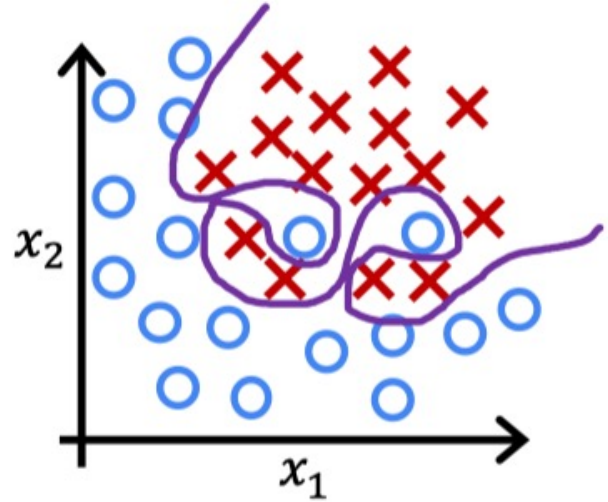
$$z = w_1 x_1 + w_2 x_2 + b$$

- **Underfitting**



$$z = w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + b$$

- **Just right**

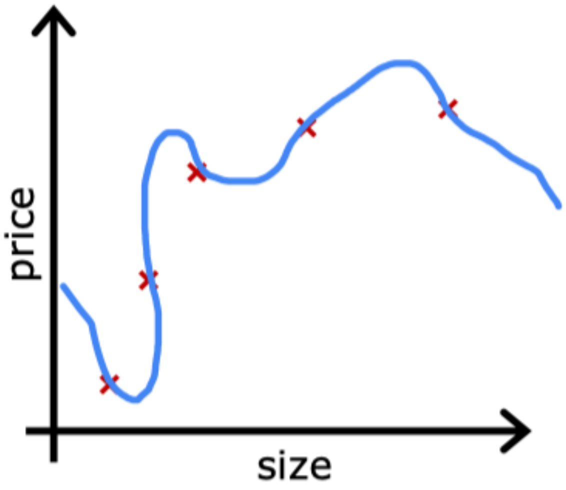


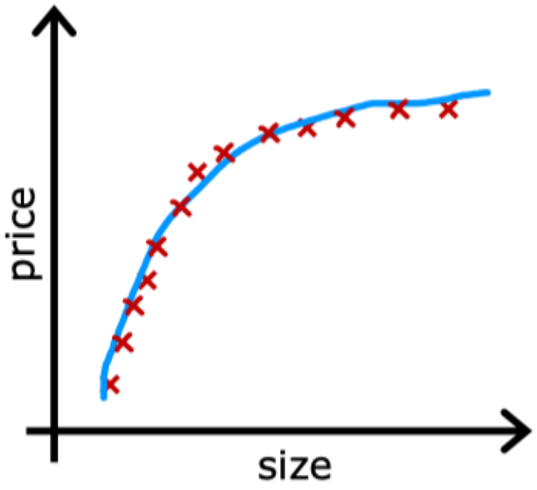
$$z = w_1 x_1^3 + w_2 x_2^3 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + \ldots + b$$

