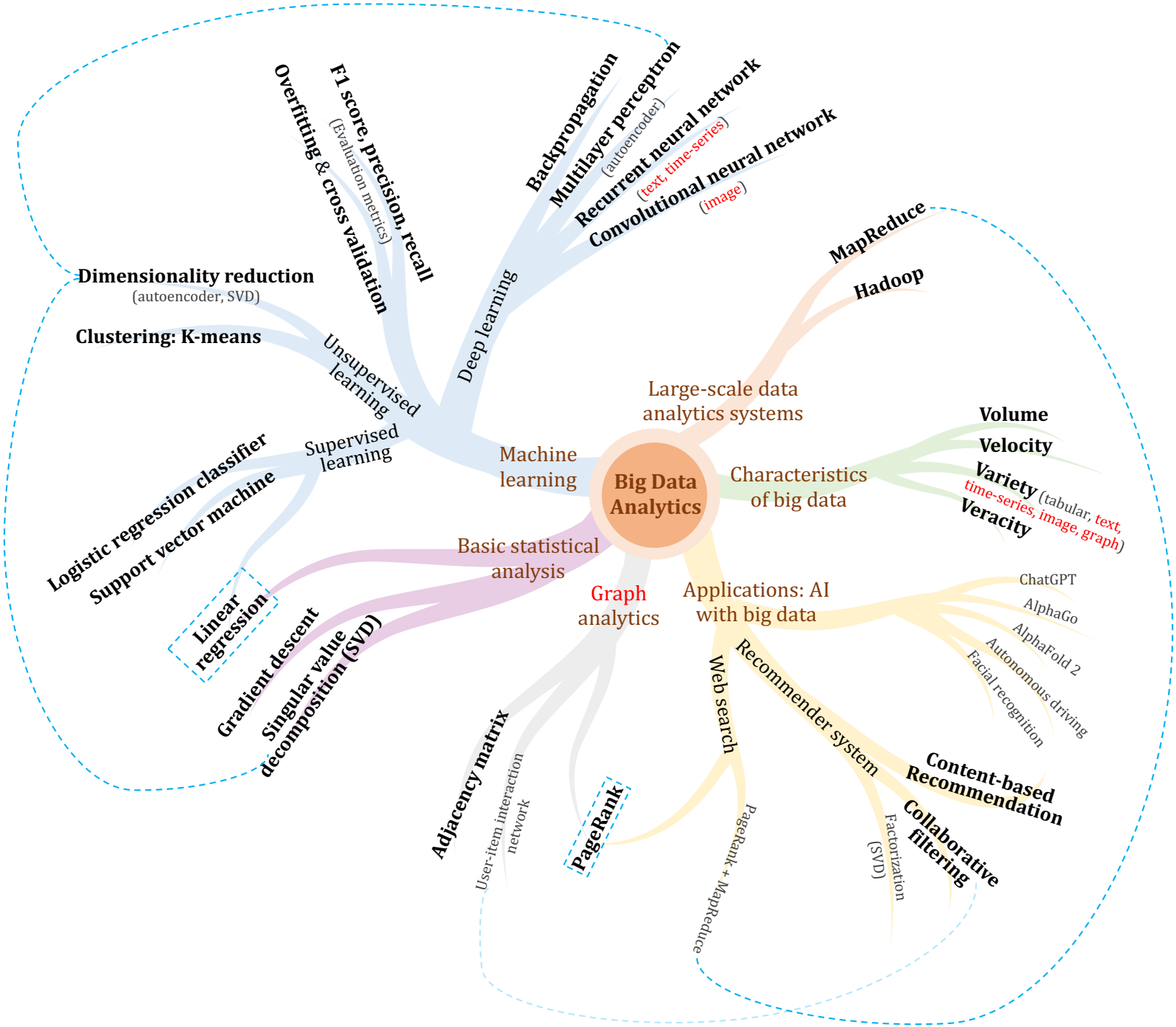
THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

COMP4434 Big Data Analytics

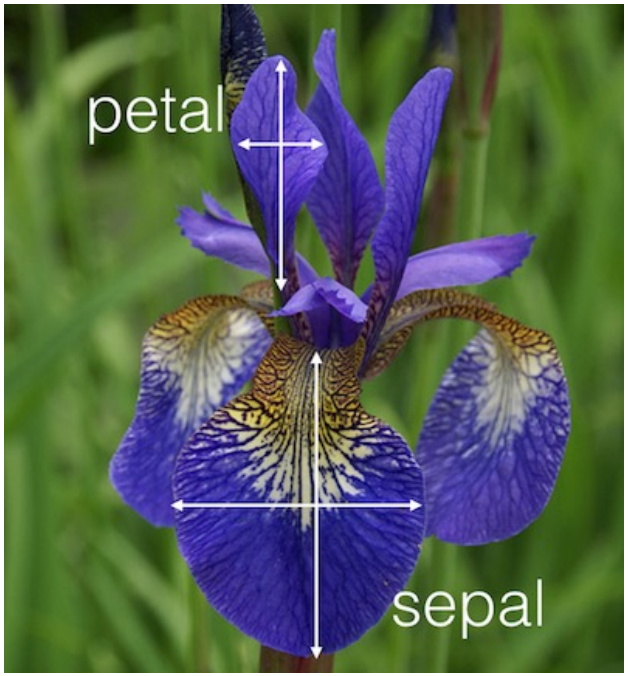
Lecture 3 Logistic Regression Classifier, F1 Score, & Overfitting

HUANG Xiao

xiaohuang@comp.polyu.edu.hk


Big Data Analytics
Machine learning
Deep learning
Backpropagation
Multilayer perceptron
(autoencoder)
Recurrent neural network
(text, time-series)
Convolutional neural network
(image)
Unsupervised learning
Dimensionality reduction
(autoencoder, SVD)
Clustering: K-means
Supervised learning
F1 score, precision, recall
(Evaluation metrics)
Overfitting & cross validation
Logistic regression classifier
Support vector machine
Linear regression
Basic statistical analysis
Gradient descent
Singular value decomposition (SVD)
Large-scale data analytics systems
MapReduce
Hadoop
Characteristics of big data
Volume
Velocity
Variety (tabular, text, time-series, image, graph)
Veracity
Applications: AI with big data
ChatGPT
AlphaGo
AlphaFold 2
Autonomous driving
Facial recognition
Recommender system
Content-based Recommendation
Collaborative filtering
Factorization (SVD)
Web search
PageRank + MapReduce
Graph analytics
PageRank
Adjacency matrix
User-item interaction network

Structured Data: Tabular Data



Sepal		Petal	
Length	Width	Length	Width
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2
5.8	4.0	1.2	0.2
5.6	3.0	4.5	1.5
5.8	2.7	4.1	1.0
6.2	2.2	4.5	1.5
5.6	2.5	3.9	1.1
5.9	3.2	4.8	1.8
7.1	3.0	5.9	2.1
6.3	2.9	5.6	1.8
6.5	3.0	5.8	2.2
7.6	3.0	6.6	2.1

