UNIVERSITY OF TORONTO

Engineering

Supervised ML

Supervised Learning Algorithms

- K-Nearest Neighbors
- Decision Trees
- Linear Model for Regression
- Linear Model for Classification
- Naïve Bayes Method

K-Nearest Neighbors

- Understanding the algorithm
- Analyzing the decision boundaries
- Challenges

Nearest Neighbours

- Goal: To find the class of a new input vector, $\boldsymbol{x}$
- How can we find the nearest neighbors and use them for our task?
- **Formulation:** we will apply Euclidean distance to find neighbors.

$$\left\|x^a - x^b\right\|_2 = \sqrt{\sum_{j=1}^{d} \left(x_j^a - x_j^b\right)^2}$$

K-Nearest Neighbours Algorithm

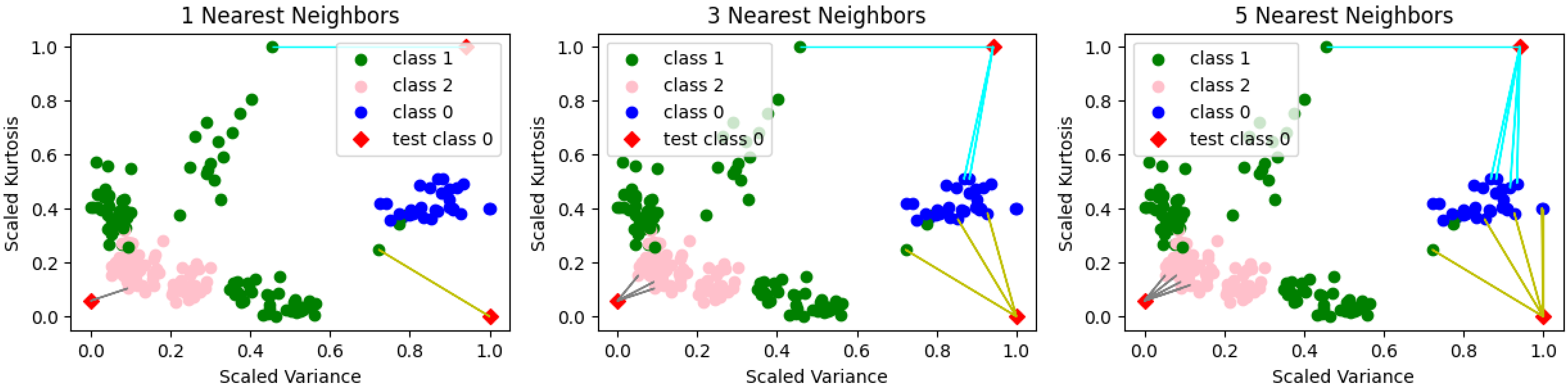
1. Find k examples, $\{x_i, t_i\}$, closest to the test sample x.
2. Output Calculation:
   – Regression output is defined as follows:

$$y = \frac{1}{k}\sum_{i=1}^{k} t_i$$

   – Classification output is defined as follows:

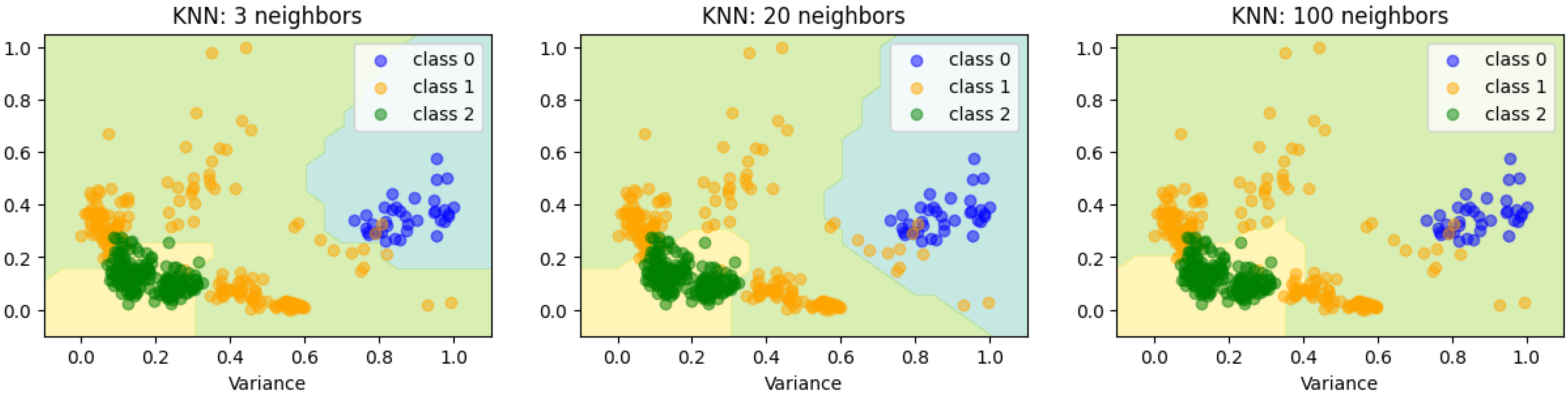
$$y = argmax_t \sum_{i=1}^{k} \mathbb{I}\{t = t_i\}$$

K-Nearest Neighbours



- What can you observe from these plots?

Nearest Neighbours: Decision Boundaries



- What is the effect of increasing the number of neighbors?

K-Nearest Neighbours: Trade-offs

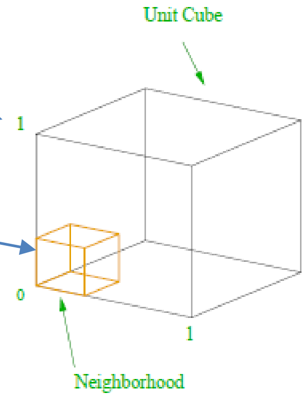
- Small k
  - **Captures local patterns**
  - **Overfitting issues**
- Large k
  - **Stable predictions**
  - **Underfitting issues**
- Recommended: $k = n^{\frac{2}{2+d}}$, where n: no. of data points; d: no. of dimensions.

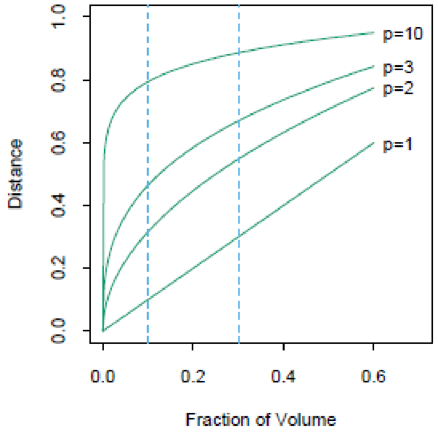
Curse of Dimensionality

- In high dimensions, most points are further away.

