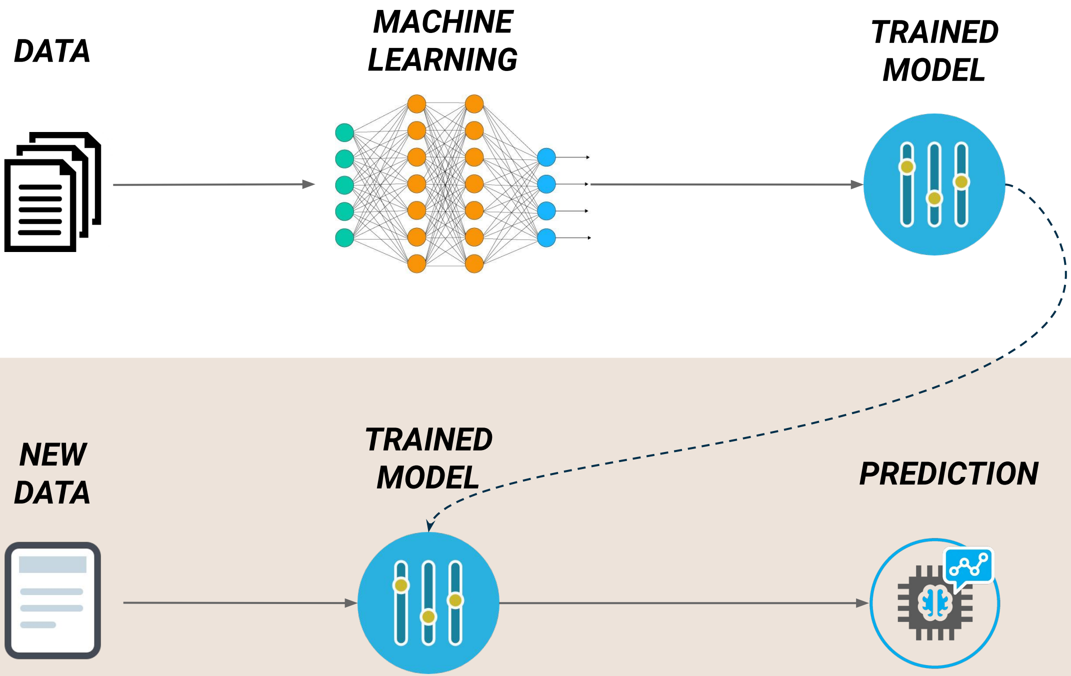
ECE ING4
MACHINE LEARNING

Jeremy Cohen

Logistic Regression

Week 2 Review

Machine Learning Process



DATA

MACHINE LEARNING

TRAINED MODEL

TRAINING

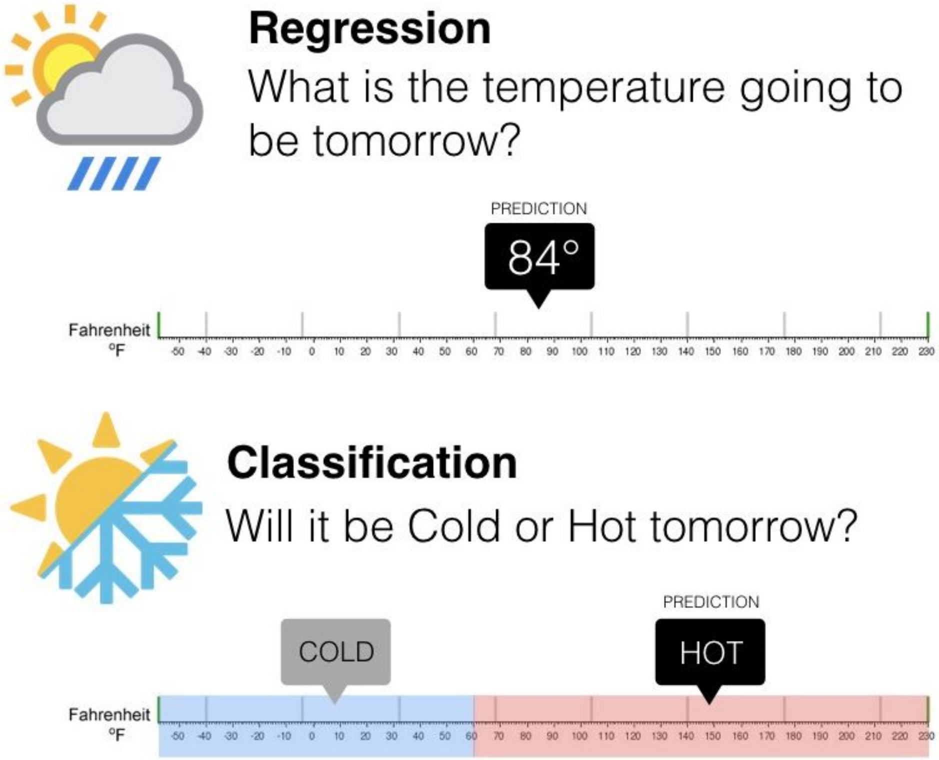
NEW DATA

TRAINED MODEL

PREDICTION

PREDICTION

Classification vs Regression



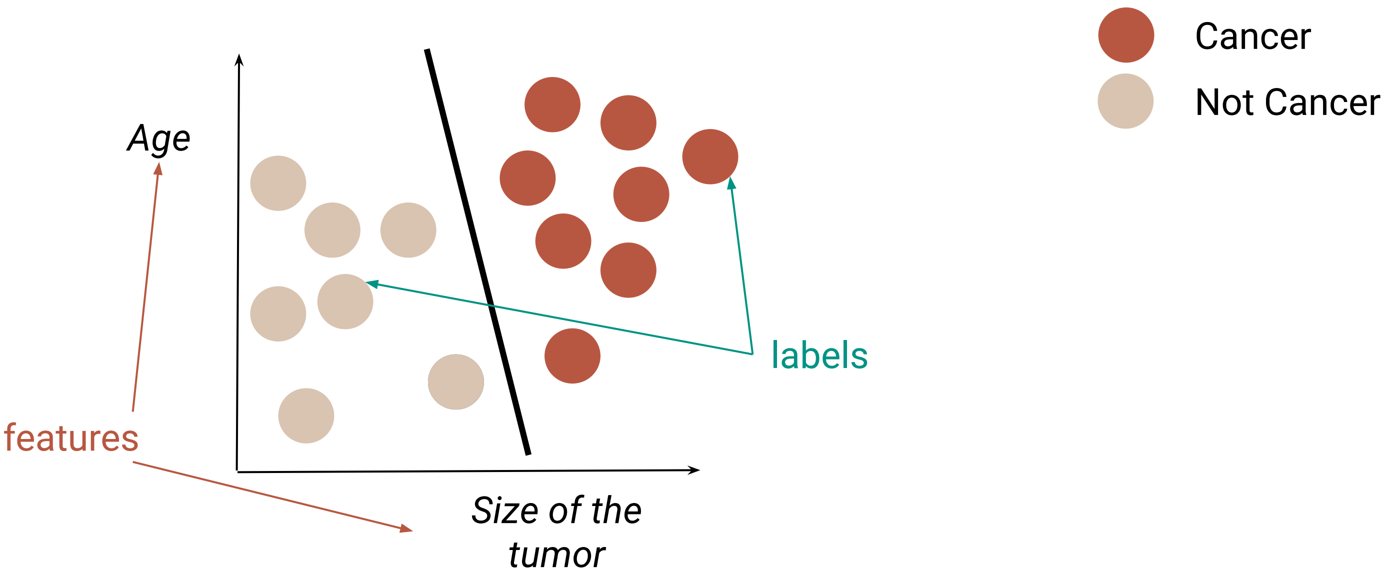
Regression

What is the temperature going to be tomorrow?

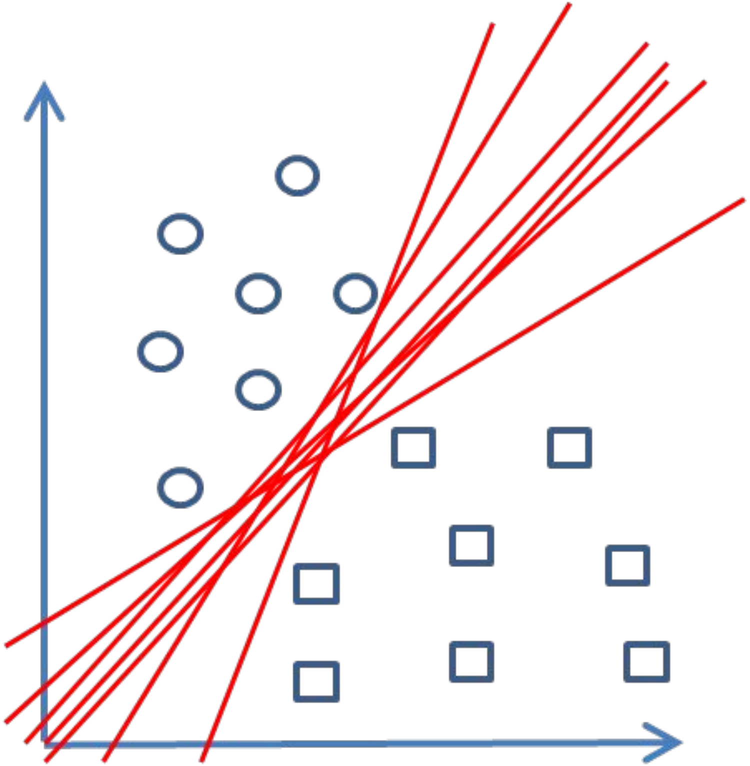
Classification

Will it be Cold or Hot tomorrow?