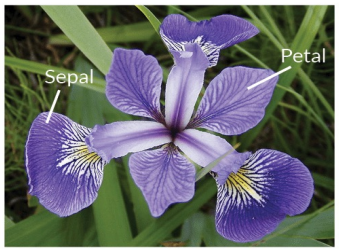
Instance/tuple

attribute/feature

Tabular Data with Labels

- Attributes of a table $A_1, A_2, \ldots, A_d$
  - We also call "**attribute**" as "**feature**"
  - Number of features d represents the dimensionality
  - A data object x is represented as $(x_1, x_2, \ldots, x_d)$
- In some datasets, each data object has a **label**

Sepal		Petal		Class Label
Length	Width	Length	Width	
4.9	3.0	1.4	0.2	Setosa
4.7	3.2	1.3	0.2	Setosa
4.6	3.1	1.5	0.2	Setosa
5.0	3.6	1.4	0.2	Setosa
5.8	4.0	1.2	0.2	Setosa
5.6	3.0	4.5	1.5	Versicolor
5.8	2.7	4.1	1.0	Versicolor
6.2	2.2	4.5	1.5	Versicolor
5.6	2.5	3.9	1.1	Versicolor
5.9	3.2	4.8	1.8	Versicolor
7.1	3.0	5.9	2.1	Virginica
6.3	2.9	5.6	1.8	Virginica
6.5	3.0	5.8	2.2	Virginica
7.6	3.0	6.6	2.1	Virginica
4.9	2.5	4.5	1.7	Virginica



Iris Versicolor


Iris Setosa

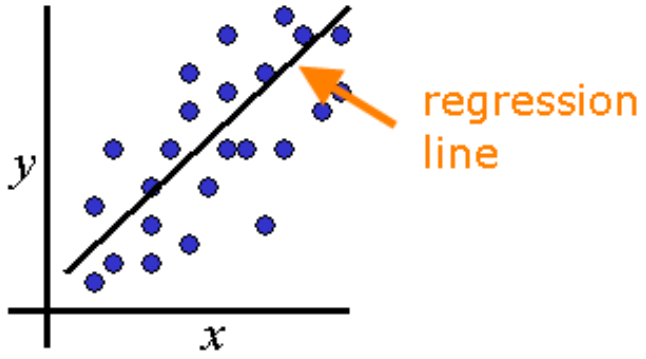

Iris Virginica

Classification (Binary vs Multi-class)

- Binary Classification
  - Email: Spam / Not Spam
  - Tumor: Malignant / Benign
  - Covid-19: Positive / Negative
  - $y \in \{0,1\}$
- Multi-class Classification
  - Email auto-tagging: Spam / Work / Personal
  - Credit Rating: Poor / Okay / Trust
  - Handwriting number: 0, 1, 2, 3, 4, ...
  - $y \in \{0,1,2, \dots\}$

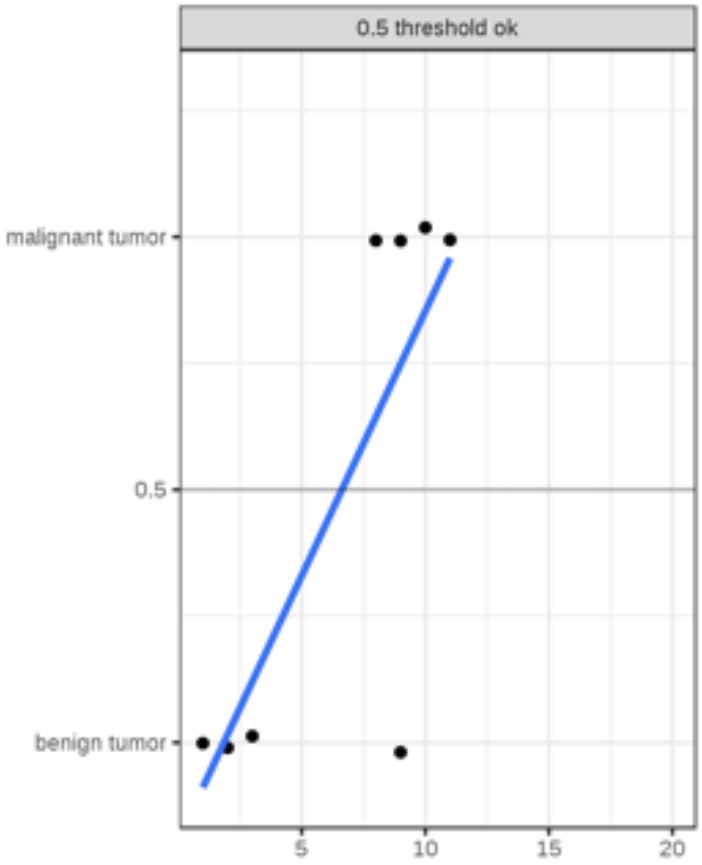
Classification by Linear Regression

- Training Examples
  - both are supervised learning (x, y)
  - Linear Regression: y is a real-value, e.g., salary
- What we need is discrete label:
  - 0: malignant; 1: benign
- Can we use Linear Regression Model to do classification? Any disadvantages?
  - Yes, we can, but not good
  - $h_\theta(x) = \theta_0 + \theta_1 x$



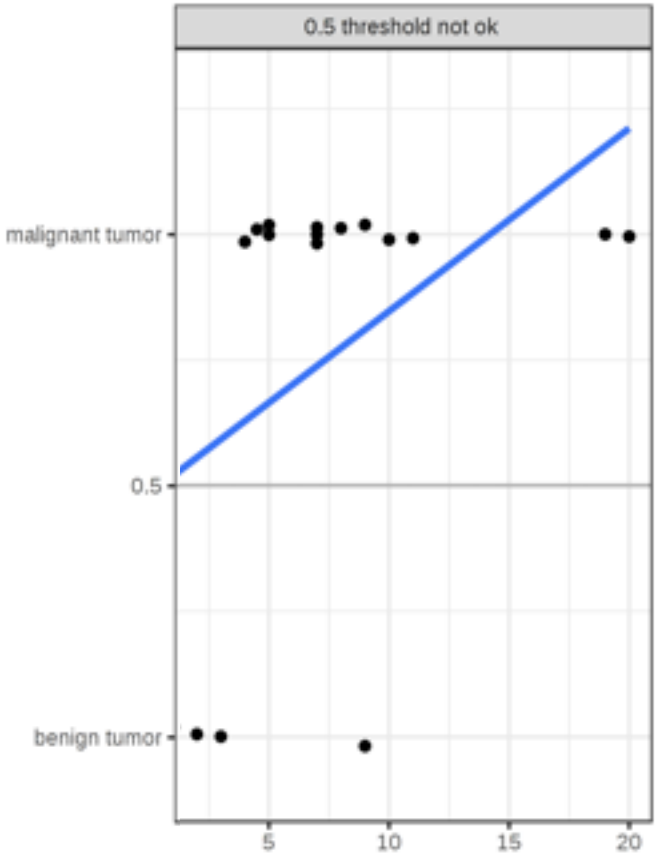
Classification by Linear Regression - Example

- A linear regression model classifies tumors as malignant (1) or benign(0) given their size
- The linear regression model minimizes the distances between the points and the hyperplane (line for single feature)
- The threshold is set as 0.5