To cover 10% of the volume, we need to cover 47% of the side length for a 3-d space.

$$V = x^3$$
$$V_f = (kx)^3$$





Poor performance as dimensions increase.

Image Source

Challenges

- Requires normalization/scaling of features.
- Requires balanced data
- Computationally expensive.
  - Calculation of Euclidean distances
  - Sorting of distances

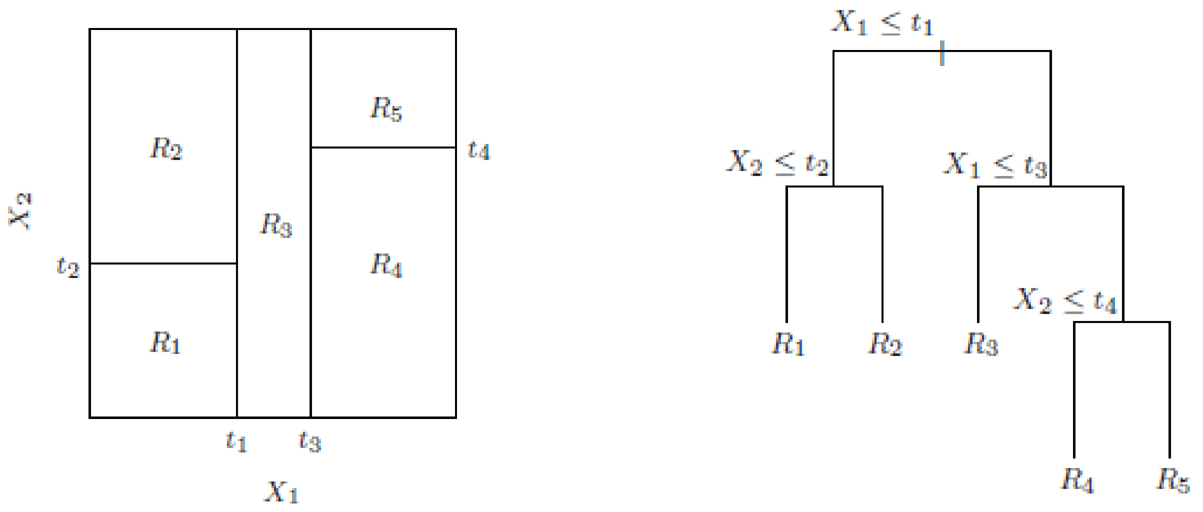
UNIVERSITY OF TORONTO | Engineering

Decision Trees

Decision Trees

- Tree-based methods partition the feature spaces into a set of rectangles.
- Splitting is continued until some stopping rule is applied.

Continuous Input, Discrete Output

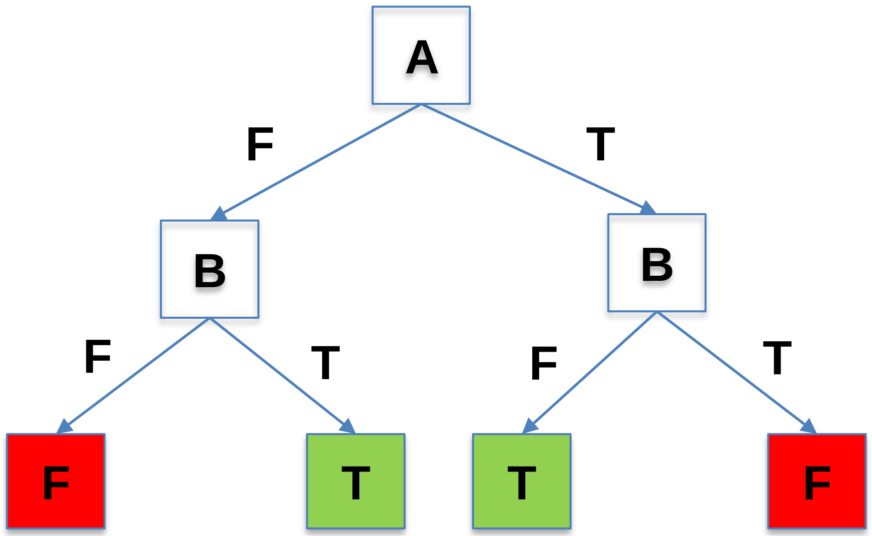


[Image Source](#)