- **Overfitting**

Dealing with Overfitting

- Collect more training examples





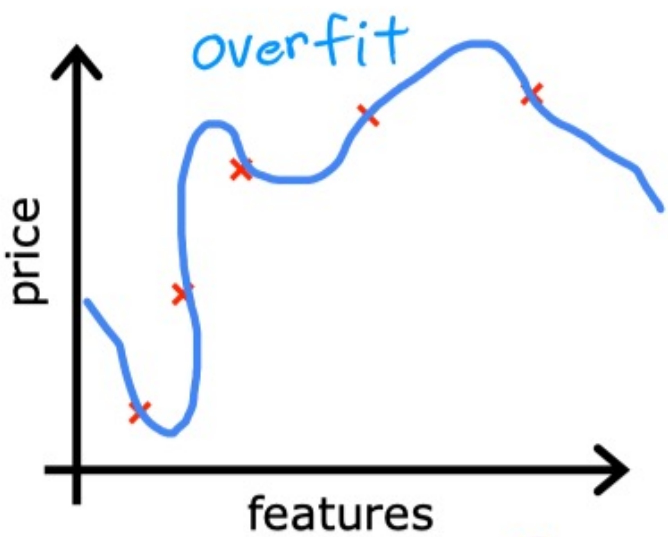
Dealing with Overfitting

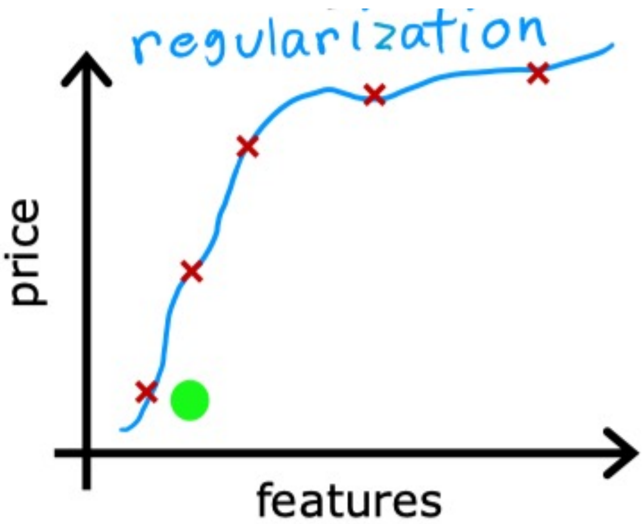
- Select features to include/exclude:
  - 100 features → 10 feature
  - 100 features + insufficient data → Overfitting
  - Just right 10 features + same data → Just right (possible)

- Disadvantage:
  - Useful features could be lost

Regularization

- Reduce the size of parameters **w**





$$f(x) = 28x - 385x^2 + 39x^3 - 174x^4 + 100$$

$$f(x) = 13x - 0.23x^2 + 0.000014x^3 - 0.0001x^4 + 10$$

Regularized Linear Regression

- Overall Loss with Regularizer:

$$J(\vec{w}, b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - f_{\vec{w},b}\left(\vec{x}^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} {w_j}^2$$

- Gradient Decent:

Repeat {

$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}\left(J(\vec{w}, b)\right),$$

where $\frac{\partial}{\partial w_j}\left(J(\vec{w}, b)\right) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right){x_j}^{(i)} + \frac{\lambda}{m}w_j$

$$b = b - \alpha\frac{\partial}{\partial b}\left(J(\vec{w}, b)\right),$$

where $\frac{\partial}{\partial b}\left(J(\vec{w}, b)\right) = \frac{1}{m}\sum_{i=1}^{m}\left(f_{\vec{w},b}\left(\vec{x}^{(i)}\right) - y^{(i)}\right)$

} simultaneous updates

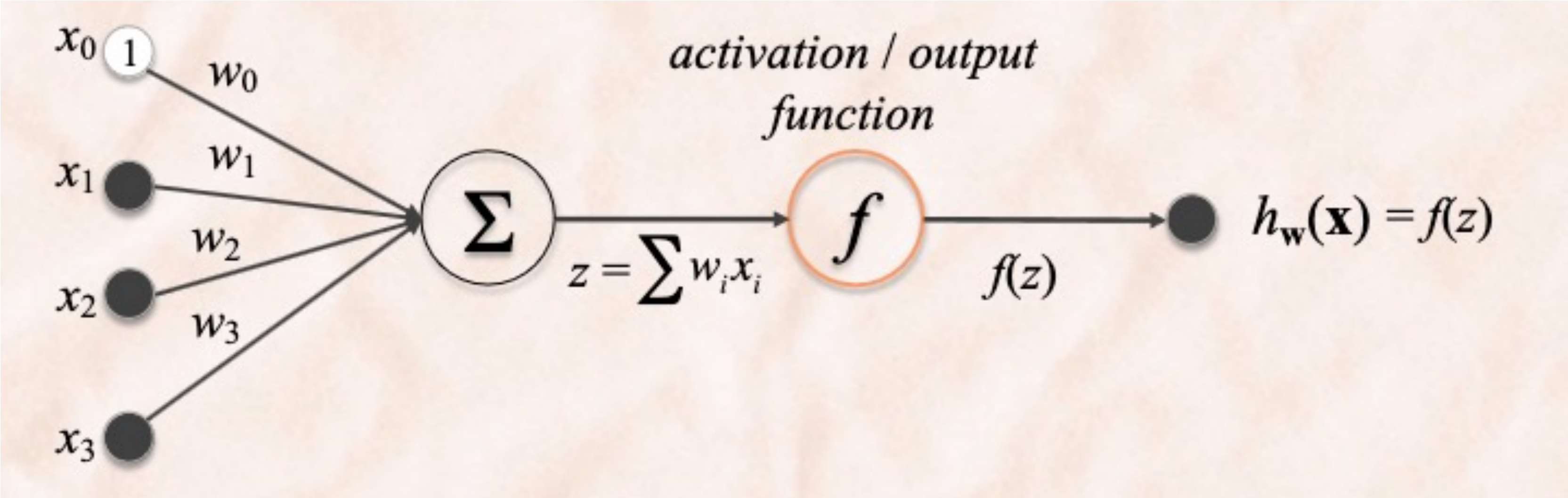
Machine Learning Objective

- Find a model **M**:
  - that ***fits the training data*** + that is ***simple***

$$\hat{\mathbf{M}} = \underset{\mathbf{M}}{\operatorname{argmin}} \; \mathit{Complexity}(\mathbf{M}) + \mathit{Error}(\mathbf{M}, \mathit{Data})$$

- **Inductive hypothesis**: Models that perform well on training examples are expected to do well on test (unseen) examples.
- **Occam's Razor**: Simpler models are expected to do better than complex models on test examples (assuming similar training performance).