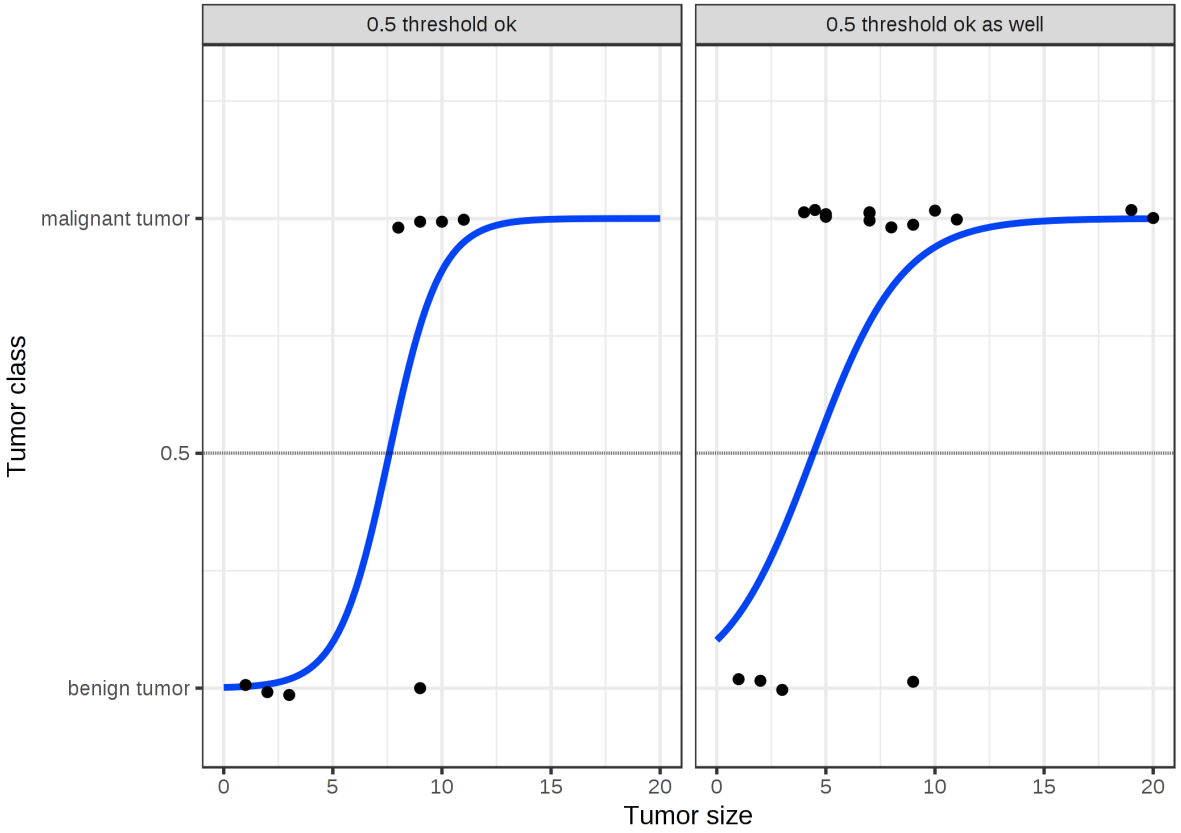
Classification by Linear Regression - Example

- After introducing a few more malignant tumor cases, the regression line shifts and a threshold of 0.5 no longer separates the classes
- **Conclusion: Linear regression is sensitive to imbalance data for classification problem.**



Logistic Regression

- New model outputs **probabilities**
- It works better in both cases using 0.5 as a threshold
- The inclusion of additional points does not affect the estimated curve too much



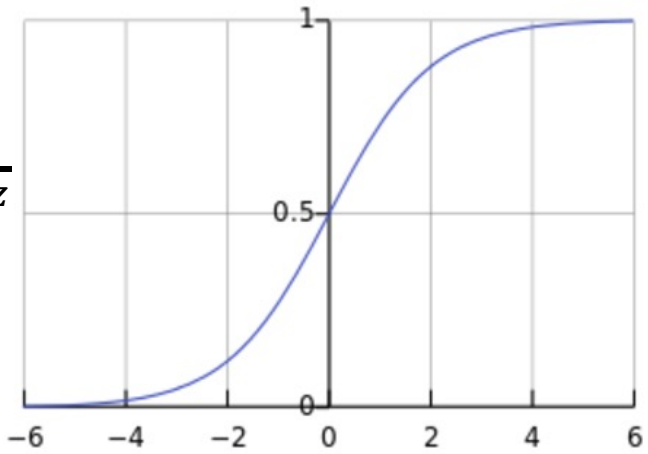
Hypothesis Model

- Linear Regression: $-\infty < h_\theta(x) < +\infty$

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n = \theta^T x$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

- Logistic Regression: $0 < h_\theta(x) < 1$
  - $h_\theta(x) = g(\theta^T x)$
  - logistic/sigmoid function $g(z) = \frac{1}{1+e^{-z}}$
  - $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$
  - https://www.wolframalpha.com/
    y = 1/(1+e^-x), x from -6 to 6



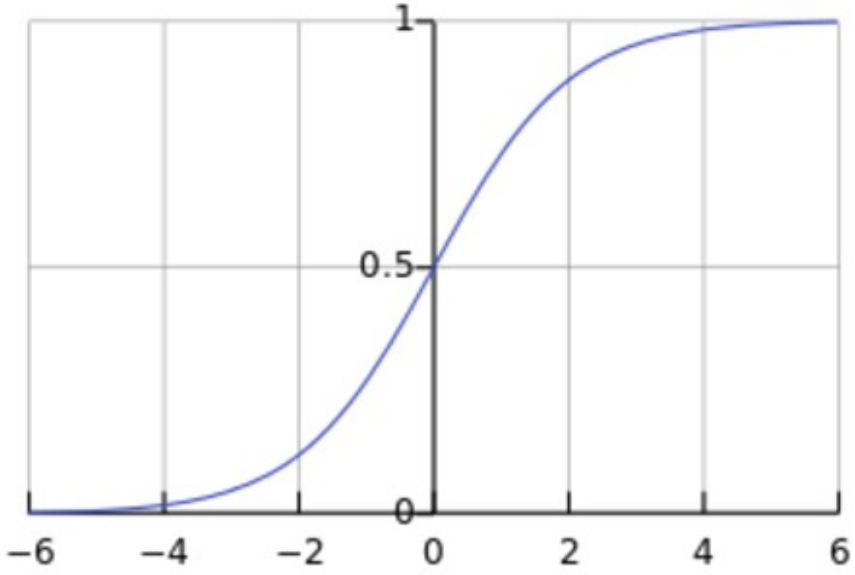
Representation

- $h_\theta(x)$ represents the estimated **probability** that $y = 1$ on input x
- $h_\theta(x) = P\{y = 1|x, \theta\}$ means probability of $y = 1$, given x, under parameter values θ
- $P\{y = 0|x, \theta\} = 1 - h_\theta(x)$