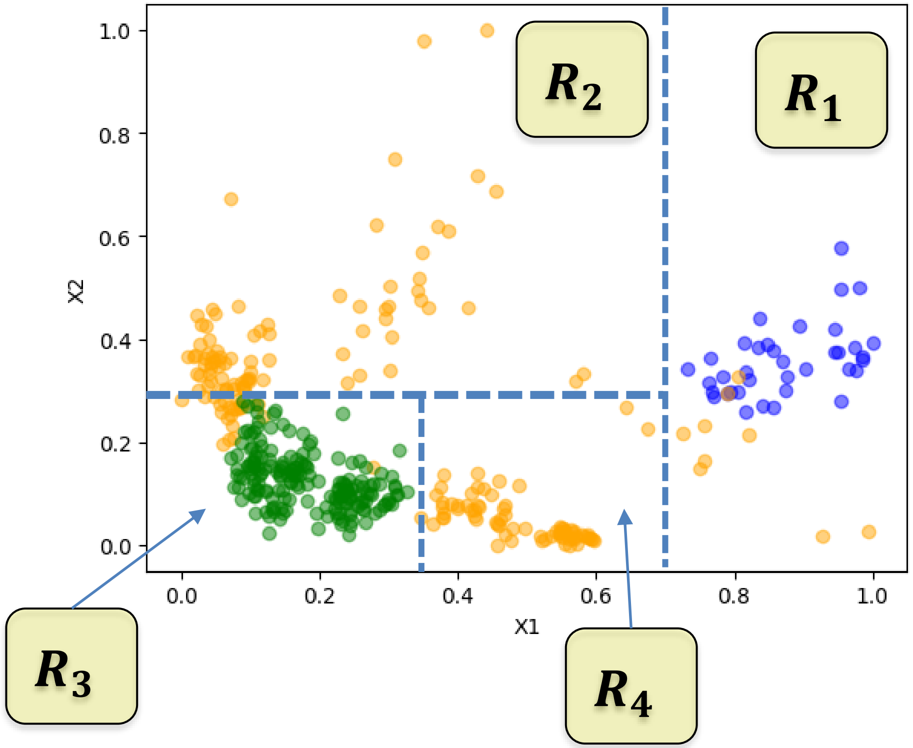
Decision Trees: Discrete Attributes

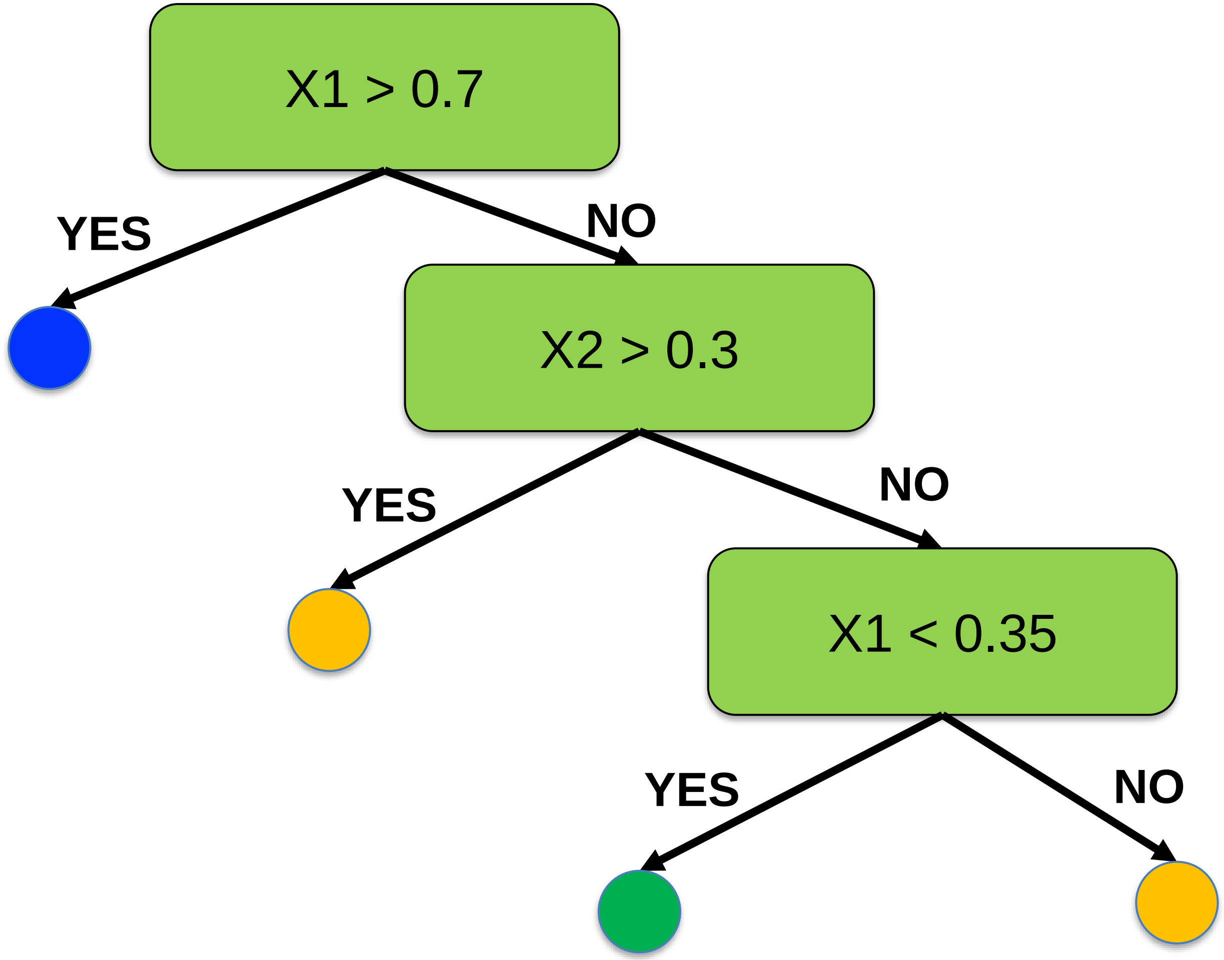
- Discrete Input, Discrete Output

A	B	A XOR B
F	F	F
F	T	T
T	F	T
T	T	F



Defining Regions: Continuous Attributes





Decision Tree: Classification and Regression

- Let's consider that the training examples in the region R_m are:

$$\{(\mathrm{x}^{m_1}, \mathrm{t}^{m_1}), \ldots, (\mathrm{x}^{m_k}, \mathrm{t}^{m_k})\}$$

- Classification tree:
  - Output is $y \in (1, 2, \ldots, C)$.
  - Leaf output y^m is the frequently occurring target value in that split.

$$y^m \leftarrow \underset{t \in \{1,2,\ldots,C\}}{\operatorname{argmax}} \sum_{m_i} \mathbb{I}\{t = t^{m_i}\}$$

Decision Tree: Classification and Regression

- Let's consider that the training examples in the region R_m are:

$$\{(\mathrm{x}^{m_1}, \mathrm{t}^{m_1}), \dots, (\mathrm{x}^{m_k}, \mathrm{t}^{m_k})\}$$

- Regression tree:
  - Output is $y \in \mathbb{R}$.
  - Leaf output y^m is the mean of the target value in that region.

Learning a Classification Tree

- How to select the attribute for splitting?

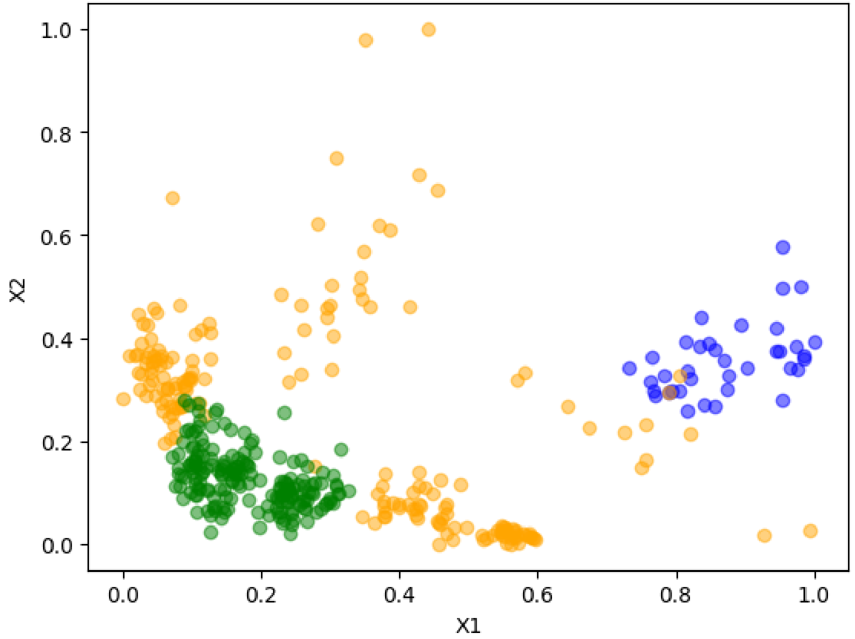
- When should the splitting stop?

Learning a Classification Tree

How to select the attribute for splitting?

- Let's first define the term: **accuracy gain.**
- We define splits such that the misclassification error (accuracy) reduces.