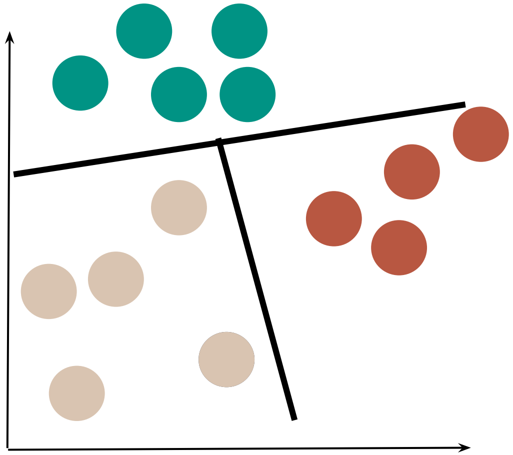
Classification



Classification

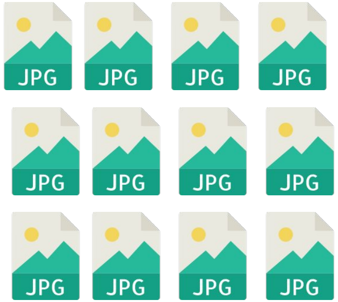


Multi-Class Classification



3 Datasets

Training Set



~70% of the dataset

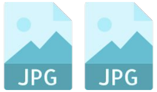
Used to **train the model**

Validation Set

~20% of the dataset

Used to **test** the model and **generalize better to new data**

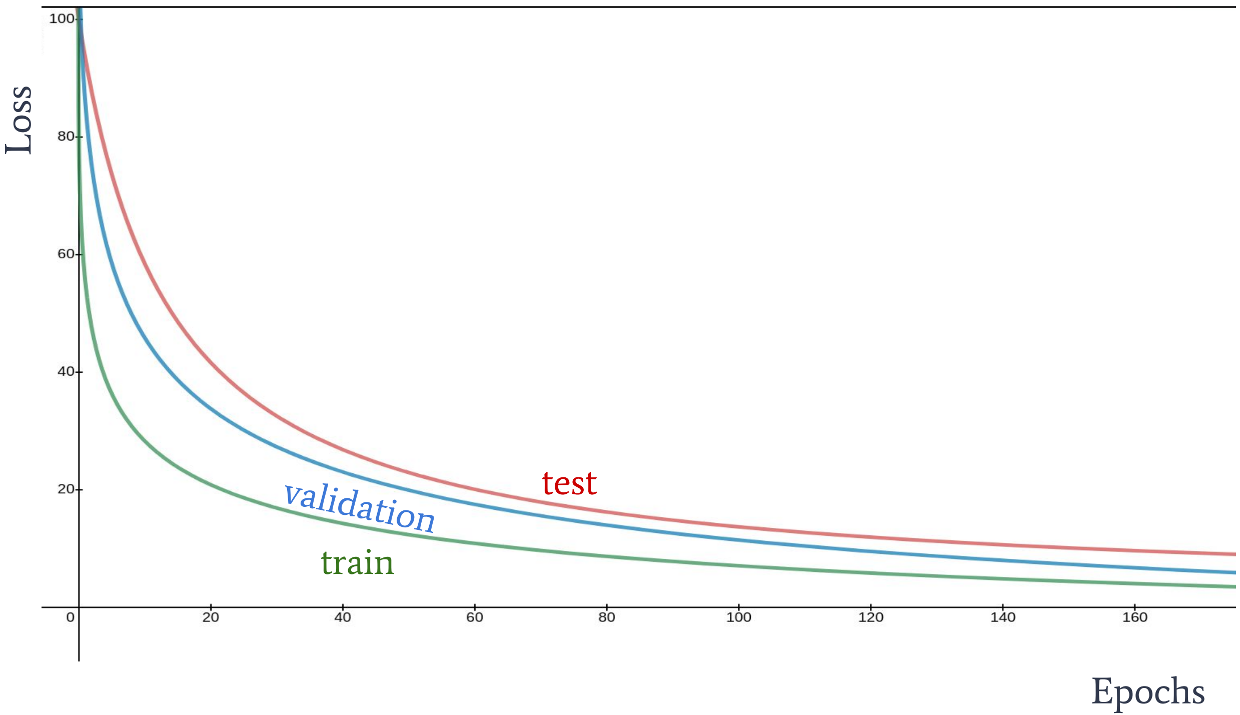
Test Set



~10% of the dataset

Used **only once** when both accuracies are good and ready for real-world

3 Datasets


Loss
100
80
60
40
20
0
20
40
60
80
100
120
140
160
test
validation
train
Epochs

Linear Regression formula

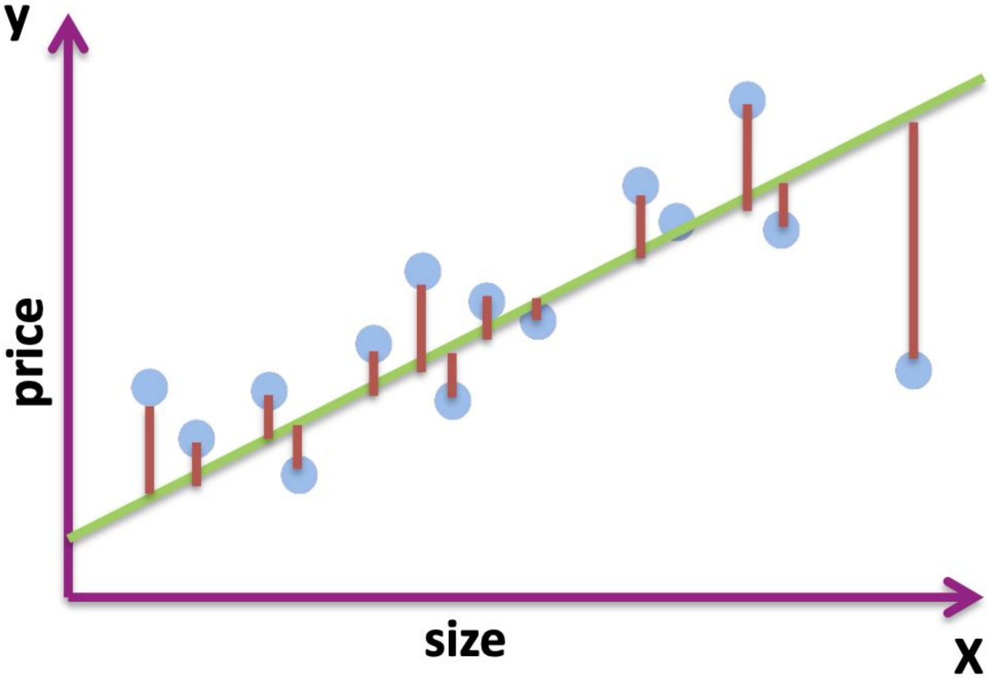
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

For convenience of notation, define $x_0 = 1$ $(x_0^{(i)} = 1)$

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

$$h_\theta(x) = \sum_{i=1}^{n} \theta_i x_i = \theta^T x$$

Mean Squared Error

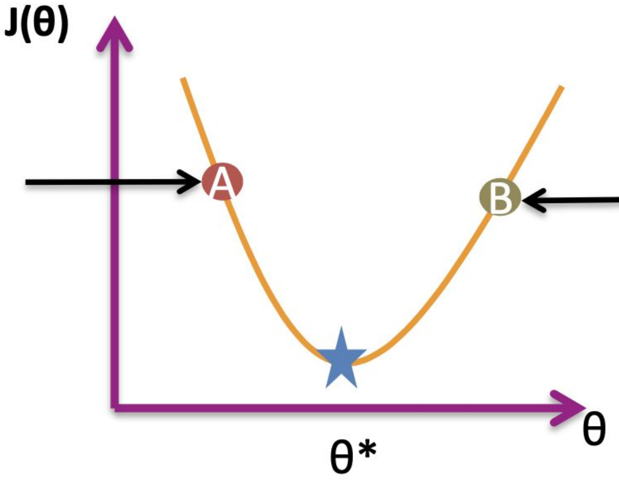


$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Gradient Descent Recap

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

In this case, the derivative (gradient) ∂J(θ) / ∂θ < 0 .
$\theta^A - \alpha*\partial J(\theta) / \partial\theta > \theta^A$
θ^A is moving to the right
θ is increasing



In this case, the derivative (gradient) ∂J(θ) / ∂θ > 0 .
$\theta^B - \alpha*\partial J(\theta) / \partial\theta < \theta^B$
θ^B is moving to the left
θ is decreasing

Gradient Descent Recap

Repeat $\{$

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

(simultaneously update θ_j for $j = 0, \ldots, n$)

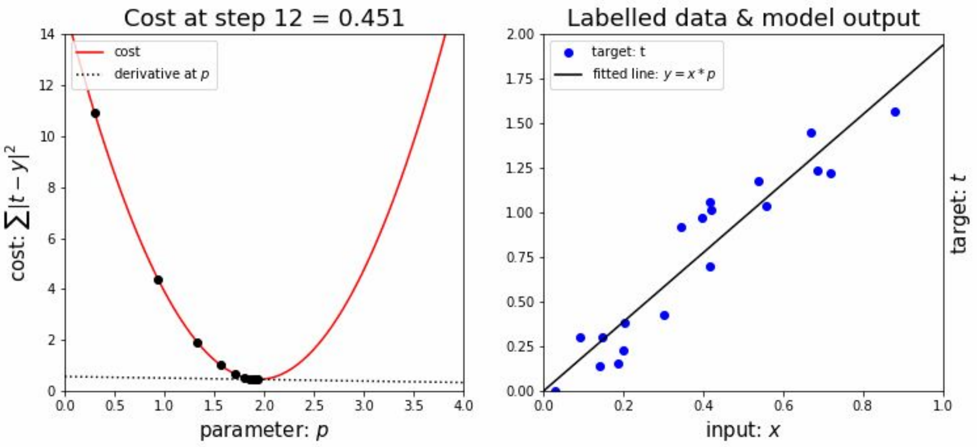
$\}$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

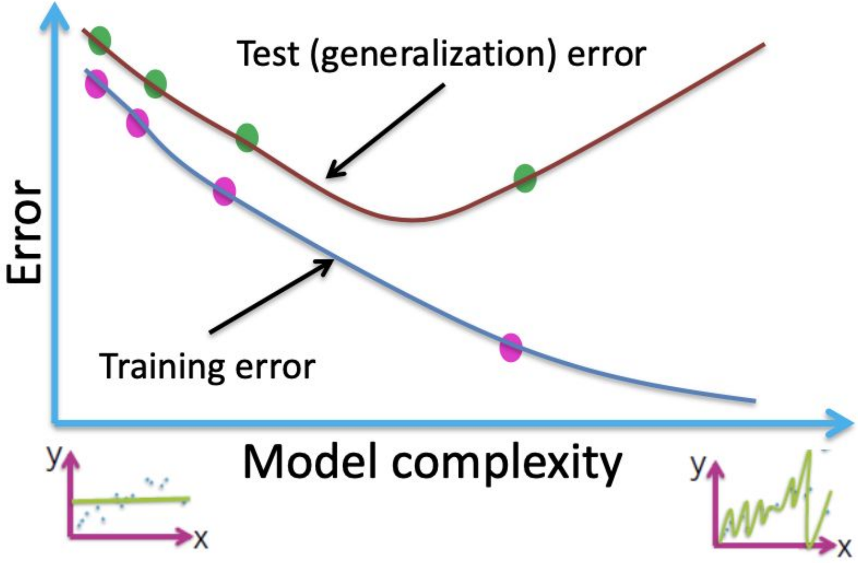
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

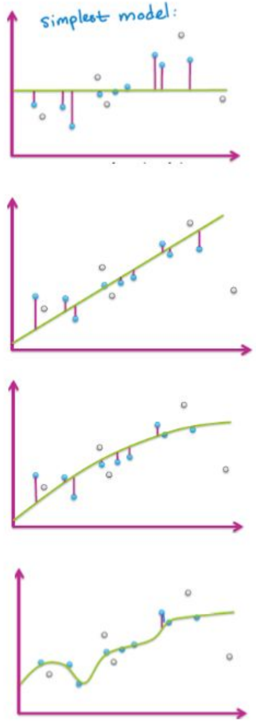
$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