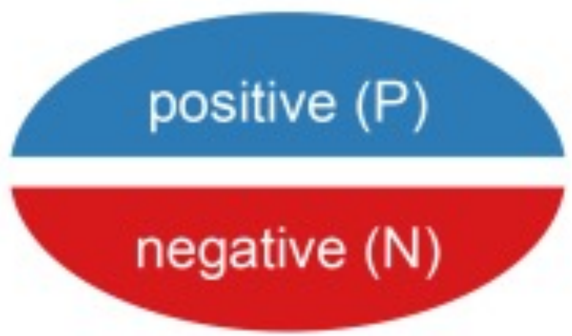
Algebraic Interpretation



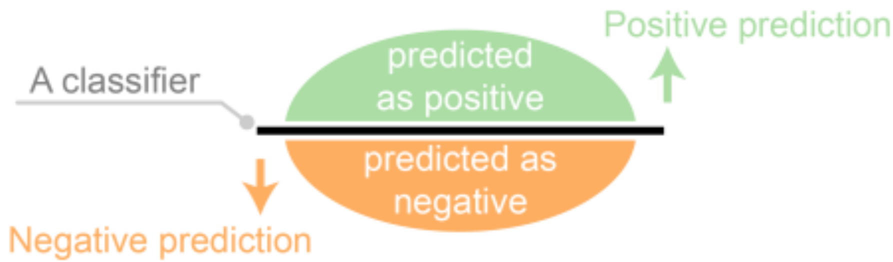
- The output of the neuron is a linear combination of inputs from other neurons, rescaled by the weights.
- summation corresponds to combination of signals
- It is often transformed through an **activation/output** function.

Binary Classification

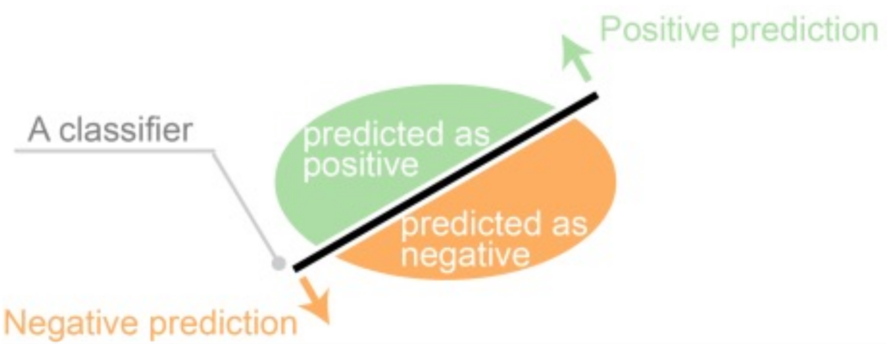
- Test dataset for evaluation:
  - In binary classification dataset, each instance will have its true label (true class): Positive Class (P) vs Negative Class (N).