Region: R



Note that no. of samples for each class are:

Class 0 (blue) = 37

Class 1(orange) = Class 2 (green) = 186

Learning a Classification Tree

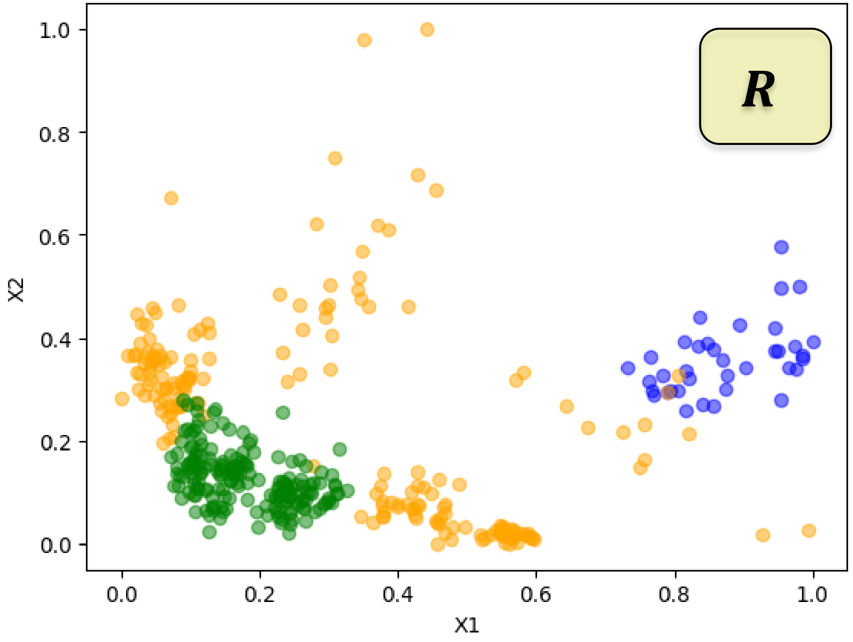
How to select the attribute for splitting?

- Loss before the split: $L(R)$
- Misclassification loss after the split:

$$\frac{|R_1|}{|R|}L(R_1) + \frac{|R_2|}{|R|}L(R_2)$$

- Accuracy gain:

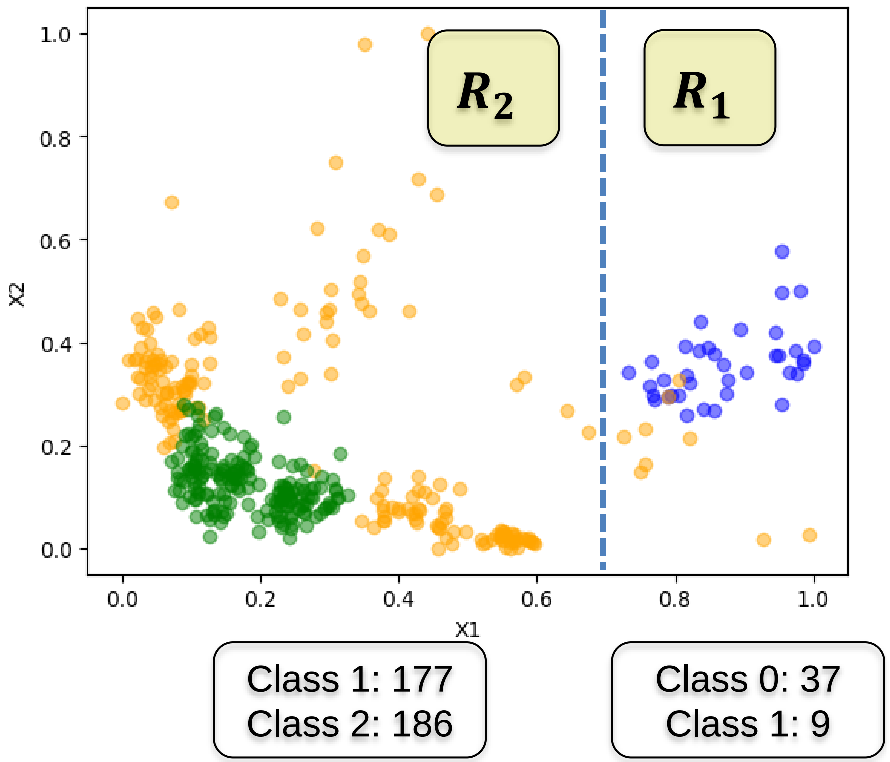
$$L(R) - \frac{|R_1|L(R_1) + |R_2|L(R_2)}{|R|}$$



Learning a Classification Tree

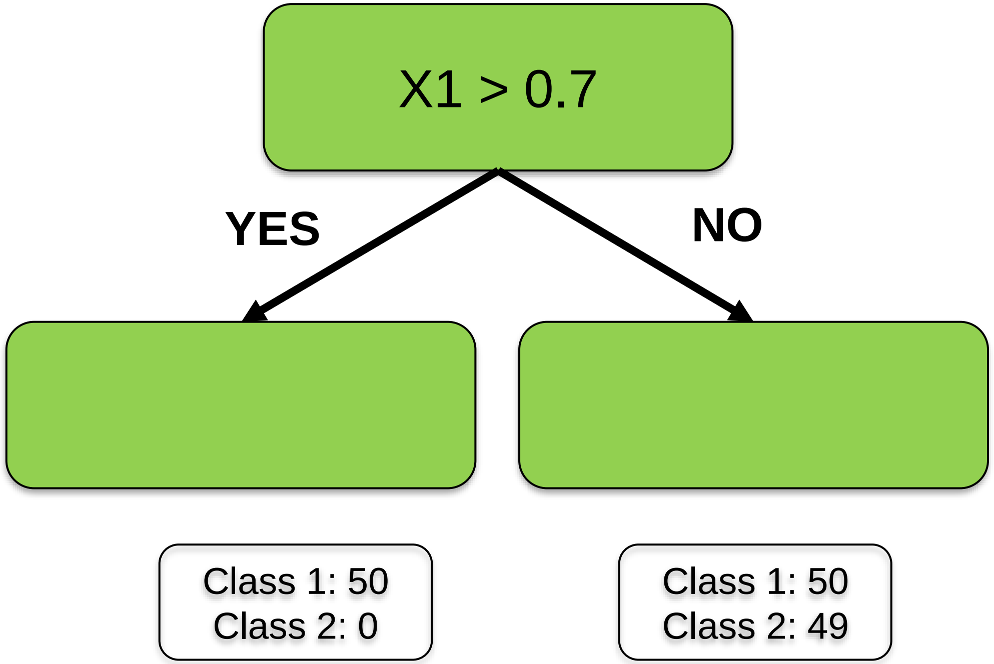
What will be the accuracy gain for this split?

Note: Misclassification Loss = $\frac{|R_1|}{|R|}L(R_1) + \frac{|R_2|}{|R|}L(R_2)$



Learning a Classification Tree

Why is accuracy not always a good measure to decide the split?



- Is such a split useful?
- What will be the accuracy gain for this split?