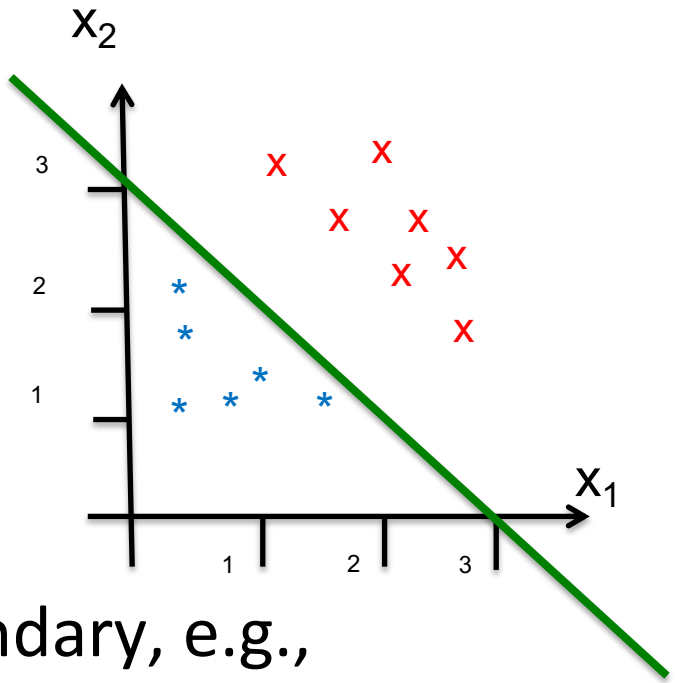
- Example
  - $\begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ number\ of\ spam\ words \end{bmatrix}$
  - $h_\theta(x) = 0.8$: this email x has 80% chance of being spam

Further Understanding

- $h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \in (0,1)$
- Predict $y = 1$ when $h_\theta(x) \geq 0.5$, i.e., $\theta^T x \geq 0$
- Predict $y = 0$ when $\theta^T x < 0$



Decision Boundary



- $\theta^T x = 0$ is the decision boundary, e.g.,
  - Assume $h_\theta(x) = g(-3 + x_1 + x_2)$
  - Decision boundary: $-3 + x_1 + x_2 = 0$, i.e., $x_1 + x_2 = 3$
  - Predict $y = 1$ when $-3 + x_1 + x_2 \geq 0$, i.e., $x_1 + x_2 \geq 3$ (red)

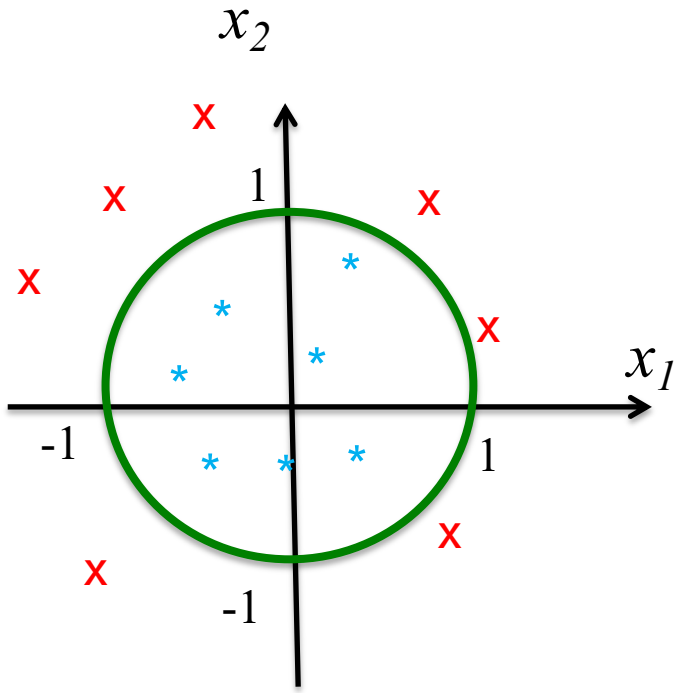
Other Decision Boundary

- Given $h_\theta(x) = g(\theta^T x) = g(\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2) = g(-1 + x_1^2 + x_2^2)$

- If $h_\theta(x) = g(-1 + x_1^2 + x_2^2)$, draw the region that predicts $y = 1$ in the (x_1, x_2) plane

Non-Linear Decision Boundary

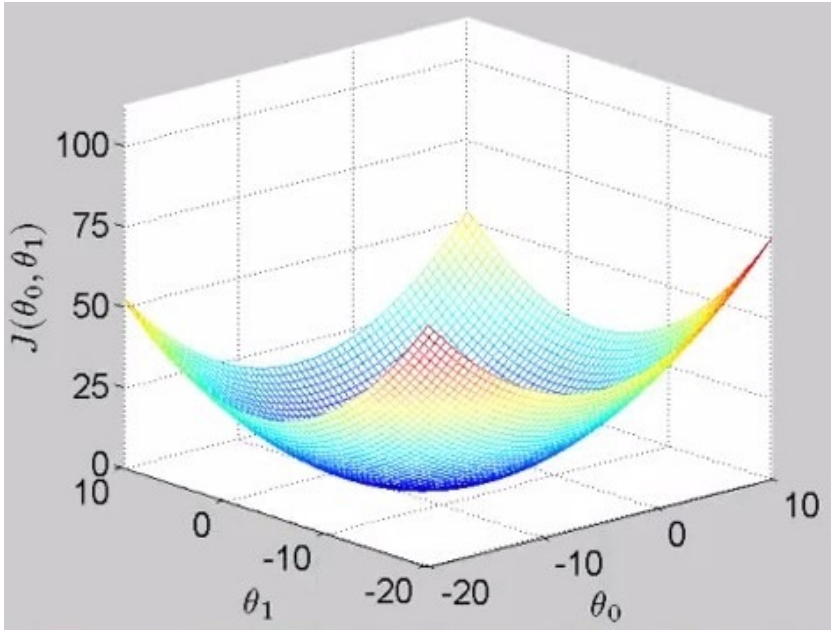
- Decision boundary $\theta^T x = 0 \rightarrow x_1^2 + x_2^2 = 1$
- Predict $y = 1$ when $x_1^2 + x_2^2 \geq 1$
- The region of $y = 1$ is outside the circle



Cost Function of Linear Regression

- Cost Function $J(\theta) = \min_{\theta_0,\theta_1,\ldots} \frac{1}{2m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

- Linear Regression
  - h_θ is **linear**
  - $J(\theta)$ is **convex**
  - $J(\theta)$ has a **single minimum**