$\ldots$



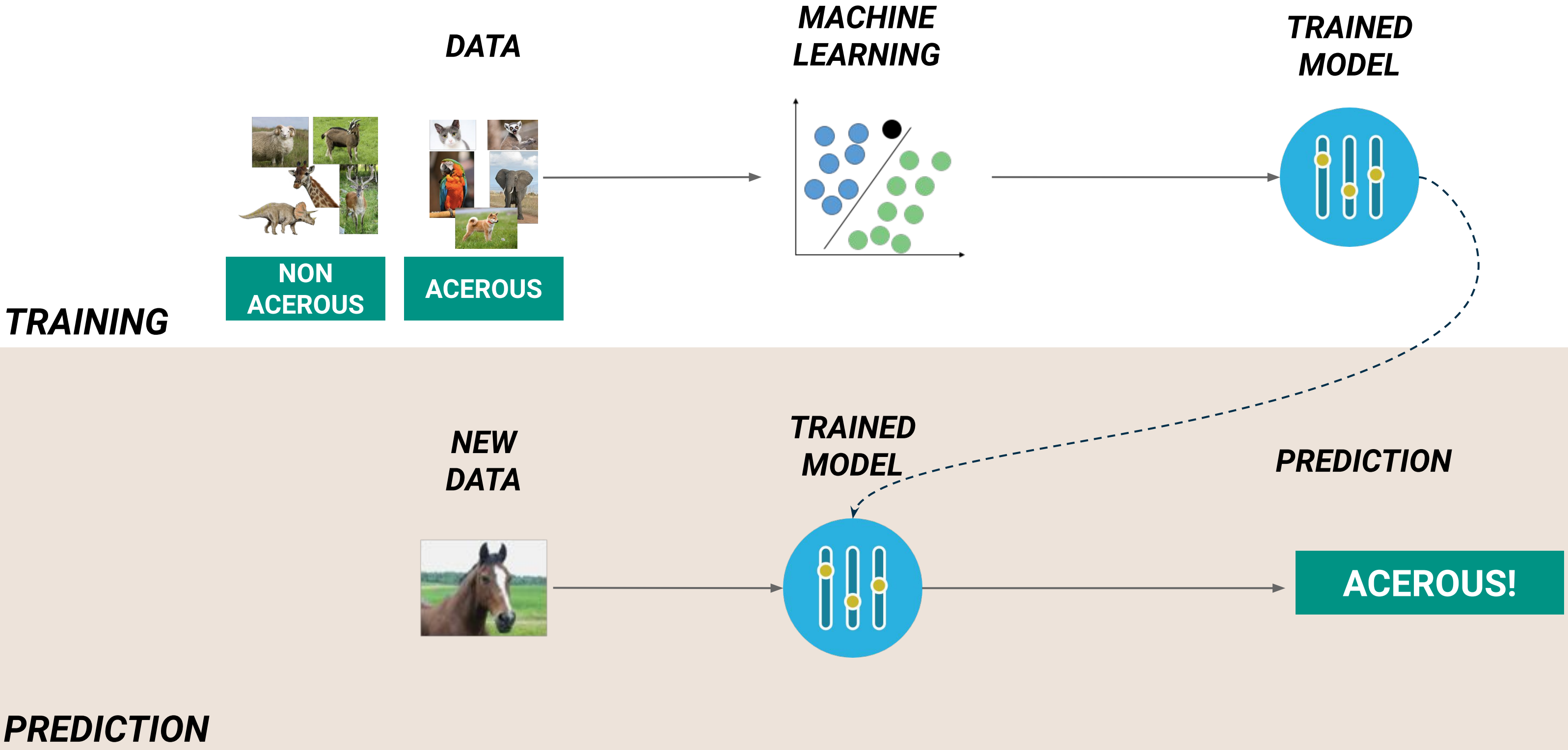
Performance





Classification

Classification Example



TRAINING

DATA

NON ACEROUS

ACEROUS

MACHINE LEARNING

TRAINED MODEL

PREDICTION

NEW DATA

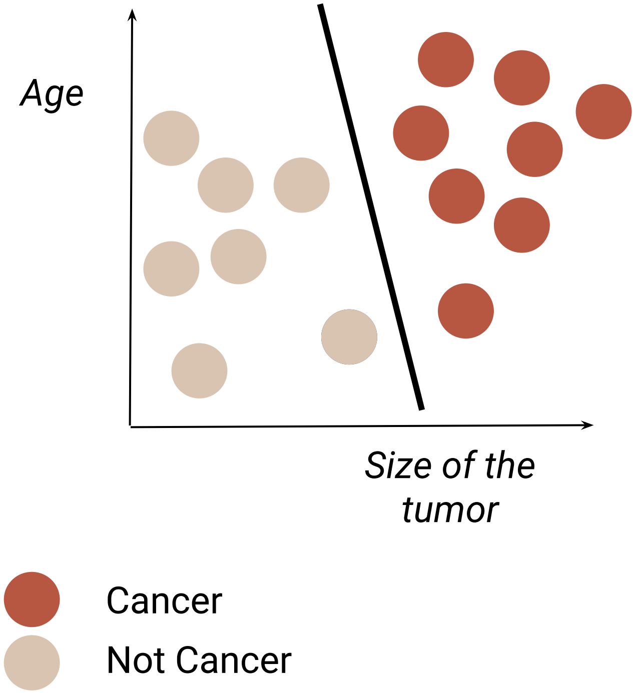
TRAINED MODEL

PREDICTION

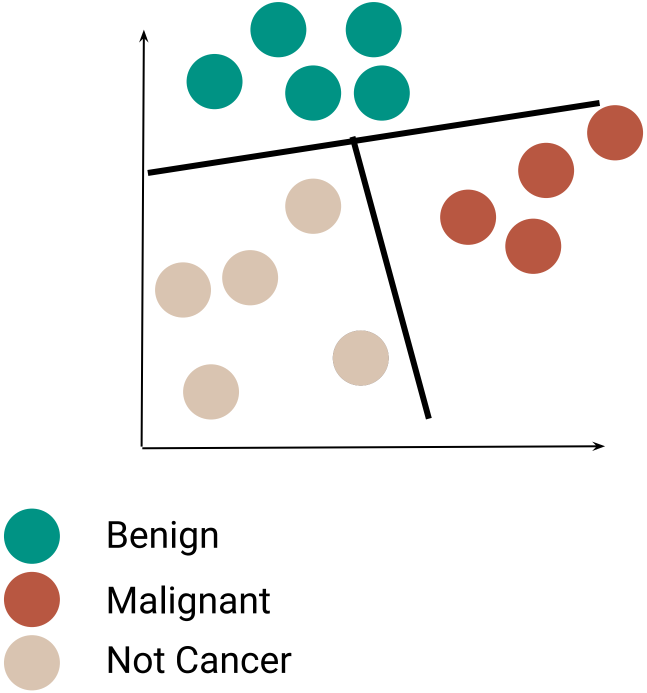
ACEROUS!

Binary vs Multi-Class Classification

Output has 2 categories

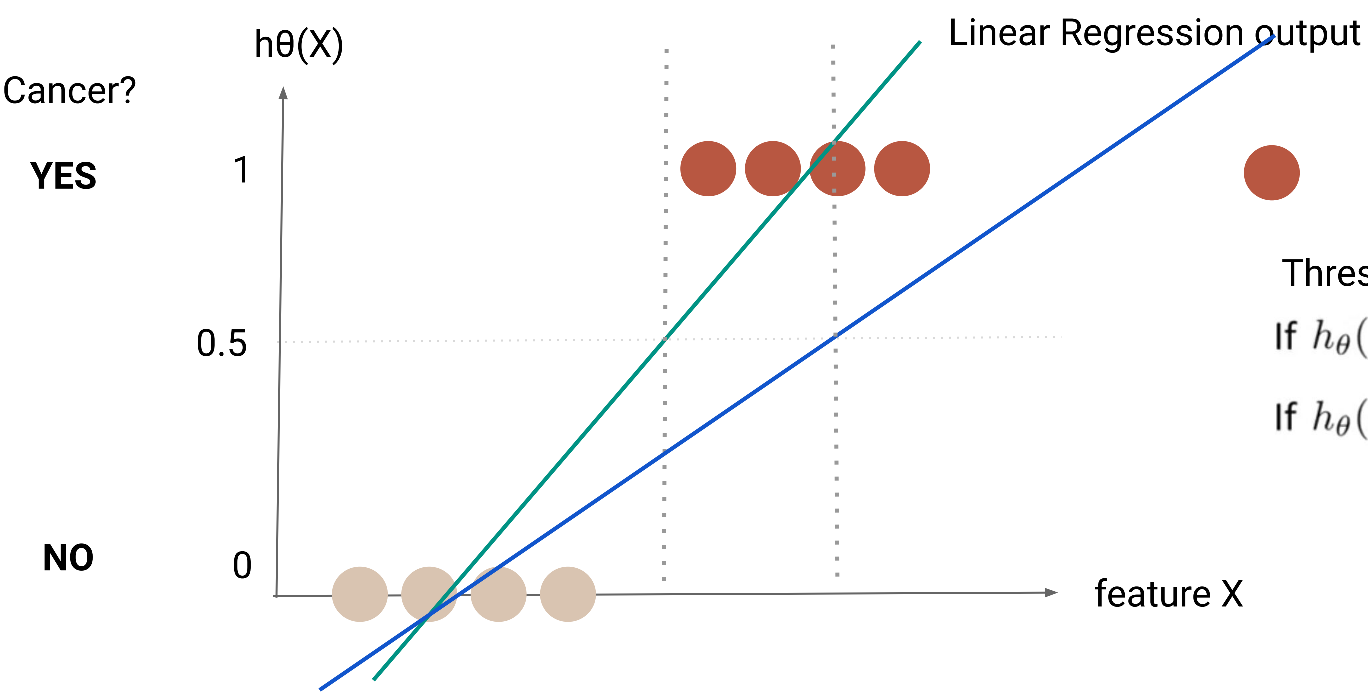


Output has more than categories



Can Linear Regression help?

$$h_\theta(x) = \sum_{i=1}^{n} \theta_i x_i = \theta^T x$$



Threshold the classifier at 0.5

If $h_\theta(x) \geq 0.5$, predict "y = 1"

If $h_\theta(x) < 0.5$, predict "y = 0"

Linear Regression vs Classification

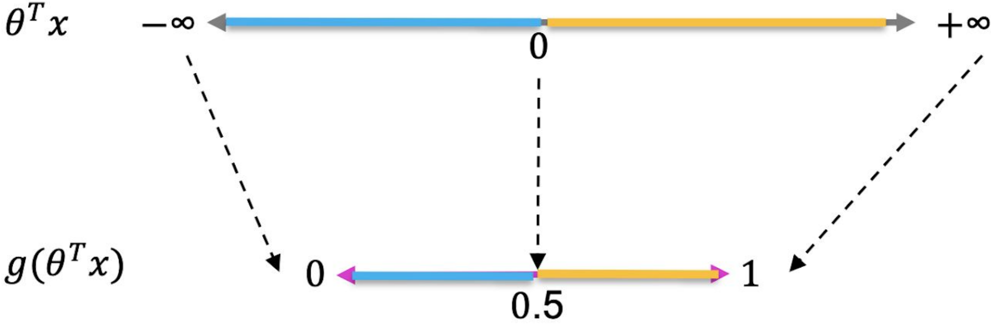
Linear Regression output : $- \infty < h_{\theta}(x) < + \infty$

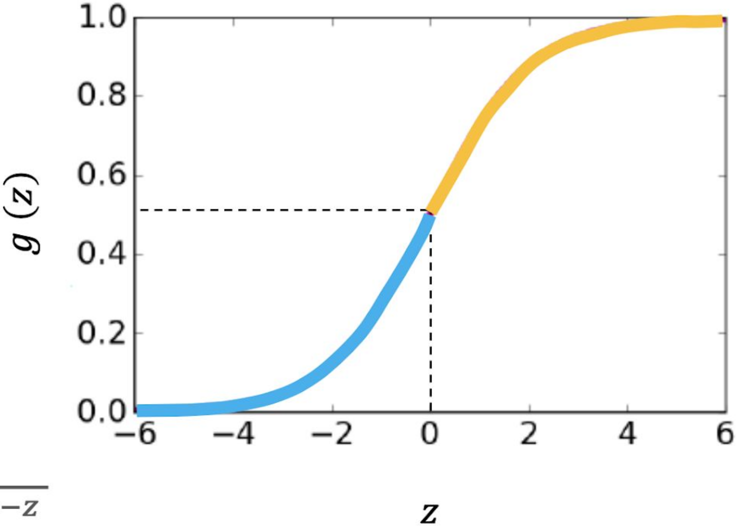
Classification output: $y = 0$ or 1

Logistic Regression output : $0 \leqq h_{\theta}(x) \leqq 1$

Linear Regression can’t help

We want 0 ≦ hθ(x) ≦ 1





- Use the Sigmoid / Logistic Function : $g(z) = \frac{1}{1 + e^{-z}}$

$$h_\theta(x) = g(\theta^T x)$$

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

Logistic Regression

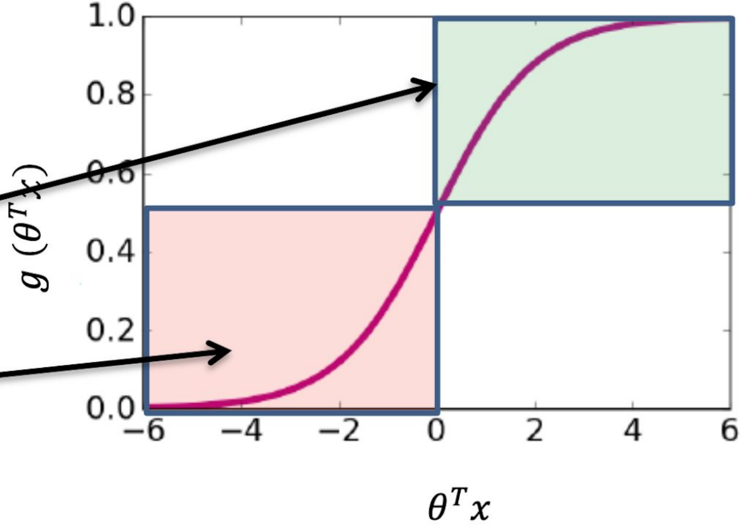
$h_\theta(x)$ = **estimated probability that y = 1** given the input x parameterized by θ

$$h_\theta(x) = \mathrm{P}(\mathrm{y} = 1 |\ \mathrm{x}; \theta) = 1 - P(\mathrm{y} = 0 |\ \mathrm{x}; \theta)$$