Learning a Classification Tree

How to select a good split?

- Low Uncertainty: All examples in the leaf have same class.
- High Uncertainty: The leaf node cannot separate the classes efficiently.

To measure uncertainty, we can use counts at leaves to define probability distributions and apply concepts of information theory.

Quantifying Uncertainty

Entropy: It is a measure of randomness. High entropy means randomness is higher, meaning challenging to classify.

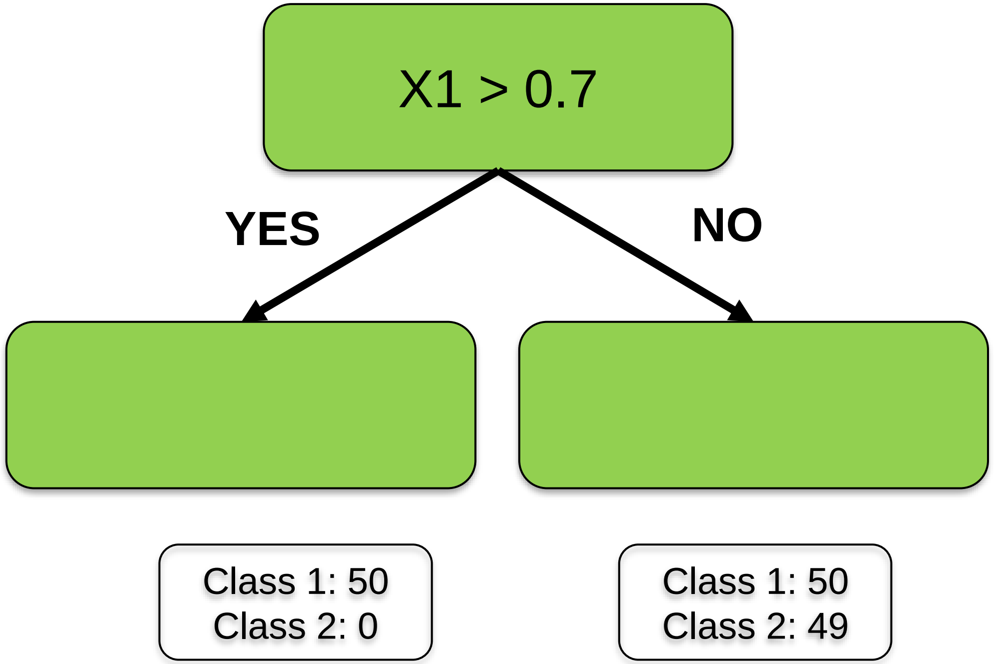
$$H(x) = -\sum_{x \in X} p(x) \log p(x)$$

Information Gain: $\mathrm{IG}(Y|B) = H(Y) - H(Y|B)$

Note: Variables are selected with thresholds such that highest gain is attained.

Learning a Classification Tree

What will be the information gain for this split?

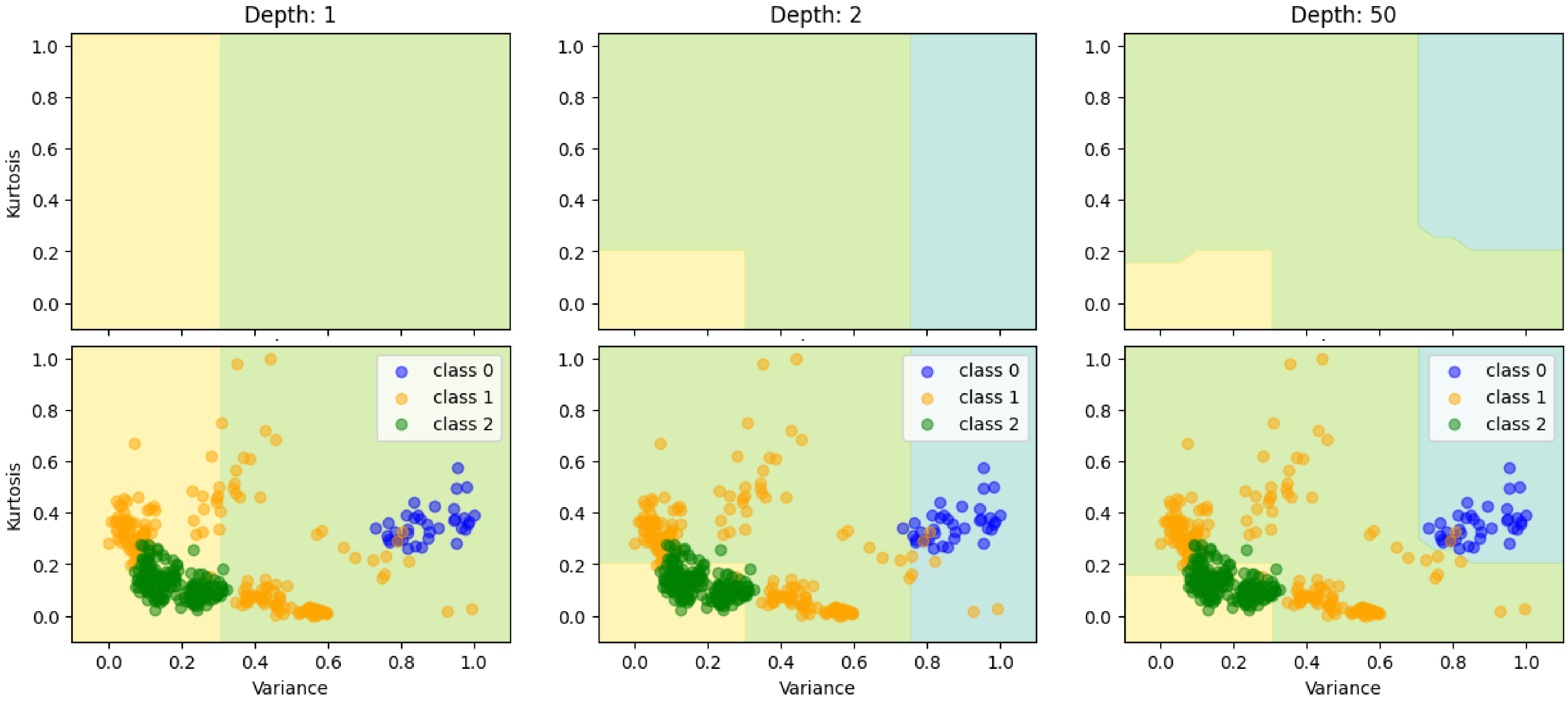


Learning a Classification Tree

- Requires: A training set.
- Recursive approach: Keep splitting on the most informative feature.
- Termination Strategy:
  - End if the region contains samples from the same class. **Overfits, Expensive**
  - Maximum Depth
  - Minimum Samples per leaf
  - Pruning Techniques
- Note: Gini Index can be used instead of Information Gain.

$$\textbf{Gini Index} = 1 - \sum_{i=1}^{K} p_i^{\,2}$$

Decision Boundaries



Advantages

- Robust to noise, scale of features.
- Can extract essential features from a highly dimensional dataset.
- More interpretable.
- Computationally efficient when compared with KNN.