Regression Metrics

- MSE: Mean Square Error

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_t - y_t)^2$$

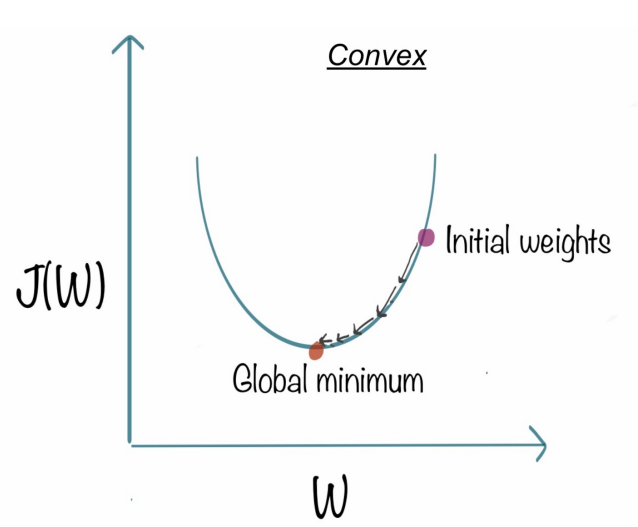
- MAE: Mean Absolute Error

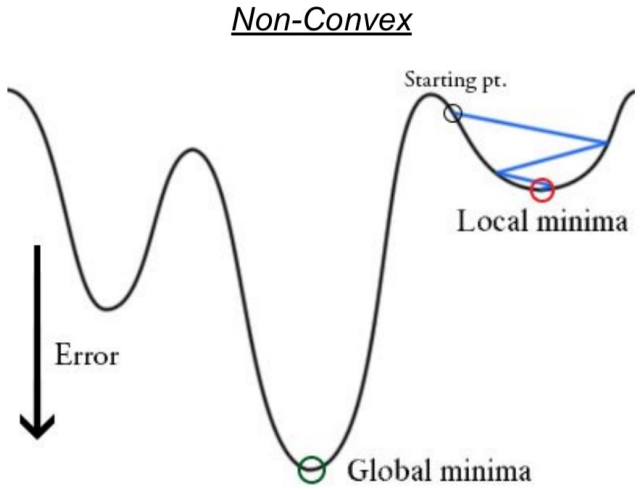
$$\frac{1}{n}\sum_{i=1}^{n}|\hat{y}_t - y_t|$$

- MAPE (Mean Absolute Percentage Error)

$$\frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_t - y_t}{y_t}\right|$$

Apply MSE to Logistic Regression





- We can apply the same cost function for logistic regression
- Problems
  - $J(\theta)$ would become non-convex. Why? https://towardsdatascience.com/why-not-mse-as-a-loss-function-for-logistic-regression-589816b5e03c
  - It has multiple local minimums
  - Gradient descent will be stuck in a local minimum

Logistic Loss

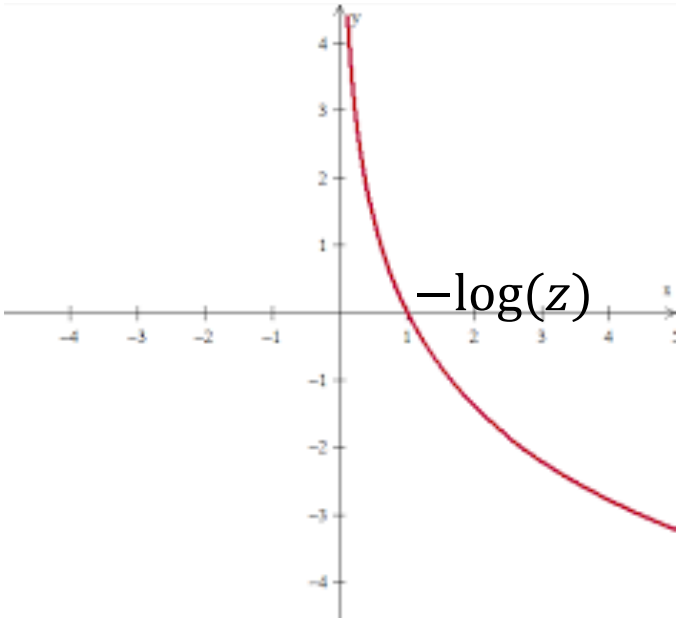
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right)$$

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & y = 1 \\ -\log(1 - h_\theta(x)) & y = 0 \end{cases}$$

Logistic Loss - Heavy Penalty

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & y = 1 \\ -\log(1 - h_\theta(x)) & y = 0 \end{cases}$$

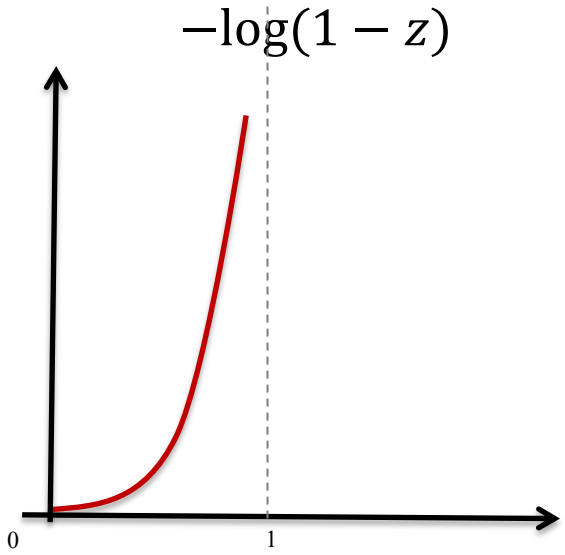
- $Cost(h_\theta(x), y) \to 0$
  - When $h_\theta(x) \to 1$, i.e., predict $y = 1$
  - Good predication, low cost
- $Cost(h_\theta(x), y) \to \infty$
  - When $h_\theta(x) \to 0$, i.e., predict $y = 0$
  - Bad predication!
  - High cost represents penalty



Understanding $Cost(h_\theta(x), y = 0)$

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & y = 1 \\ -\log(1 - h_\theta(x)) & y = 0 \end{cases}$$

- $Cost(h_\theta(x), y) \to 0$
  - When $h_\theta(x) \to 0$, i.e., predict $y = 0$
  - Good predication, low cost
- $Cost(h_\theta(x), y) \to \infty$
  - When $h_\theta(x) \to 1$, i.e., predict $y = 1$
  - Bad predication!
  - High cost represents penalty



Cost Function

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

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & y = 1 \\ -\log(1 - h_\theta(x)) & y = 0 \end{cases}$$

$$Cost(h_\theta(x), y) = -y \log h_\theta(x) - (1 - y)\log(1 - h_\theta(x))$$

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

Gradient Descent Algorithm

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

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

Repeat until convergence {

$$\theta_j = \theta_j - \alpha\frac{\partial J(\theta_0, \theta_1, \cdots)}{\partial \theta_j}$$

}

$$\frac{\partial J(\theta_0, \theta_1, \cdots)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

Looks identical to linear regression!

Classification Metrics

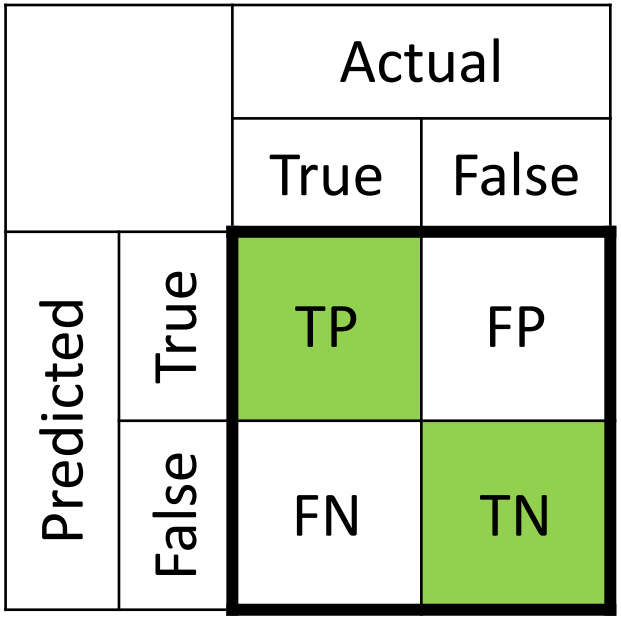
- True Positives (**TP**): the actual class of the data point was True and the predicted is also True
- True Negatives (**TN**): the actual class of the data point was False and the predicted is also False
- False Positives (**FP**): the actual class of the data point was False and the predicted is True
- False Negatives (**FN**): the actual class of the data point was True and the predicted is False

		Actual	
		True	False
Predicted	True	TP	FP
	False	FN	TN

Accuracy

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

- A good measure when the target variable classes in the data are nearly balanced
  - 60% classes in our fruit images are apples and 40% are oranges
- NEVER used as a measure when the target variable classes in the data are a majority of one class (Why?)