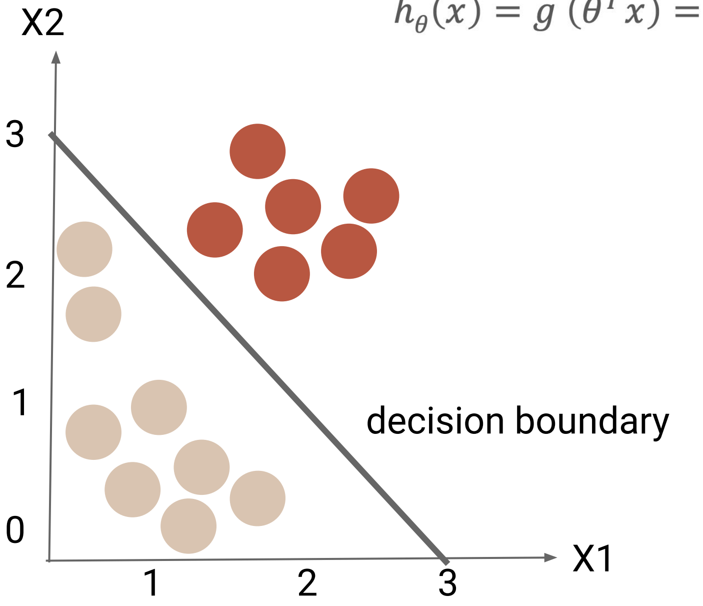
$$\mathrm{P}(\mathrm{y} = 1 |\ \mathrm{x}; \theta) + P(\mathrm{y} = 0 |\ \mathrm{x}; \theta) = 1$$

Hypothesis:

- □ $y = 1\ when\ g(\theta^T x) \geq 0{,}5$ ➔ $\theta^T x \geq 0$
- □ $y = 0\ when\ g(\theta^T x) < 0{,}5$ ➔ $\theta^T x < 0$



Logistic Regression



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

If $h_\theta(x) \geq 0.5$, predict "y = 1" $\theta^T x \geq 0$

If $h_\theta(x) < 0.5$, predict "y = 0" $\theta^T x < 0$

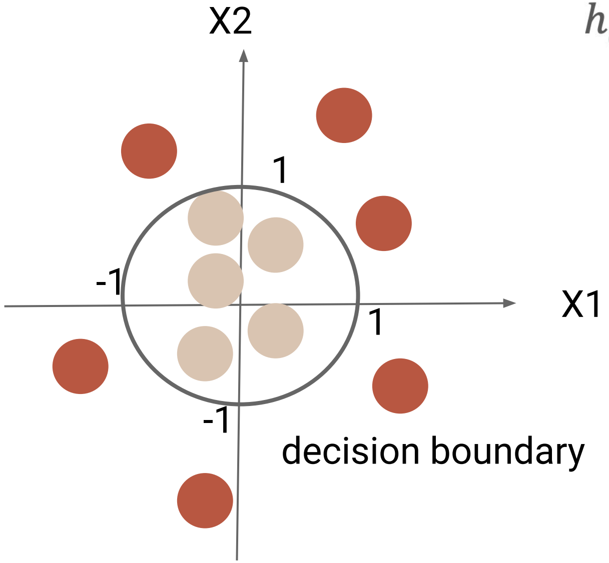
$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$$\theta = \begin{matrix} -3 \\ 1 \\ 1 \end{matrix}$$

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

x1 + x2 ≧ 3

Non-Linearities



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

Example: $h_\theta(x) = g\left(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 {x_1}^2 + \theta_4 {x_2}^2\right)$

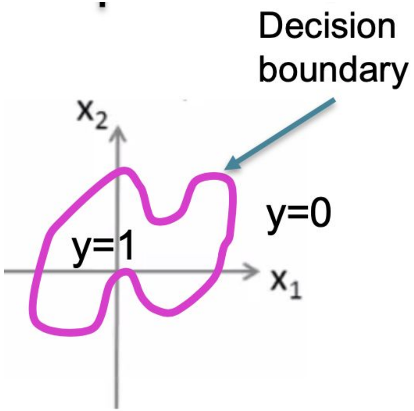
- $\theta = [-1, 0, 0, 1, 1]$
- Predict 1 if $-1 + {x_1}^2 + {x_2}^2 \geq 0 \Rightarrow {x_1}^2 + {x_2}^2 \geq 1$
- Predict 0 if $-1 + {x_1}^2 + {x_2}^2 < 0 \Rightarrow {x_1}^2 + {x_2}^2 < 1$

We can work with non-linearly separable data

Non-Linearities

As with polynomial regression, we can have more complex decision boundaries by adding higher polynomial terms

$$h_\theta(x) = g\,(\theta_0 + \theta_1\mathrm{x}_1 + \theta_2\mathrm{x}_2 + \theta_3\mathrm{x}_1{}^2 + \theta_4\mathrm{x}_2{}^2 + \theta_5\mathrm{x}_1{}^3 + \theta_6\mathrm{x}_2{}^3)$$

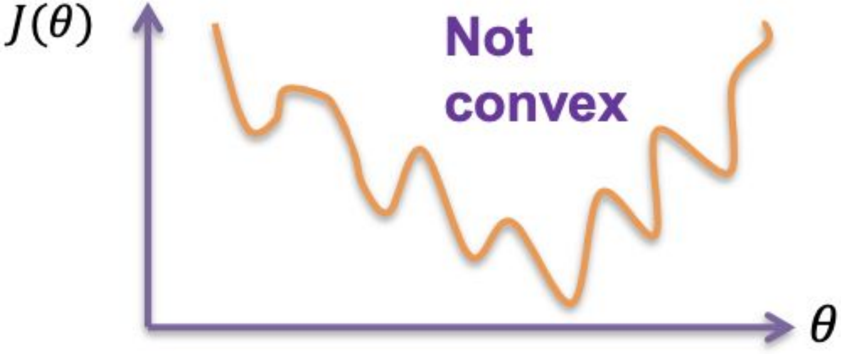


How to choose θ ?

Cost Function

Cost Function for Linear Regression

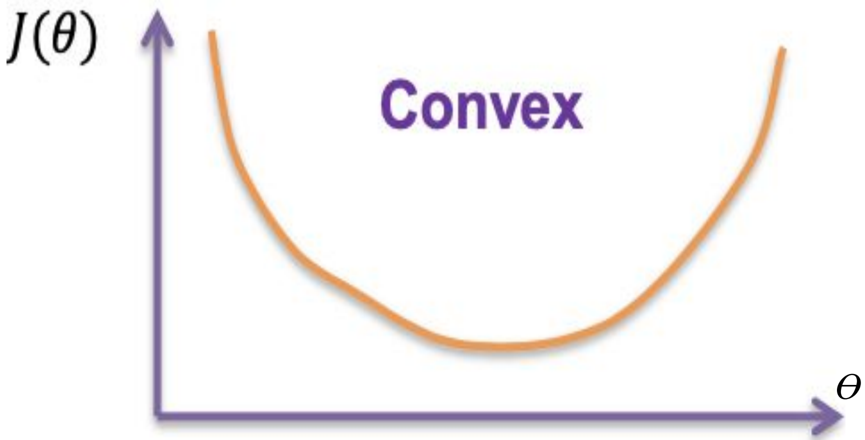
$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$



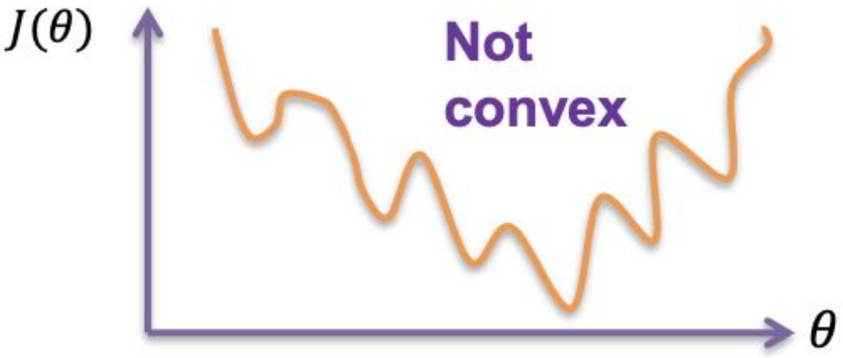
$$\boxed{\frac{1}{1 + e^{-\theta^T x}}}$$

Cost Function for Linear Regression

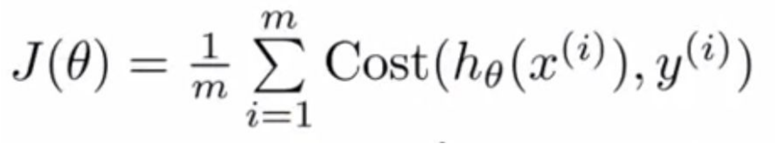
Possible to run Gradient Descent



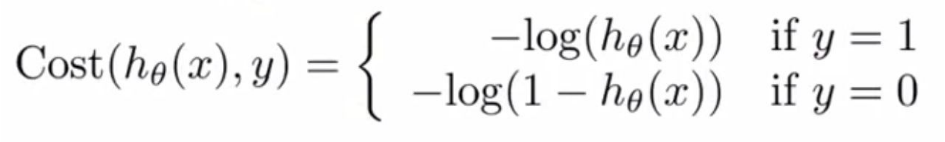
Impossible to run Gradient Descent



Cost Function for Logistic Regression



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



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

Cost Function for Logistic Regression

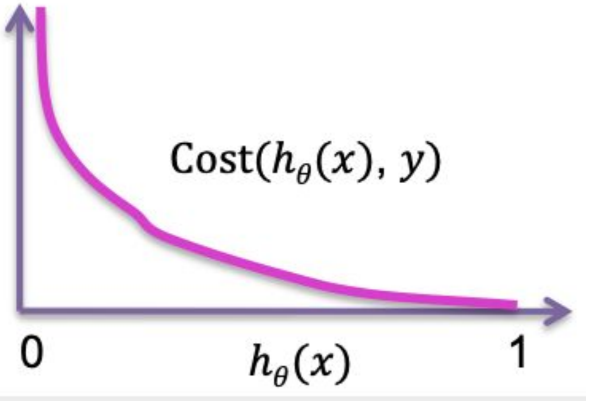
$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