Regression with Linear Models

Linear Regression

- Model: Uses Linear Function over the input space.

$$y = f(x) = \sum_j w_j x_j + b$$

  - y is the output from the model
  - w is the weight matrix
  - b is the bias (or intercept)

Linear Regression: Loss Function

- How to determine the quality of the predictions?
- Loss function is defined to measure how close the predictions are.
- Squared error:

$$\mathcal{L}(y, t) = \frac{1}{2}(y - t)^2$$

- To get accurate predictions, we would like to have lower **residual**.

Linear Regression: Loss Function

- Cost Function: This is the average loss for all the examples.

$$\mathcal{J}(\mathrm{w}, b) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\left(y^{(i)}, t^{(i)}\right)$$

$$\mathcal{J}(\mathrm{w}, b) = \frac{1}{2N}\sum_{i=1}^{N}\left(\mathrm{w}^{T}\mathrm{x}^{(i)} + b - t^{(i)}\right)^{2}$$

Linear Regression: Vector Notation

- To find the best fit line, we need to minimize the cost function.

$$\text{Minimize: } \mathcal{J}(\mathrm{w}, b) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\left(y^{(i)}, t^{(i)}\right)$$

- Vectorize:

$$\mathrm{X} = \begin{bmatrix} 1 & [x^{(1)}]' \\ 1 & [x^{(2)}]' \\ 1 & \vdots \end{bmatrix} \in \mathbb{R}^{N \times D+1}, \mathrm{w} = \begin{bmatrix} b \\ w_1 \\ \vdots \end{bmatrix} \in \mathbb{R}^{D+1}$$

$$y = \mathrm{X}\mathrm{w}$$

Linear Regression: Direct Solution

- We know that the minimum cost occurs when partial derivatives are zero.

$$\frac{\partial \mathcal{J}}{\partial w_j} = 0, \qquad \frac{\partial \mathcal{J}}{\partial b} = 0$$

- If direct solution is not possible, then we aim to reduce them as much as possible using Gradient Descent.

Linear Regression: Direct Solution

- Given:

$$\mathcal{J} = \frac{1}{2N}\|y - \mathrm{t}\|^2 \Longrightarrow \mathcal{J} = \frac{1}{2N}(\mathrm{Xw} - \mathrm{t})\,'(\mathrm{Xw} - \mathrm{t})$$

- We have:

$$\frac{\partial \mathcal{J}}{\partial \mathrm{w}} = \frac{1}{N}\mathrm{X}'(\mathrm{Xw} - \mathrm{t}) = 0 \;\Longrightarrow\; (\mathrm{X}'\mathrm{X})\mathrm{w} = \mathrm{X}'\mathrm{t}$$

$$\mathbf{w}^{\mathbf{LS}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{t}$$

Probabilistic Interpretation of Squared Error

- Why do we measure the quality of fit using Squared Error?

Probabilistic Interpretation of Squared Error

- Suppose that: $t^{(i)} \sim p\left(y \middle| x^{(i)}, \mathrm{w}\right)$
- $\mathcal{D} = \left\{\left(x^{(1)}, t^{(1)}\right), \dots, \left(x^{(N)}, t^{(N)}\right)\right\}$
- The likelihood function is $\Pr(\mathcal{D}|\mathrm{w})$
- We need to find the parameters such that it maximizes the likelihood function.

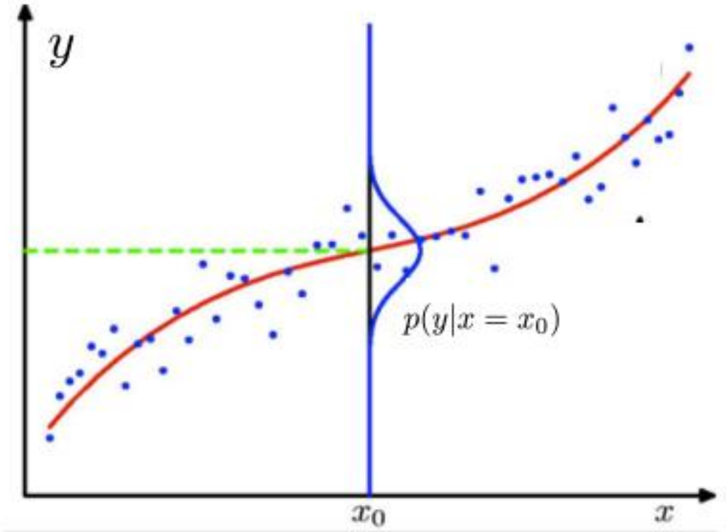


Image Source

Maximum Likelihood Estimation

- For independent samples, the likelihood function is the product of likelihoods.

$$p\left(t^{(1)}, t^{(2)}, \ldots, t^{(N)} | \mathrm{x}^{(1)}, \mathrm{x}^{(2)}, \ldots, \mathrm{x}^{(N)}, \mathrm{w}\right) = \prod_{i=1}^{N} p\left(t^{(i)} | \mathrm{x}^{(i)}, \mathrm{w}\right) = L(\mathrm{w})$$

- For computational efficiency, we minimize the negative log-likelihood.

$$l(\mathrm{w}) = -\log L(\mathrm{w}) = -\sum_{i=1}^{N} \log p\left(t^{(i)} | \mathrm{x}^{(i)}, \mathrm{w}\right)$$