Limitation of Accuracy

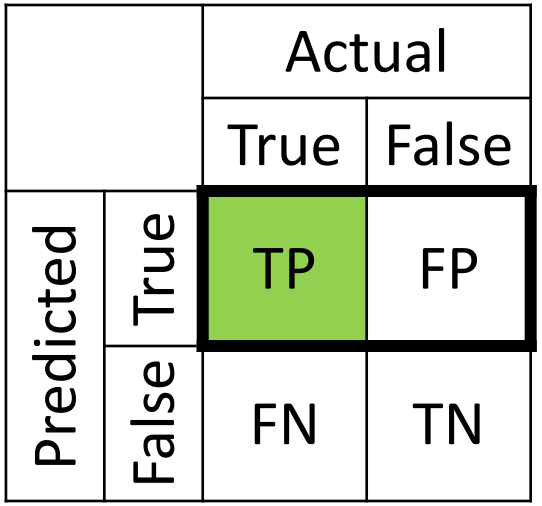
$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

- Example: In daily life, 5 people in 100 people have cancer.
- Consider a fake cancer detection model **only outputs 'health'**, its accuracy can achieve **95%**.
- Although its accuracy is good, is it a good model? NO.
- Observation: Accuracy performs bad when the target variable classes in the data are a majority of one class.

Precision and Recall

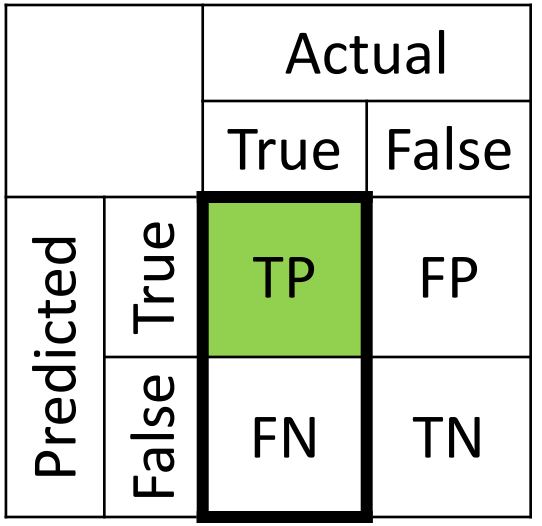
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- Precision is about being precise; even if we managed to capture only one True case, and we captured it correctly, then we are 100% precise



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

- Recall is not so much about capturing cases correctly but more about capturing all True cases



F1 Score

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

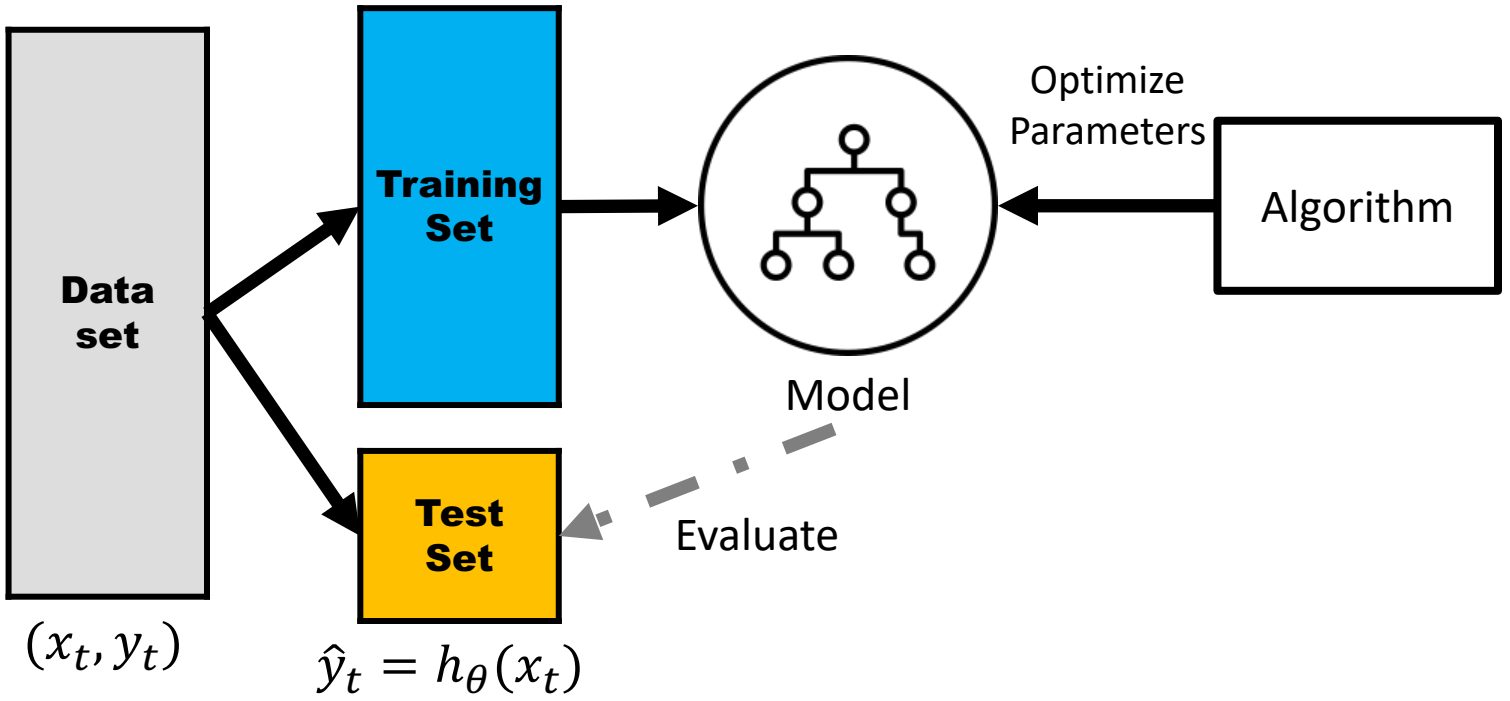
- A single score that represents both Precision and Recall
- F1 Score is the harmonic mean of Precision and Recall
- Different with arithmetic mean, harmonic mean is closer to the smaller number as compared to the larger number
- If Precision is 0.01 and Recall is 0.99, then arithmetic mean is 0.5 and harmonic mean is 0.0198
- Therefore, F1 score of the previous cancer detection model will be 0 (“positive” refers to having cancer)

Exercise

- Assume that there are 1,000 documents in total. Among them, 700 documents are related to big data analysis. You build a model to identify documents related to big data analysis. As a result, your model returns 800 documents, but only 550 of them are relevant to big data analysis. What is the recall of your model? What is the F1 score of your model?