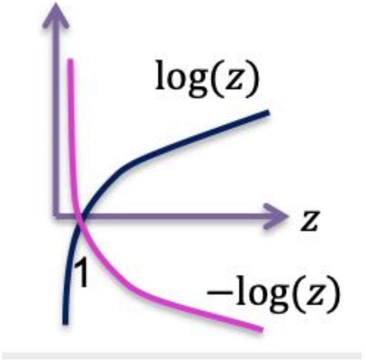
$\mathrm{Cost} = 0 \text{ if } y = 1, h_\theta(x) = 1$

But as $h_\theta(x) \rightarrow 0$

$Cost \rightarrow \infty$

y=1





Cost Function for Logistic Regression

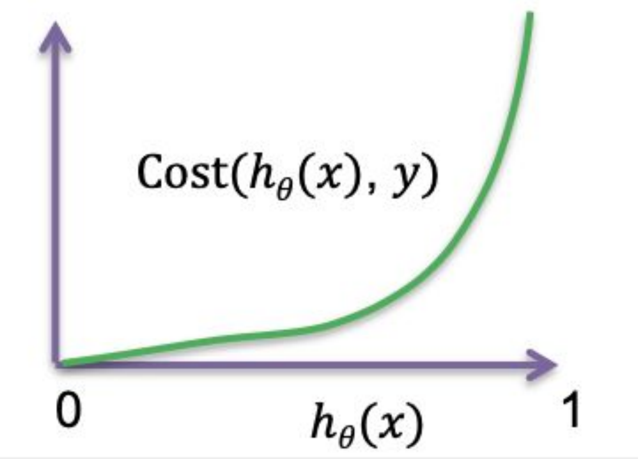
$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

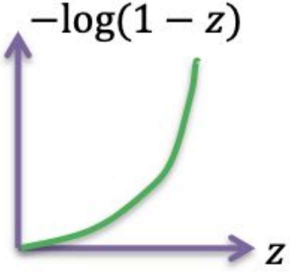
$$\mathrm{Cost} = 0 \text{ if } y = 0, h_\theta(x) = 0$$

$$\text{But as} \quad h_\theta(x) \to 1$$

$$Cost \to \infty$$

y=0



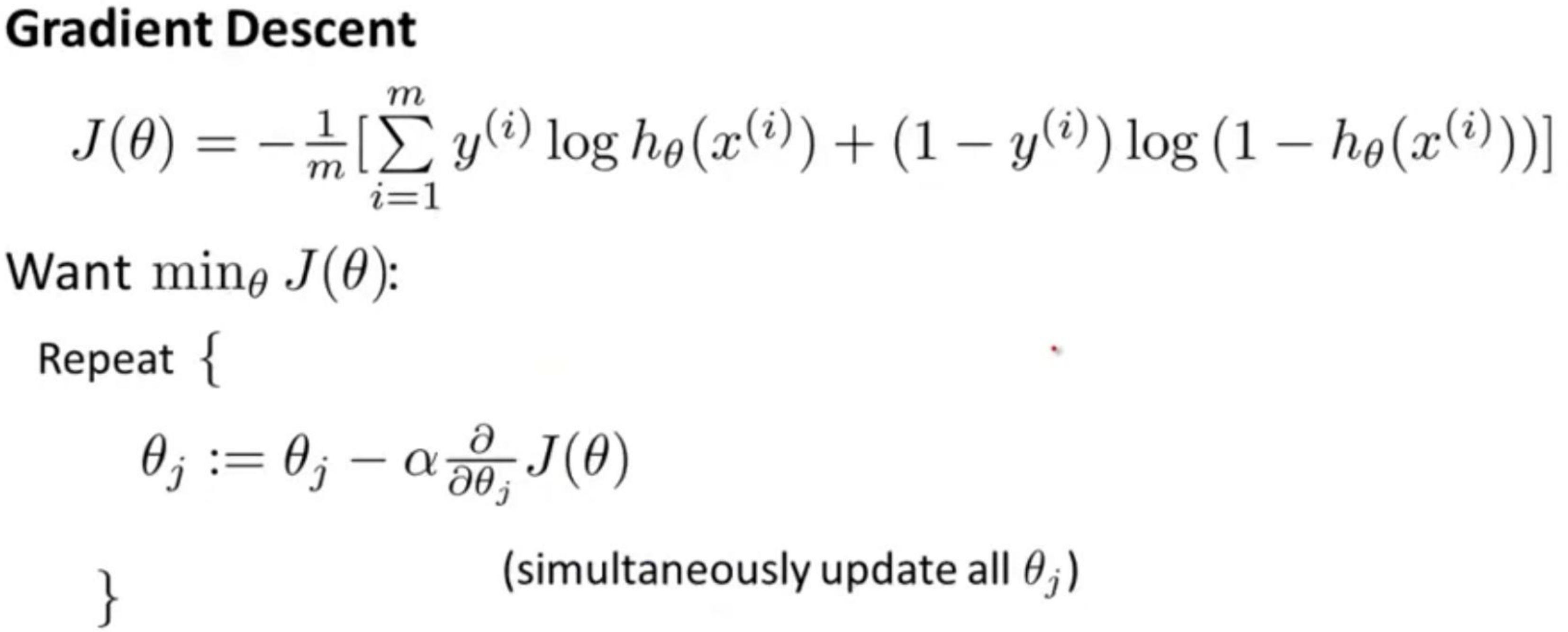


Cost Function for Logistic Regression

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

$$\begin{aligned} J(\theta) &= \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)}) \\ &= -\frac{1}{m} [\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))] \end{aligned}$$

Cost Function for Logistic Regression



Gradient Descent

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

Want $\min_\theta J(\theta)$:

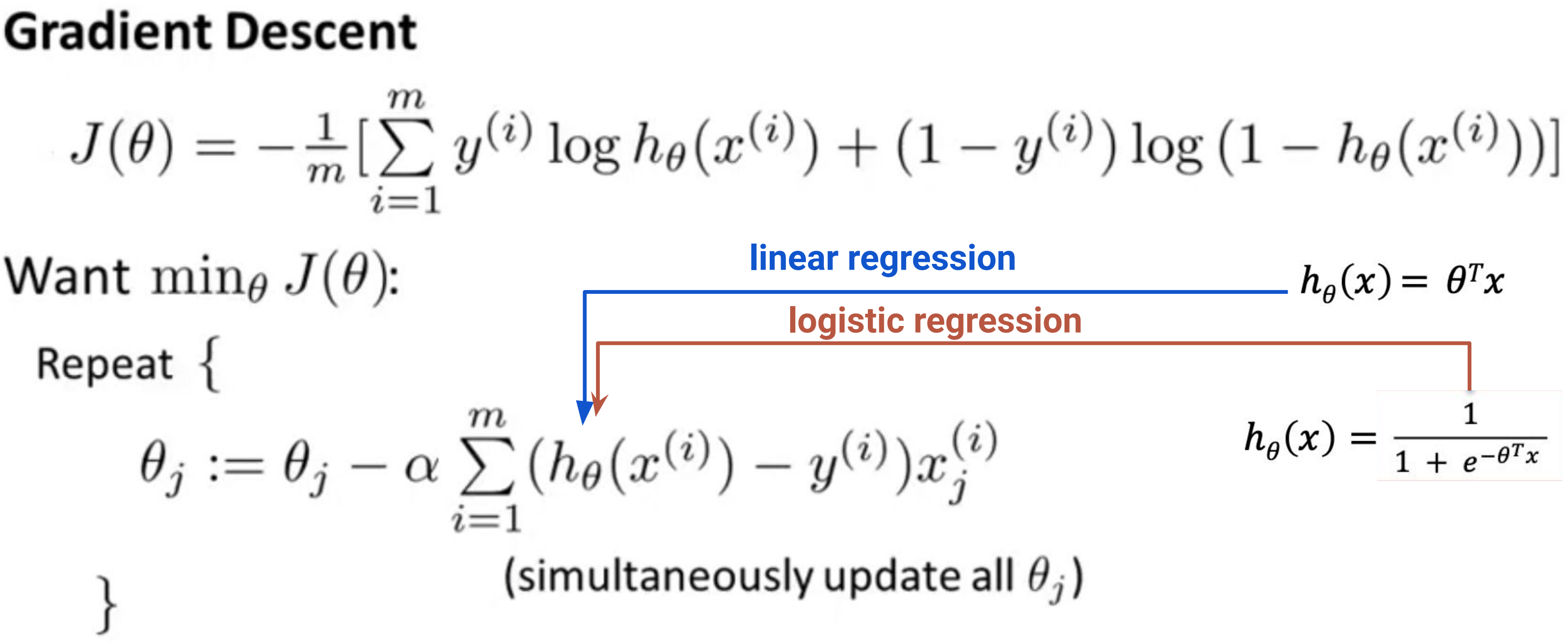
Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

(simultaneously update all θ_j)

$\}$

Cost Function for Logistic Regression



Gradient Descent

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

Want $\min_\theta J(\theta)$:

Repeat $\{$

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

(simultaneously update all θ_j)

$\}$

linear regression

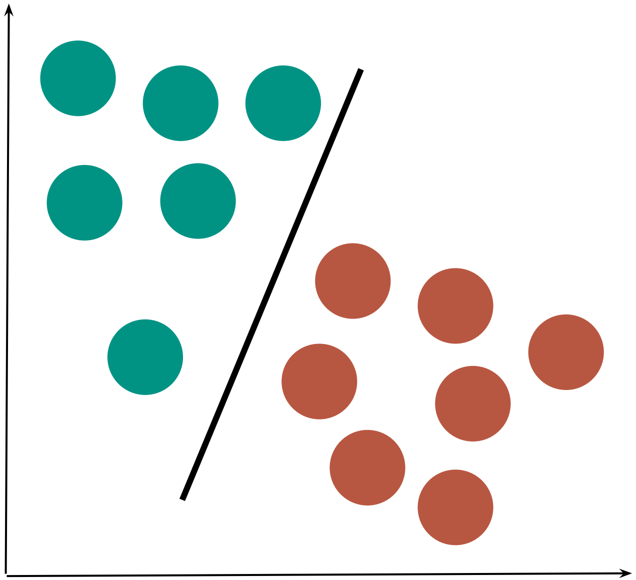
$h_\theta(x) = \theta^T x$

logistic regression

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

Support Vector Machine

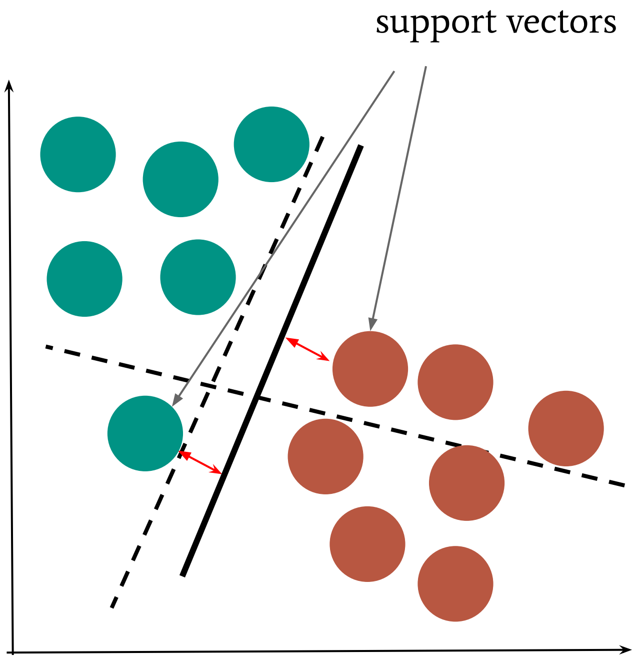
SVM



GOAL

Separate the dataset into classes with as much correctly classified points as possible