- Predictions on test dataset:
  - A perfect classifier



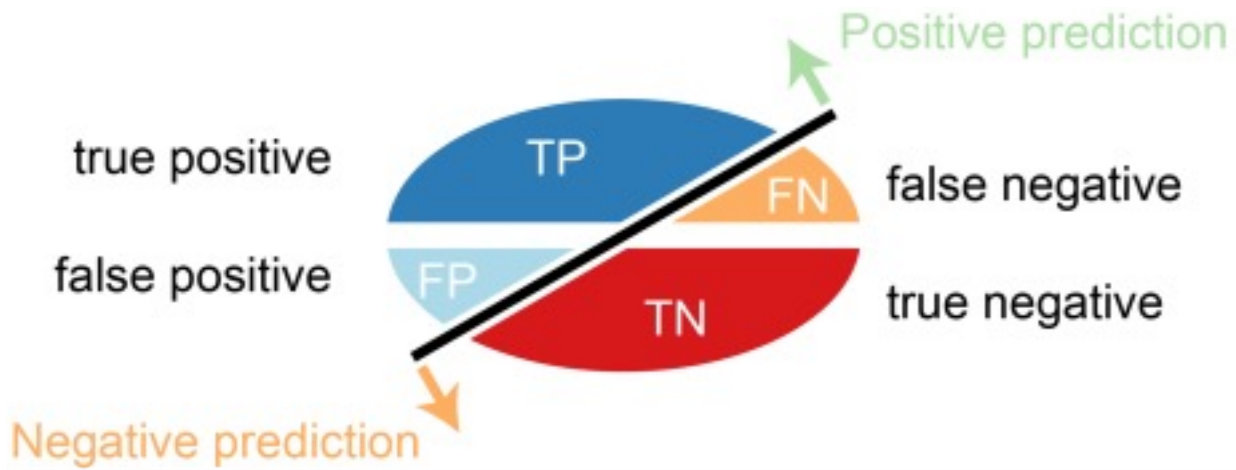
  - A real-world classifier



Images from https://classeval.wordpress.com/introduction/basic-evaluation-measures/

Confusion Matrix

- **Confusion matrix** (a 2x2 table) is composed of four outcomes of classification:
  - True positive (TP): correct positive prediction
  - False positive (FP): incorrect positive prediction
  - True negative (TN): correct negative prediction
  - False negative (FN): incorrect negative prediction



True	Prediction	Positive	Negative
Positive		# of TPs	# of FNs
Negative		# of FPs	# of TNs

Basic Measurements

- **Accuracy** is calculated as the number of all correct predictions divided by the total number of the dataset.

Accuracy: (TP + TN) / (P + N)

- **Recall** (sensitivity, true positive rate) is calculated as the number of correct positive predictions divided by the total number of positives.

Sensitivity: TP / P

- **Precision** is calculated as the number of correct positive predictions divided by the total number of positive predictions.

Precision: TP / (TP + FP)

- **F1 Score** is a harmonic mean of precision and recall.

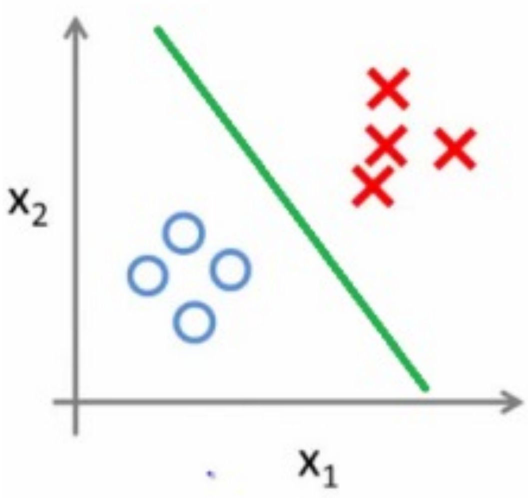
$$F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{2\mathrm{tp}}{2\mathrm{tp} + \mathrm{fp} + \mathrm{fn}}$$

Multi-class Classification

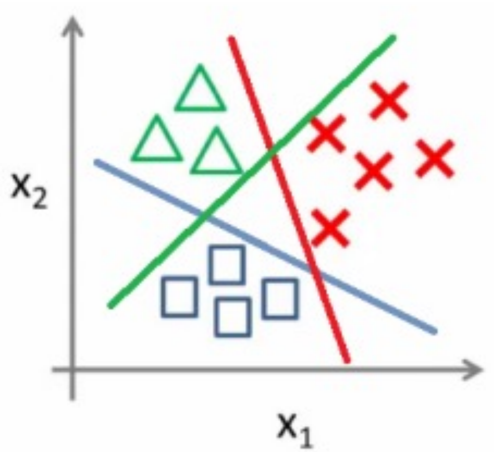
- **Multi-class Classification:**
  - To classify instances into one of more than two classes. (i.e., there are more than two possible categories or labels)

- **Strategies:**
  - One-vs-All (One-vs-Rest)
  - One-vs-One
  - Softmax Regression (Later)
  - Decision Trees (Later)

- Binary classification:



- Multi-class classification:

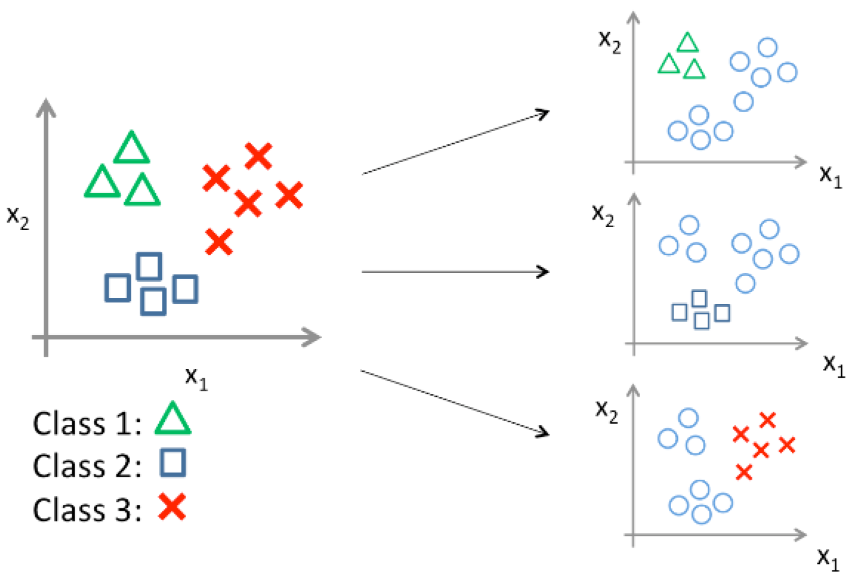


Images from: https://utkuufuk.com/2018/06/03/one-vs-all-classification/

One-vs-All

- **One-vs-all classification** breaks down N classes present in the dataset into N binary classifier models that aims to classify a data point as either part of the current class or not.

- Suppose you have classes 1, 2, and 3.
  - Model A: 1 or 2,3 (1 or not 1)
  - Model B: 2 or 1,3 (2 or not 2)
  - Model C: 3 or 1,2 (3 or not 3)



- At prediction time, the class that corresponds to the classifier with the highest confidence score is the predicted class.
  - Model A: $P(x = 1)$ and $P(x \neq 1)$
  - Model B: $P(x = 2)$ and $P(x \neq 2)$
  - Model C: $P(x = 3)$ and $P(x \neq 3)$
  - Among $P(x = 1)$, $P(x = 2)$, and $P(x = 3)$, which one is the highest?

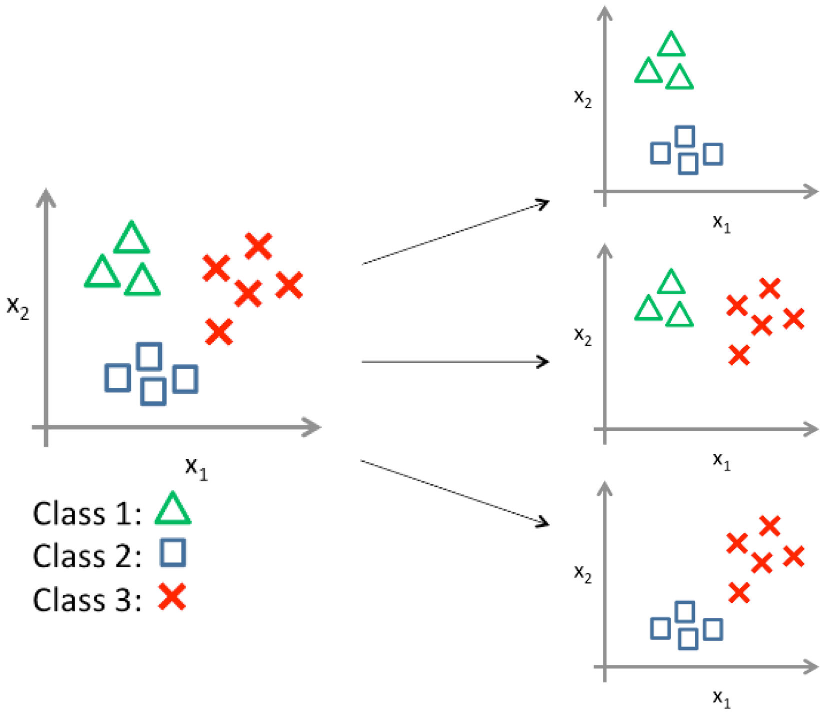
Images from: https://www.cc.gatech.edu/classes/AY2016/cs4476_fall/results/proj4/html/jnanda3/index.html

One-vs-one

- **One-vs-one classification** breaks down *N* classes present in the dataset into ***N*(N-1)/2*** binary classifier models – one for each pair of classes.

- Suppose you have classes 1, 2, and 3.
  - Model A: 1 or 2
  - Model B: 1 or 3
  - Model C: 2 or 3

- At prediction time, each classifier votes for a class, and the class with the most votes is the predicted class.
  - Model A: Vote for 1 or 2
  - Model B: Vote for 1 or 3
  - Model C: Vote for 2 or 3
  - Classes 1, 2, and 3, which one has the most votes?



Questions?

UNIVERSITY OF NORTH CAROLINA
CHARLOTTE