- TP = 550. FP = 250. FN = 150
- Precision = 550/800.
- Recall = 550/700.
- F1 = (2*550/800*550/700)/(550/800+550/700) = 0.733333333.

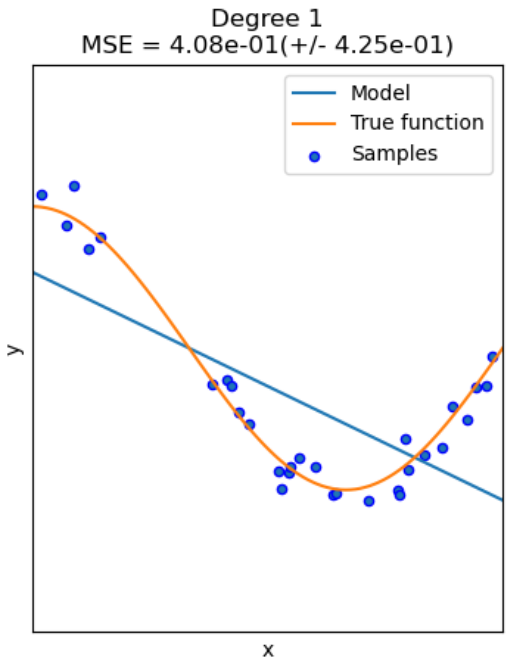
Model Evaluation

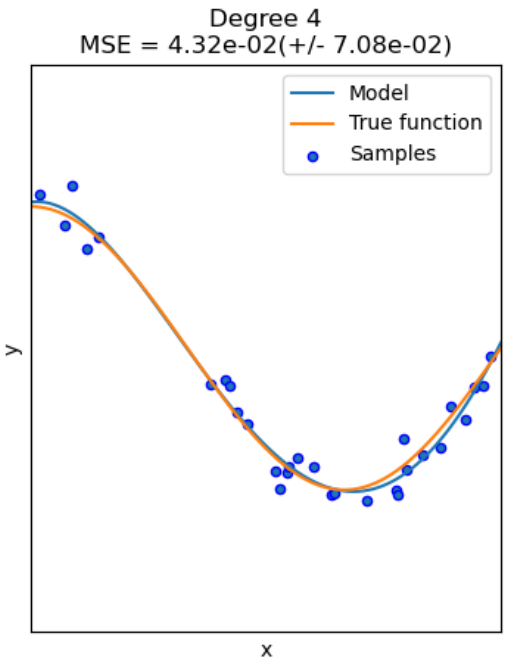


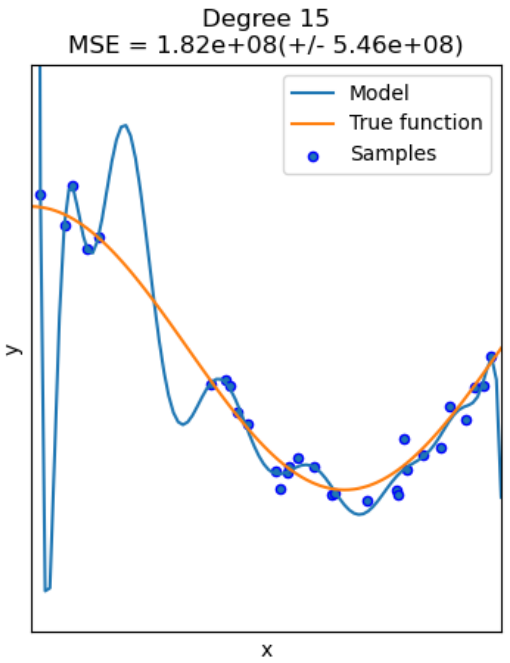
- When training the model, we can not use test set
- If we have several models, e.g., linear regression and quadratic regression, how could we evaluate them?

Underfitting and Overfitting

- Polynomial Regression with Degree = 4:
  - $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$



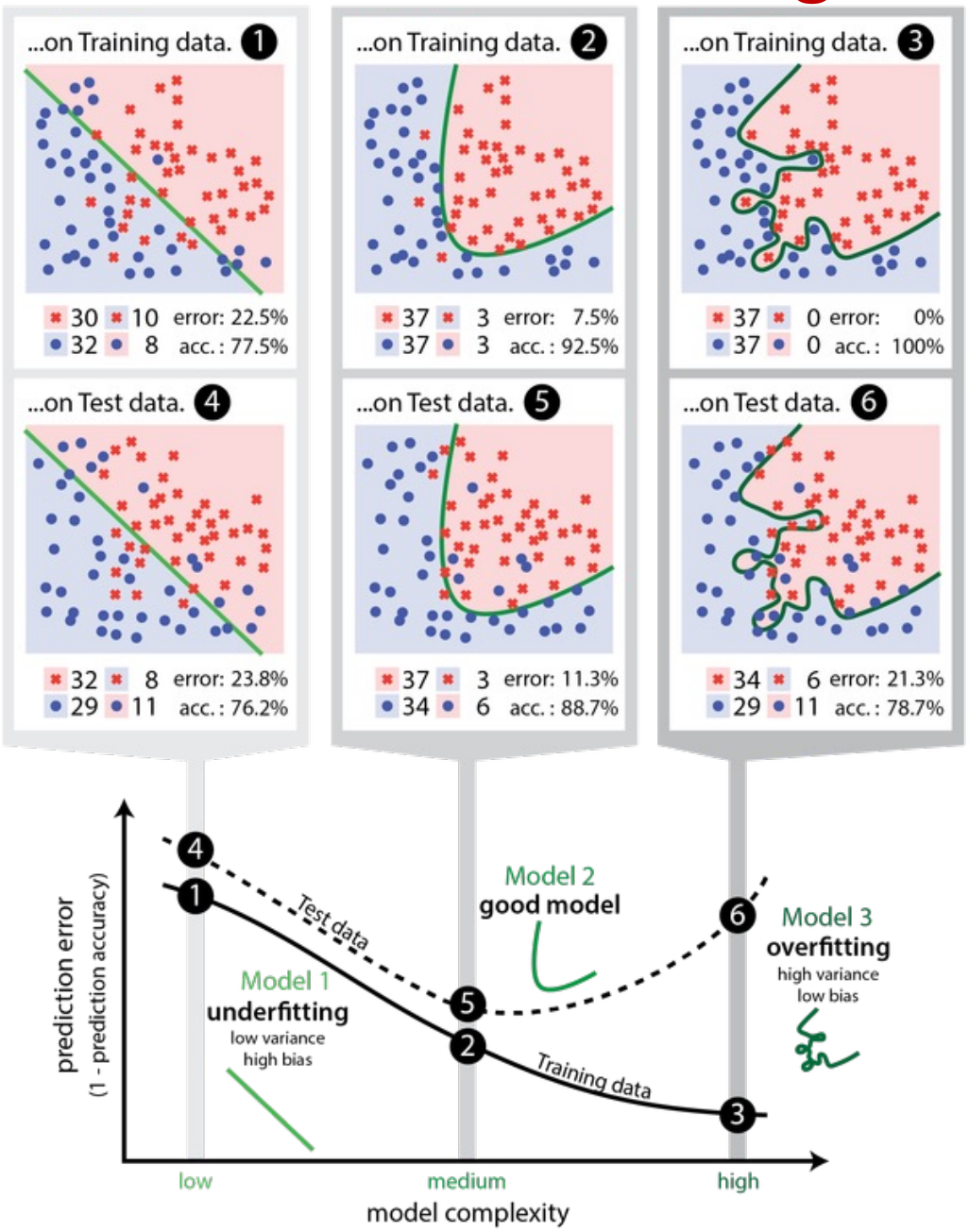





```
[-1.60931179]
[  0.46754142 -17.78954475  23.5926603   -7.26289872]
[-2.98293201e+03  1.03899495e+05 -1.87416410e+06  2.03716626e+07
 -1.44873631e+08  7.09317305e+08 -2.47066542e+09  6.24563114e+09
 -1.15676921e+10  1.56895529e+10 -1.54006639e+10  1.06457712e+10
 -4.91379699e+09  1.35920272e+09 -1.70381606e+08]
```