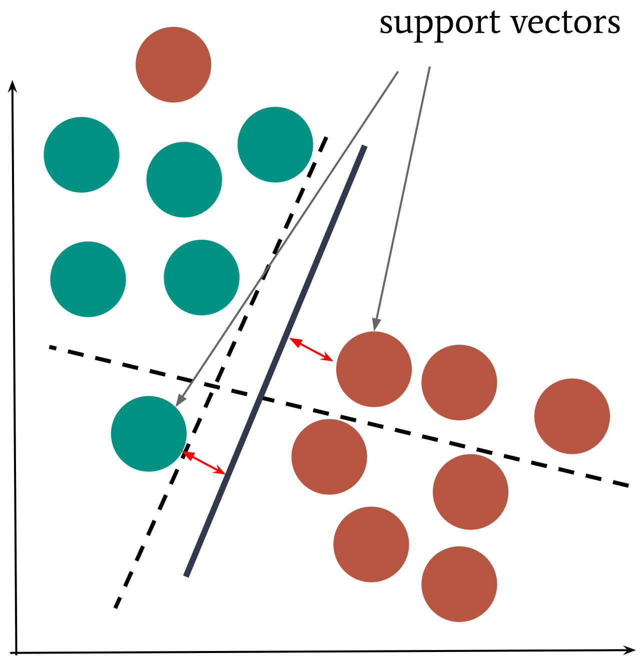
SVM



PROCESS

- Select the line that correctly classifies as many points as possible
- Select the line that maximizes distance with support vectors

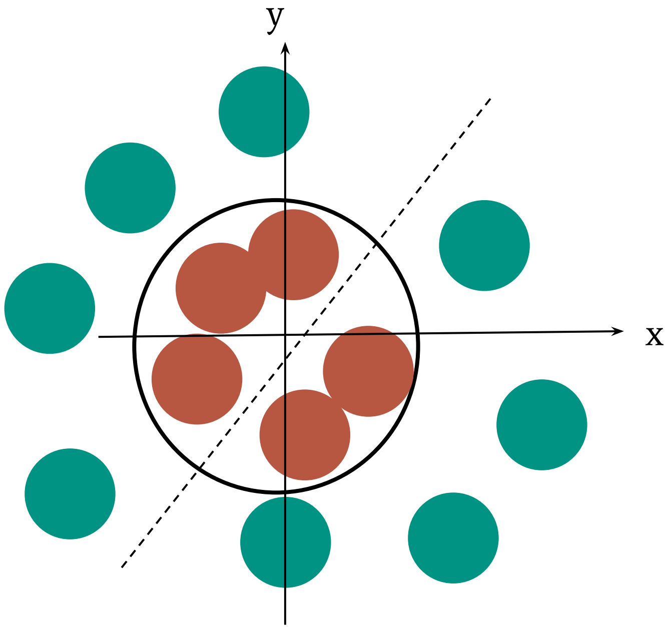
SVM



NOTABLE

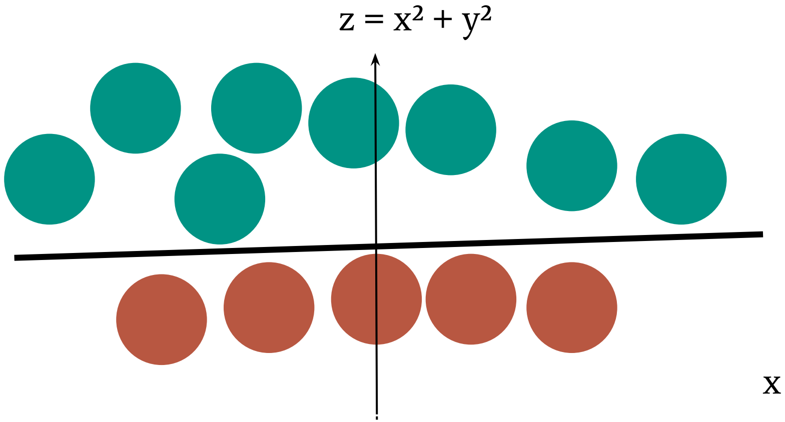
- Robust to outliers/exceptions; line will not be affected