Squared Error

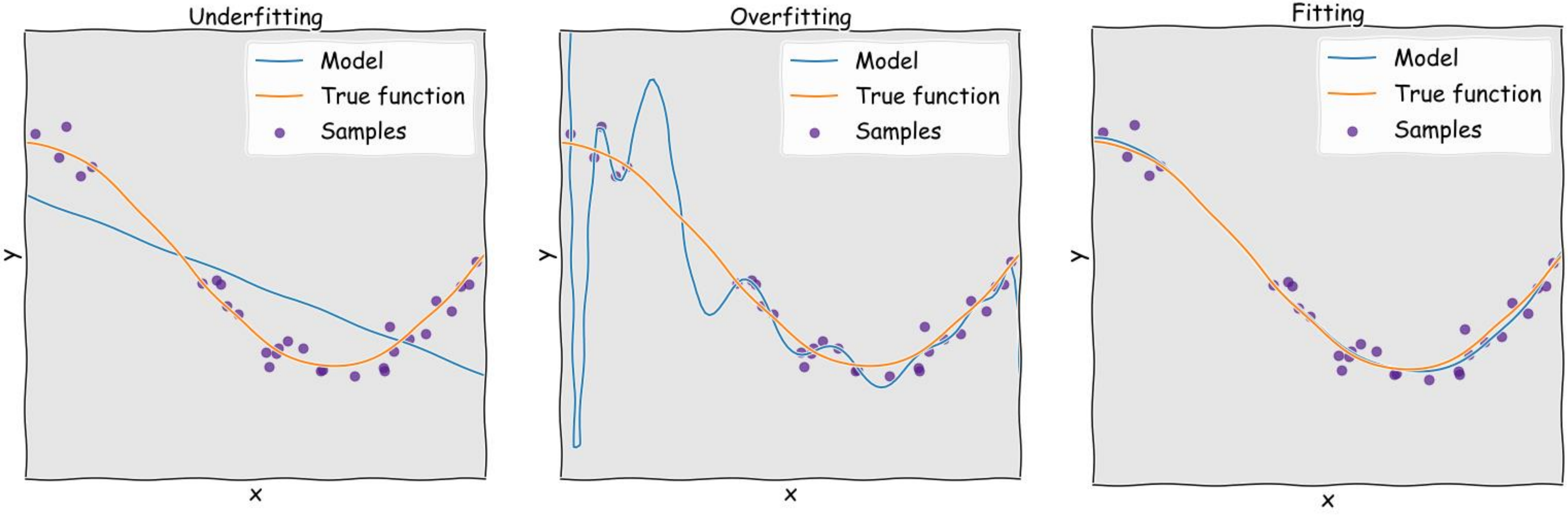
- Suppose that the residual term, $y - t$, is sampled from normal distribution with mean 0 and variance σ^2 then:

$$p\left(t^{(i)}|\mathrm{x}^{(i)},\mathrm{w}\right) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left\{-\frac{1}{2\sigma^2}\left(t^{(i)} - \mathrm{w}^T\mathrm{x}^{(i)}\right)^2\right\}$$

$$-\log p\left(t^{(i)}|\mathrm{x}^{(i)},\mathrm{w}\right) = \frac{1}{2\sigma^2}\left(t^{(i)} - \mathrm{w}^T\mathrm{x}^{(i)}\right)^2 + \log\sqrt{2\pi\sigma^2}$$

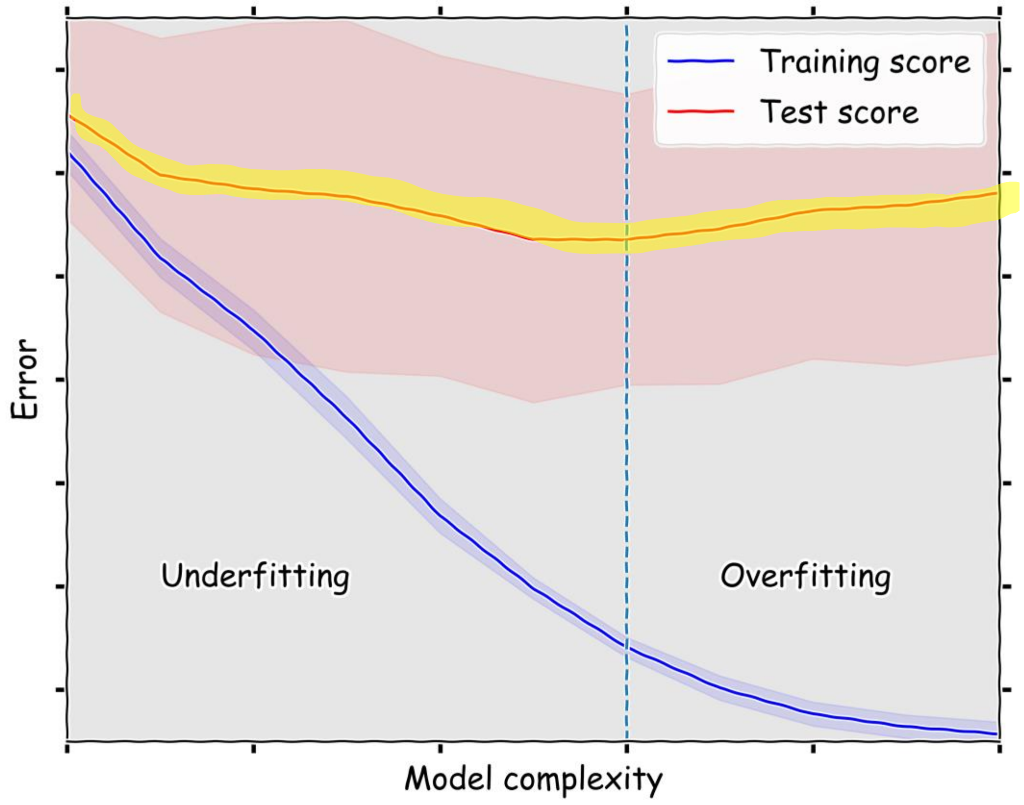
$$l(\mathrm{w}) = -\sum_{i=1}^{N}\log p\left(t^{(i)}|\mathrm{x}^{(i)},\mathrm{w}\right) = \frac{1}{2\sigma^2}\sum_{i=1}^{N}\left(t^{(i)} - \mathrm{w}^T\mathrm{x}^{(i)}\right)^2 + C$$

Under-fitting and Over-fitting



Under-fitting and Over-fitting

- When do we under- or over-fit?
  - In relation with data
    - If model is too simple, **Under-fitting**
    - If model is too complex, **Over-fitting**



- How to know we are over- or under-fitting during training?