https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

Underfitting and Overfitting



Two classes separated by an elliptical arc

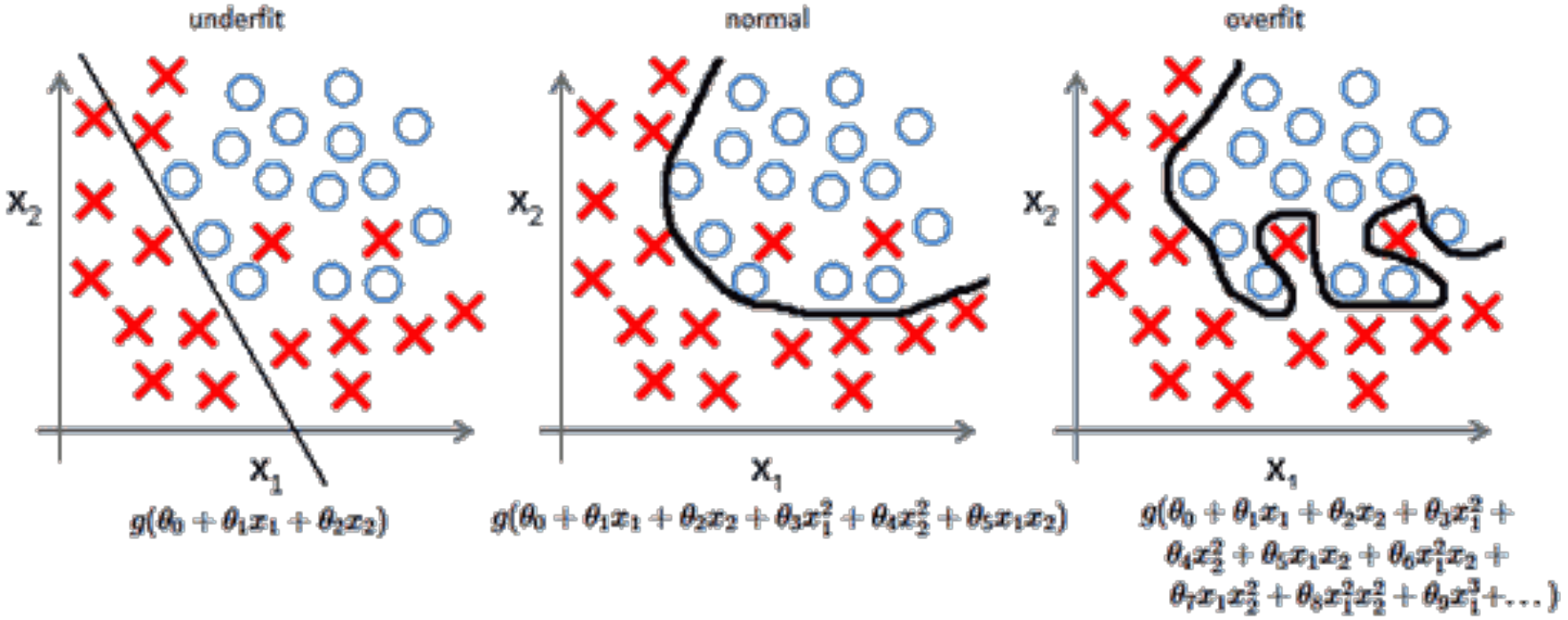
Underfitting
a model does not fit the data well enough

Overfitting
a model is too closely fit to a limited set of data and lose generalization ability

Overfitting

- If we have too many features, the hypothesis may fit the training set very well, but fail to generalize to new examples (high variance)
- More broadly, variance also represents how similar the results from a model will be, if it were fed different data from the same process
- The bias error is from erroneous assumptions in the learning algorithm
- The variance error is from sensitivity to small fluctuations in the training set

Example in Logistic Regression



Underfitting
High bias

Just right

Overfitting
High variance

Address Overfitting

- Feature Reduction
  - Manual selecting which features to keep (by domain knowledge)
  - Okay esp. when some features are really useless

- Regularization
  - Keep all features, but reduce their influence by giving smaller values to the parameter θ_i
  - Okay when many features, each of which contributes a bit to predicting y

Regularized Linear Regression

- Linear Regression

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$J(\theta_0, \theta_1, \dots) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

- Regularized Linear Regression

$$J(\theta_0, \theta_1, \dots) = \frac{1}{2m} \left[\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

- The value of the cost function is NOT equivalent to prediction error. Our goal is to make prediction errors on test data small

Understanding

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

- Penalized term: penalize large parameter values $\theta_j, 1 \leq j \leq n$
- Parameter λ: control the tradeoff
  - Too small: degenerate to linear regression (overfitting)
  - Too large: penalize all features except θ_0, resulting in $h_\theta(x) = \theta_0$ (a horizontal line! underfitting)

Regularized Gradient Descent

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\left(\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} + \lambda\theta_j\right)$$

Repeat until convergence {

$$\theta_j = \theta_j\left(1 - \lambda\frac{\alpha}{m}\right) - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

}

Types of Regularization Regression

- $\|\theta\|_2$: Ridge Regression

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

- $\|\theta\|_1$: LASSO Regression

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\sum_{j=1}^{n}|\theta_j|\right]$$

LASSO regression results in sparse solutions – vector with more zero coordinates.
Good for high-dimensional problems – don't have to store all coordinates!

Supplement Material: Visual for Ridge Vs. LASSO Regression https://www.youtube.com/watch?v=Xm2C_gTAl8c

Regularized Logistic Regression

- Logistic Regression

$$h_\theta(x) = g(\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n)$$

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

- Regularized Logistic Regression

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Regularized Gradient Descent

$$h_\theta(x) = g(\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n) = \frac{1}{1 + e^{-(\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n)}}$$

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\left(\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} + \lambda\theta_j\right)$$

Repeat until convergence {

$$\theta_j = \theta_j\left(1 - \lambda\frac{\alpha}{m}\right) - \frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

}