SVM



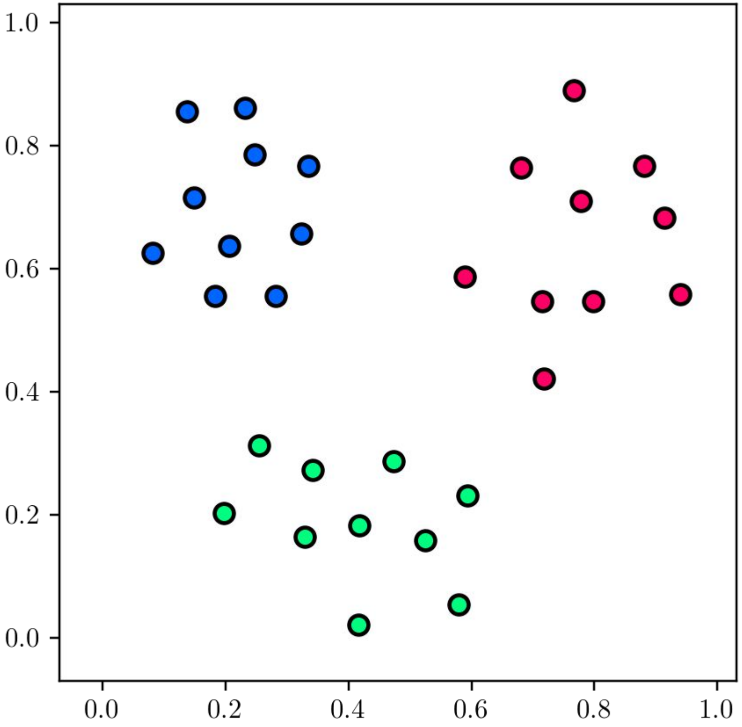
NOTABLE

- If the data is non linearly separable, we can make it

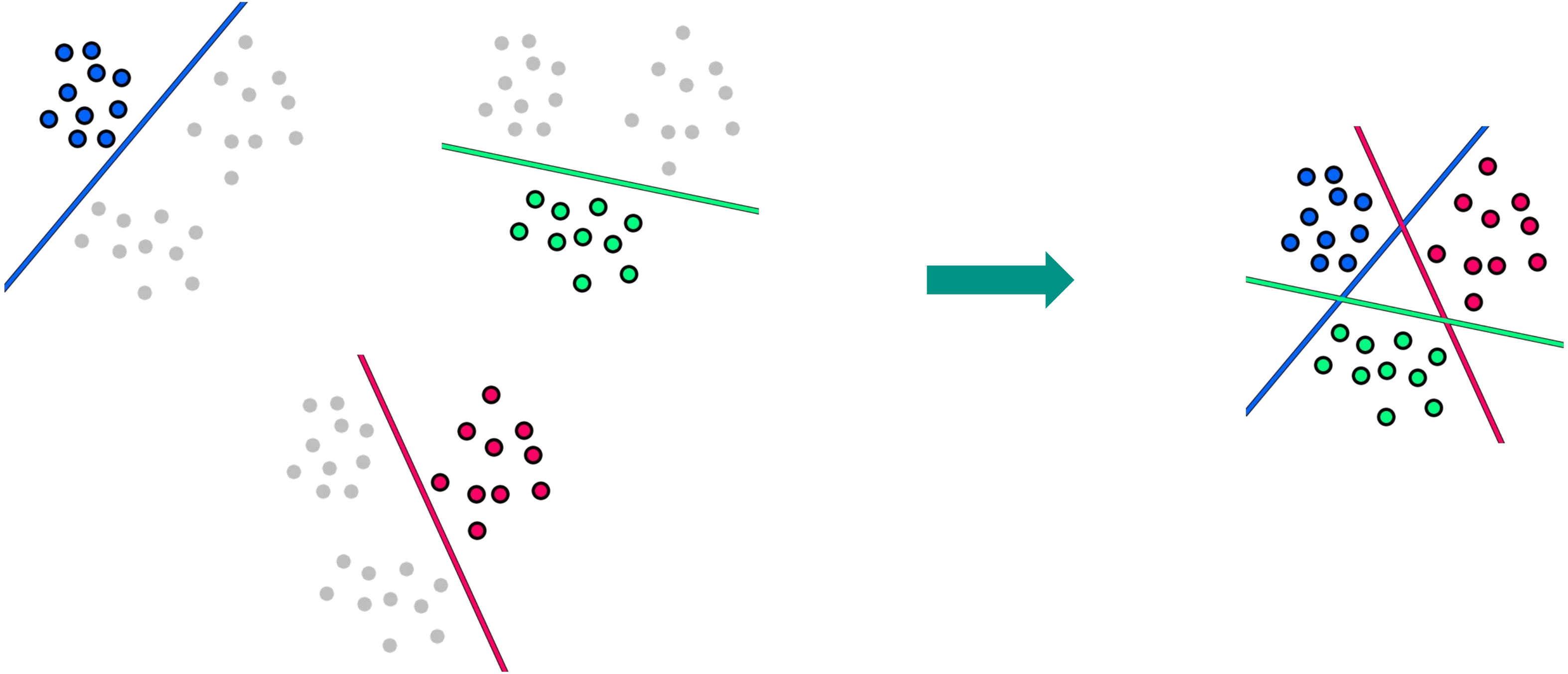


Multiclass Classification

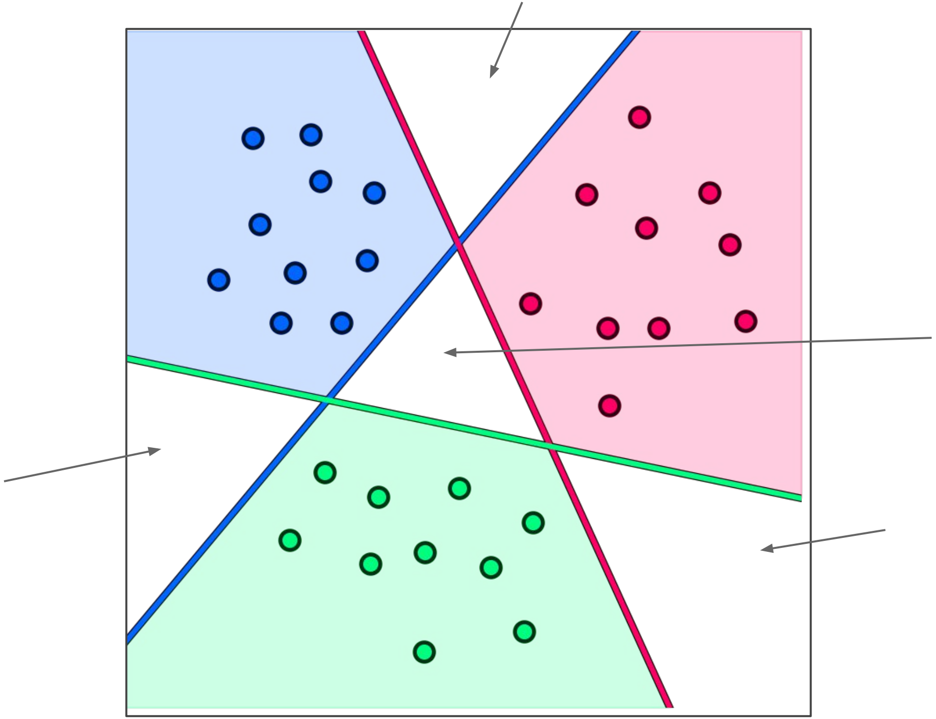
MultiClass Classification: One vs All



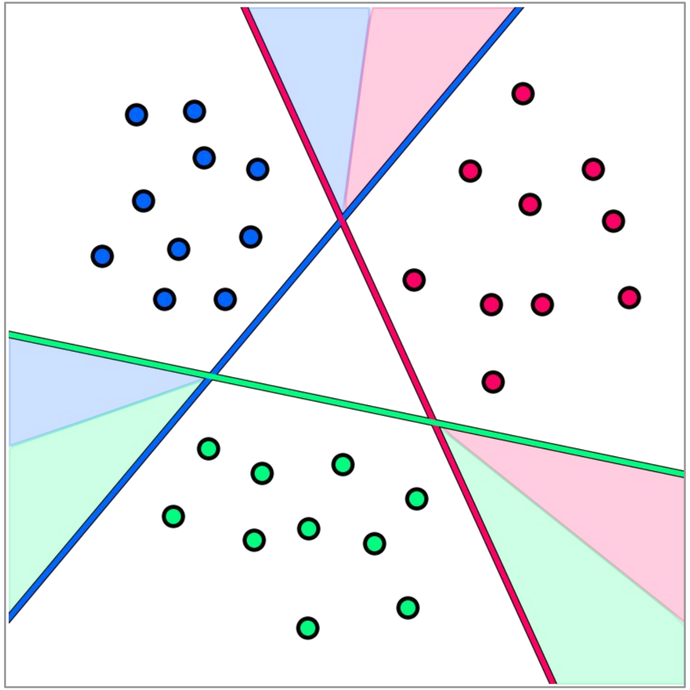
MultiClass Classification: One vs All



MultiClass Classification: One vs All

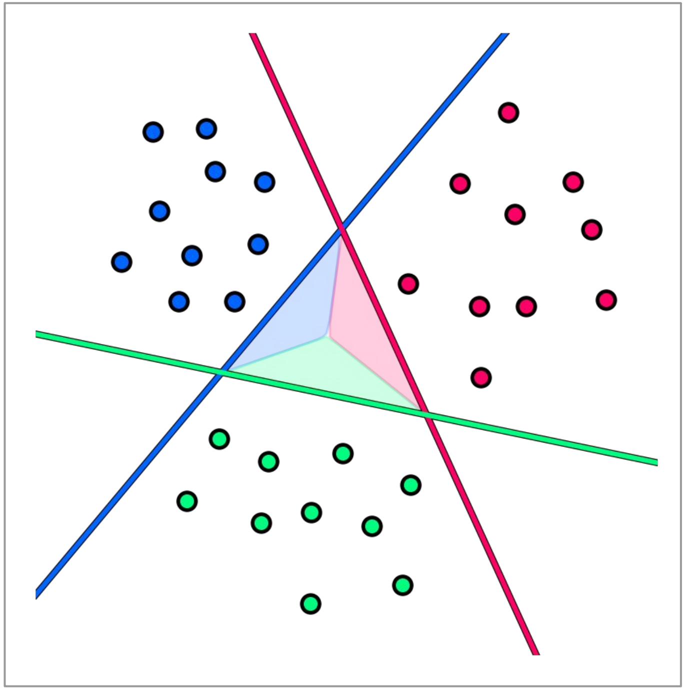


MultiClass Classification: One vs All



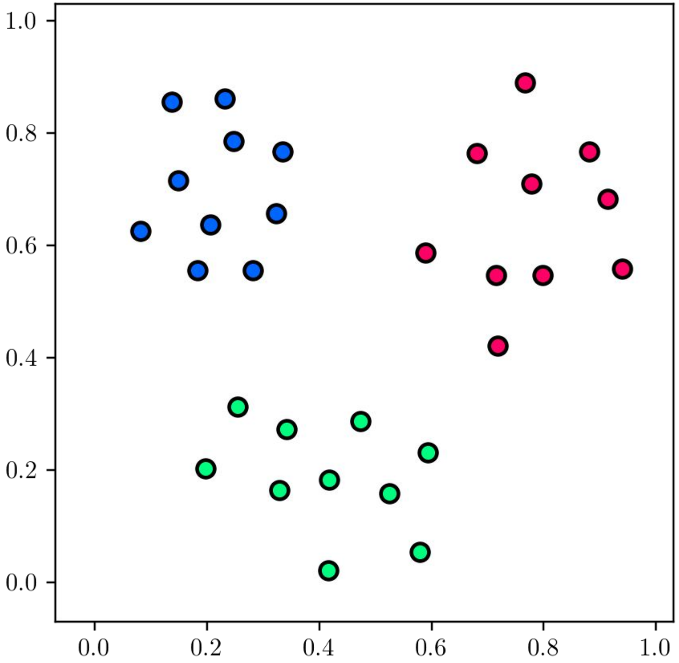
On a new input, to make a prediction, pick the class that maximizes: $max_i \; h_\theta{}^i(x)$

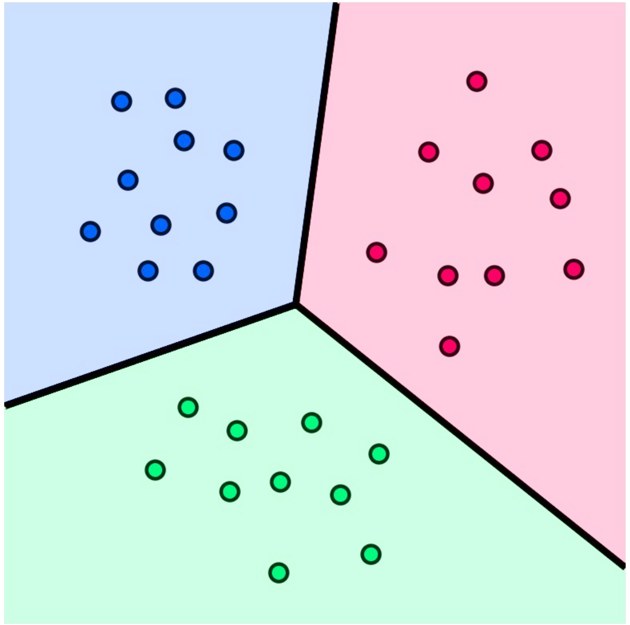
MultiClass Classification: One vs All



On a new input, to make a prediction, pick the class that maximizes: $max_i \; {h_\theta}^i(x)$

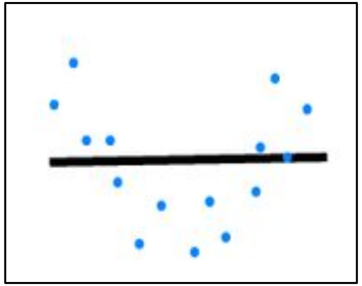
MultiClass Classification: One vs All





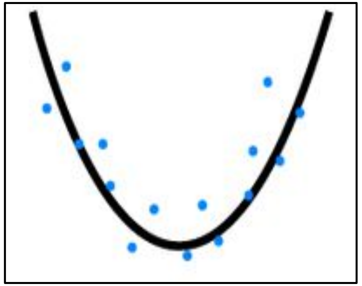
Overfitting

Linear Regression

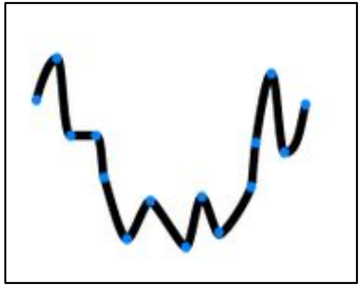


UNDERFITTING

high bias



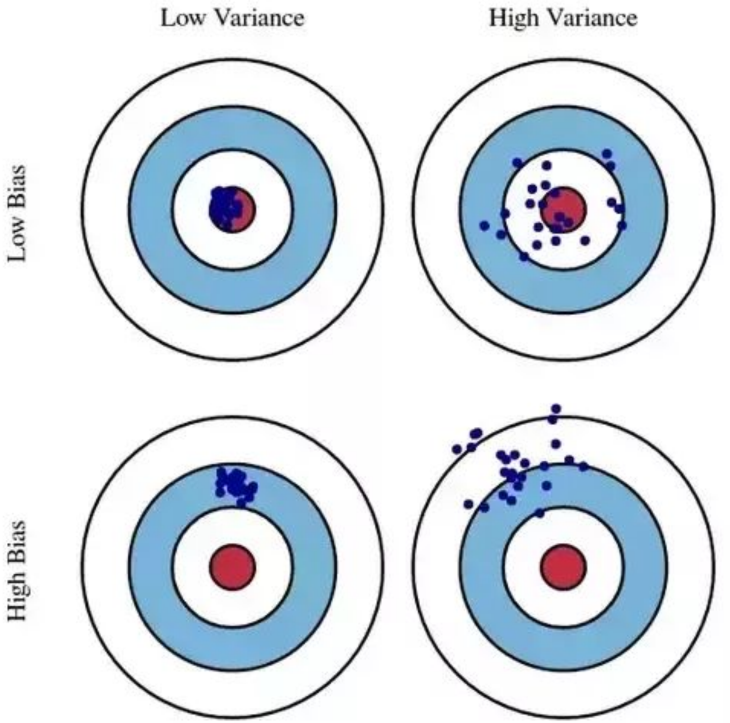
JUST RIGHT



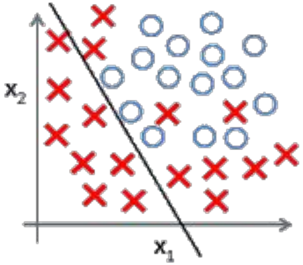
OVERFITTING

high variance

Linear Regression

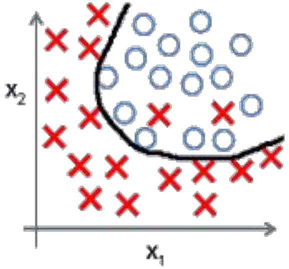


Logistic Regression

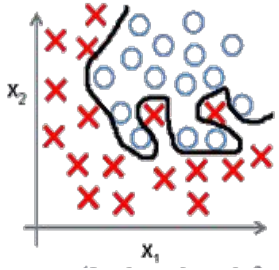


UNDERFITTING

high bias



JUST RIGHT



OVERFITTING

high variance

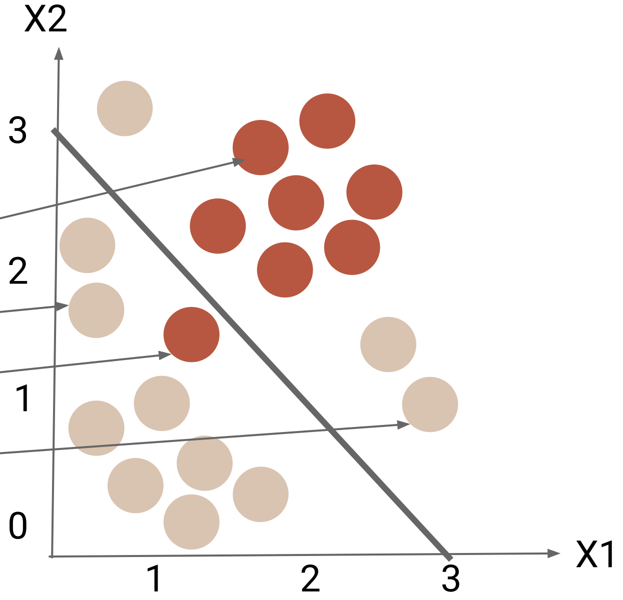
Performance

Accuracy

$$\text{Accuracy} = \frac{number\ of\ data\ points\ classified\ correctly}{all\ data\ points}$$

Confusion Matrix

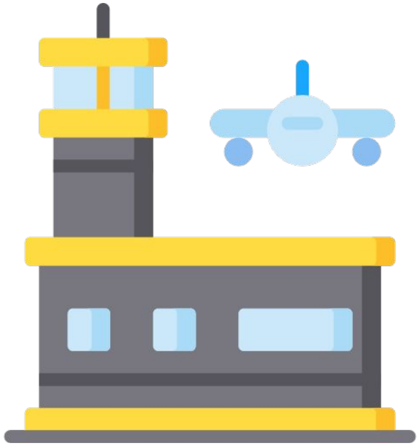
	Positive Examples	Negative Examples
Correct	TRUE POSITIVE	TRUE NEGATIVE
Wrong	FALSE NEGATIVE	FALSE POSITIVE

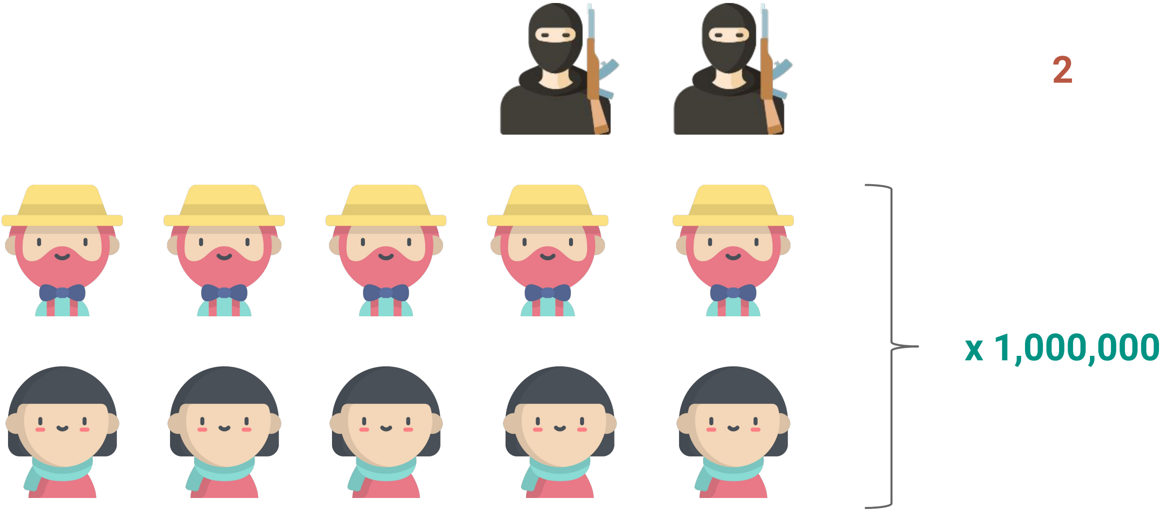


Imbalanced Classification Problem

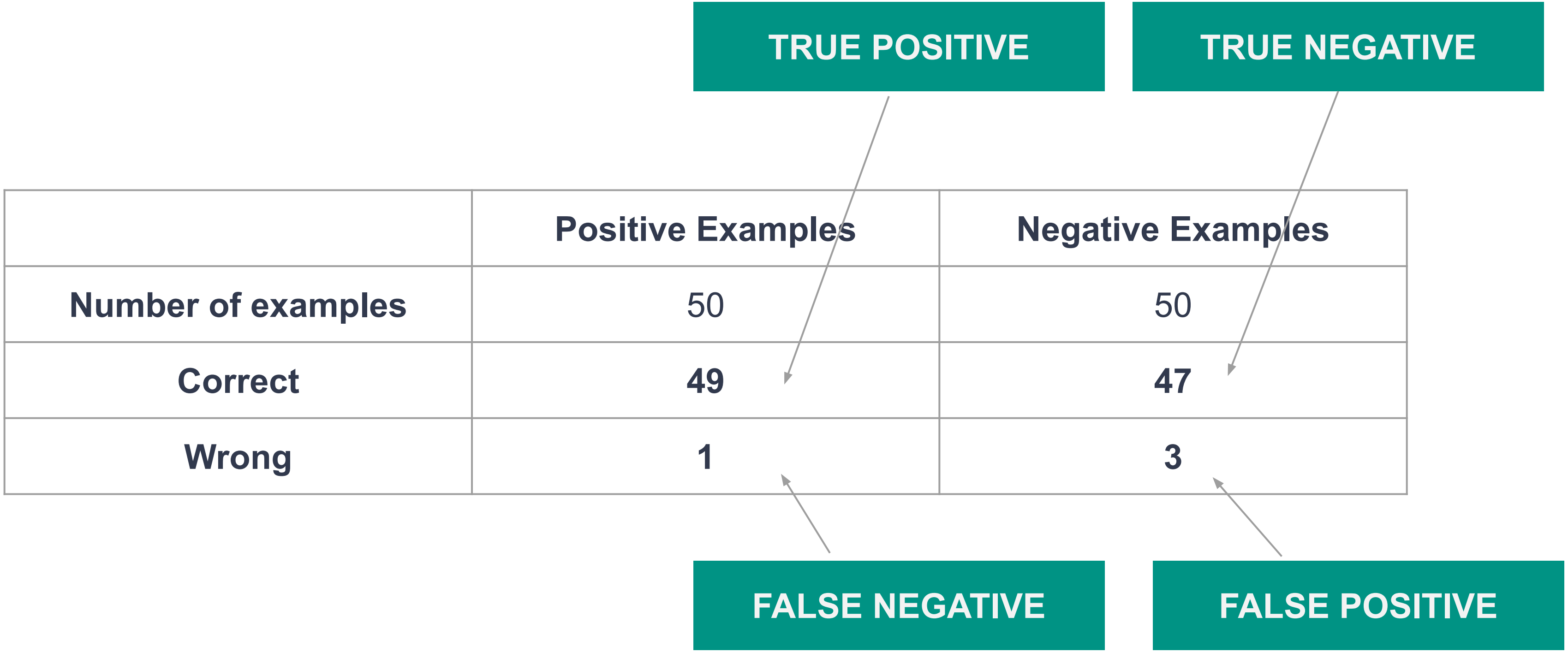
If a classifier labels everyone as not terrorist:

Accuracy = 99.9998%





Confusion Matrix



TRUE POSITIVE

TRUE NEGATIVE

	Positive Examples	Negative Examples
Number of examples	50	50
Correct	**49**	**47**
Wrong	**1**	**3**

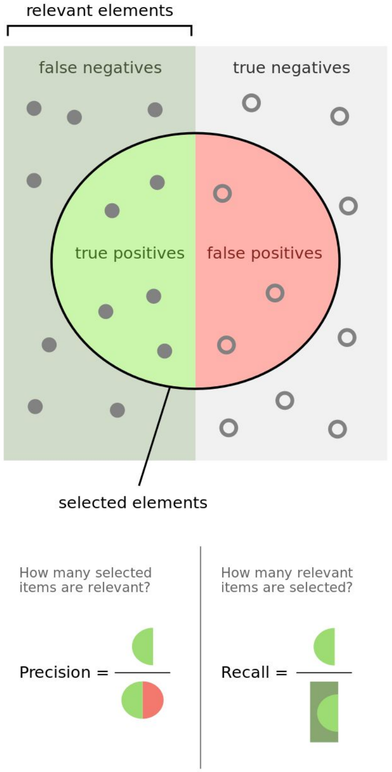
FALSE NEGATIVE

FALSE POSITIVE

Precision & Recall

$$\text{Precision} = \frac{\textit{True positive}}{\textit{True positive} + \textit{False positive}}$$

$$\text{Recall} = \frac{\textit{True positive}}{\textit{True positive} + \textit{False negative}}$$



Precision & Recall for a Balanced Dataset

	Positive	**Negative**
Num examples	53	47
Correct	**48**	**40**
Wrong	**5**	**7**

$$\textbf{Accuracy} = \frac{\text{True Positives (48) + True Negatives (40)}}{\text{All Examples (100)}} = 88\%$$

$$\textbf{Precision} = \frac{\text{True Positives (48)}}{\text{True Positives (48) + False Positive (7)}} = 87\%$$

$$\textbf{Recall} = \frac{\text{True Positives (48)}}{\text{True Positives (48) + False Negative (5)}} = 90\%$$

Precision & Recall for an imbalanced Dataset

PREDICTED/ ACTUAL	Positive	Negative
Positive	0 (TP)	0 (FP)
Negative	10 (FN)	90 (TN)

$$\text{Accuracy} = \frac{\text{True Positives (0) + True Negatives (99)}}{\text{All Examples (100)}} = 90\%$$

$$\text{Precision} = \frac{\text{True Positives (0)}}{\text{True Positives (0) + False Positive (0)}}$$

$$\text{Recall} = \frac{\text{True Positives (0)}}{\text{True Positives (0) + False Negative (10)}} = 0\%$$

If we label everyone as non-terrorists, our recall and precision are 0 or not computable

Precision & Recall for an imbalanced Dataset

PREDICTED/ACTUAL	Positive	Negative
Positive	10 (TP)	90 (FP)
Negative	0 (FN)	0 (TN)

$$\textbf{Accuracy} = \frac{\textbf{True Positives (10) + True Negatives (0)}}{\textbf{All Examples (100)}} = \textbf{10\%}$$

$$\textbf{Precision} = \frac{\textbf{True Positives (10)}}{\textbf{True Positives (10) + False Positive (90)}} = \textbf{10\%}$$

$$\textbf{Recall} = \frac{\textbf{True Positives (10)}}{\textbf{True Positives (10) + False Negative (0)}} = \textbf{100\%}$$

Recall is the ability to find relevant cases in a dataset!

Precision & Recall for an imbalanced Dataset

PREDICTED / ACTUAL	Positive	Negative
Positive	3 (TP)	0 (FP)
Negative	7 (FN)	90 (TN)

$$\textbf{Accuracy} = \frac{\text{True Positives (3) + True Negatives (90)}}{\text{All Examples (100)}} = 93\%$$

$$\textbf{Precision} = \frac{\text{True Positives (3)}}{\text{True Positives (3) + False Positive (0)}} = 100\%$$

Precision is the ability to find only the relevant data points

$$\textbf{Recall} = \frac{\text{True Positives (3)}}{\text{True Positives (3) + False Negative (7)}} = 30\%$$

Precision & Recall for an imbalanced Dataset

PREDICTED / ACTUAL	Positive	Negative
Positive	4 (TP)	3 (FP)
Negative	6 (FN)	87 (TN)

$$\text{Accuracy} = \frac{\text{True Positives (4) + True Negatives (87)}}{\text{All Examples (100)}} = 91\%$$

$$\text{Precision} = \frac{\text{True Positives (4)}}{\text{True Positives (4) + False Positive (3)}} = 57\%$$

$$\text{Recall} = \frac{\text{True Positives (4)}}{\text{True Positives (4) + False Negative (6)}} = 40\%$$

F1 Score

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Accuracy is used when the True Positives and True negatives are **more important** while **F1-score** is used when the False Negatives and False Positives are crucial.

Accuracy can be used when the class distribution is similar while **F1-score** is a **better** metric when there are imbalanced classes as in the above case

K-Fold Cross-Validation

Problem

Training Set

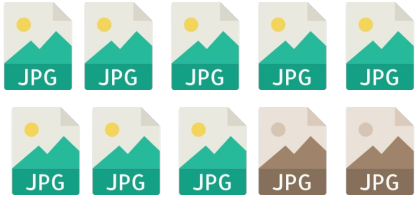
Validation Set

JPG
JPG
JPG
JPG
JPG
JPG
JPG
JPG
JPG
JPG
JPG
JPG
JPG
JPG
JPG

A part of the dataset is not used for training

K-Fold Cross Validation

Here, the test set is 20% of the dataset.
We loose ⅕ of the dataset.



Dataset

Run K-Experiments in which we:

- Choose a test set
- Train
- Test

After all the experiments, we average the results

Iteration 1	Test	Train	Train	Train	Train
Iteration 2	Train	Test	Train	Train	Train
Iteration 3	Train	Train	Test	Train	Train
Iteration 4	Train	Train	Train	Test	Train
Iteration 5	Train	Train	Train	Train	Test

Thank

You

jeremycohen.podia.com

https://www.linkedin.com/in/jeremycohen2626/