Under-fitting and Over-fitting

- Polynomial with order M
  - Training data N=10 (points)
  - Sign curve

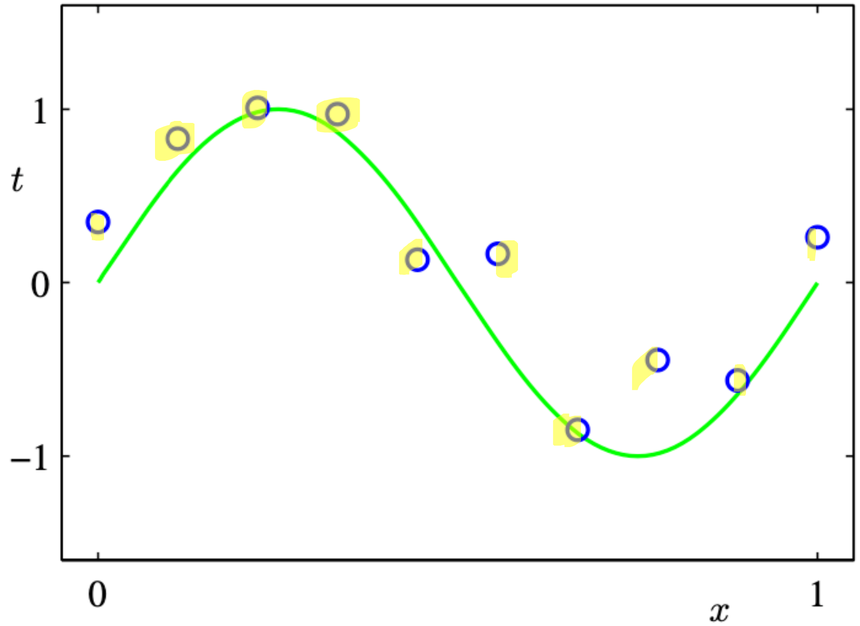
$$t = \sin(2\pi x)$$

  - Model to be used

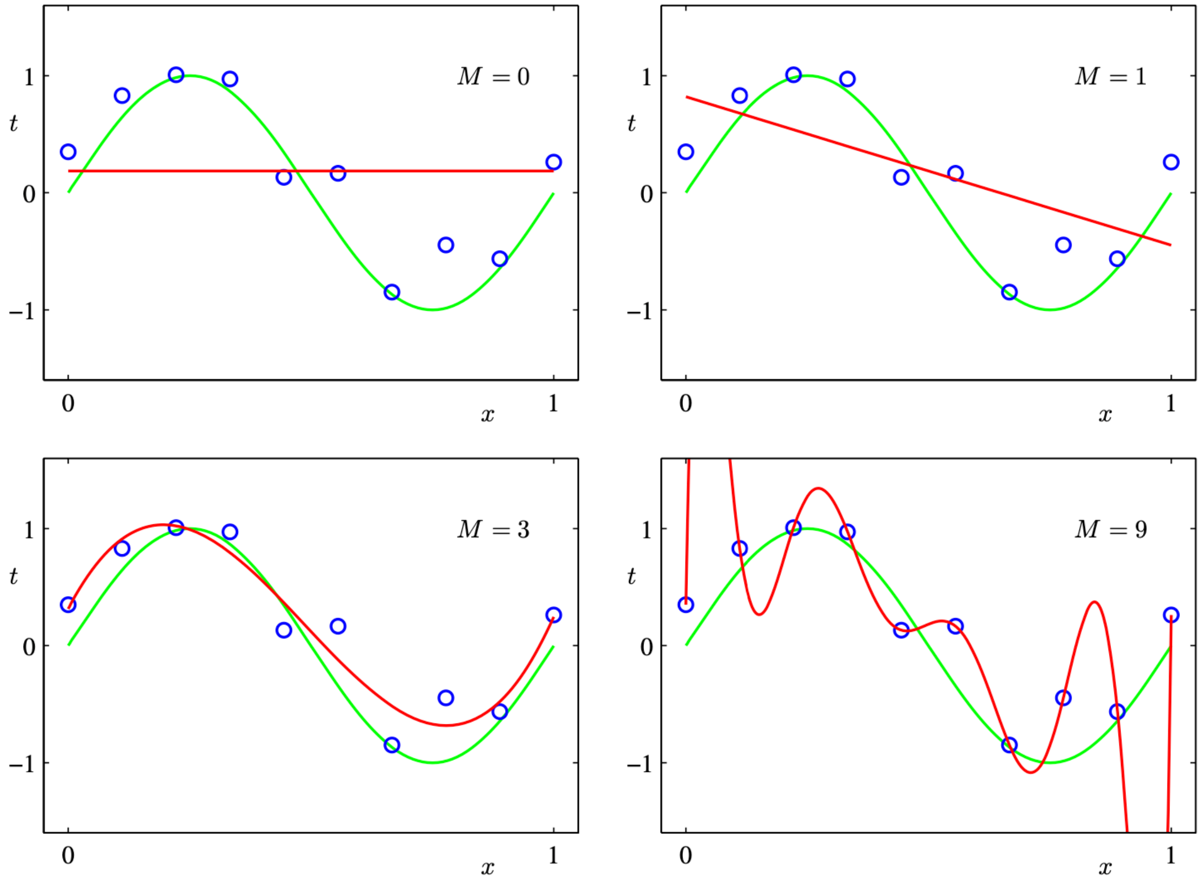
$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

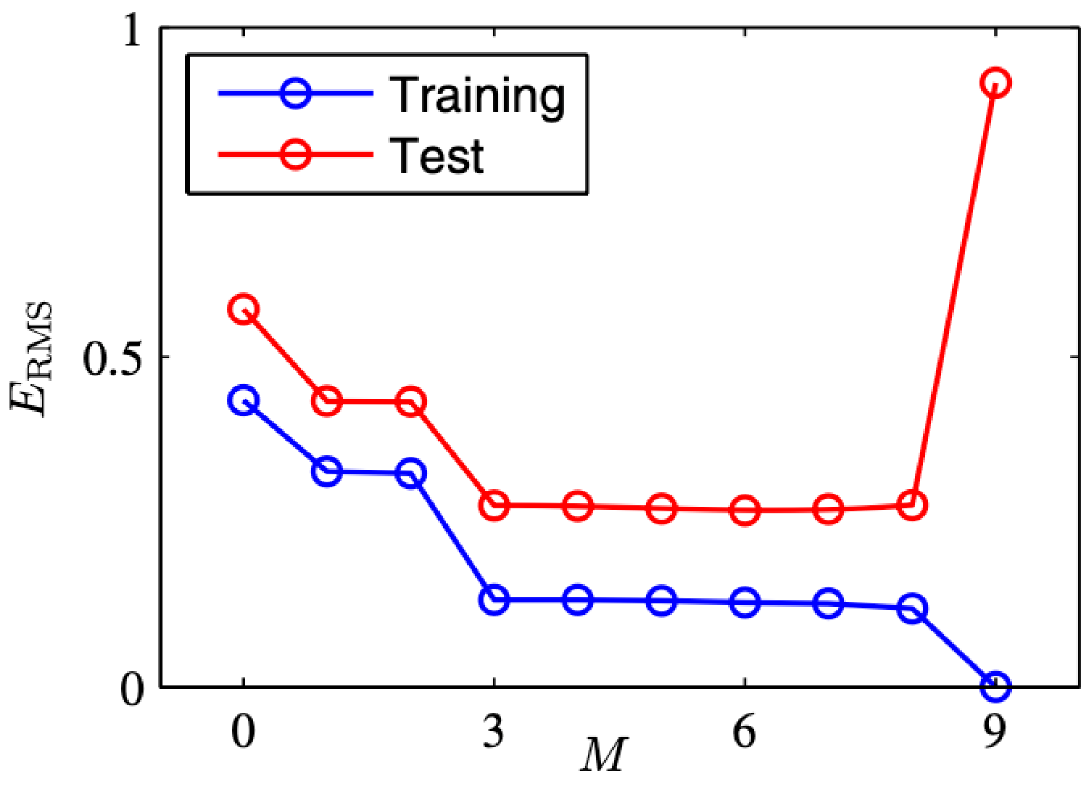
  - Error to minimize

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2$$



Under-fitting and Over-fitting





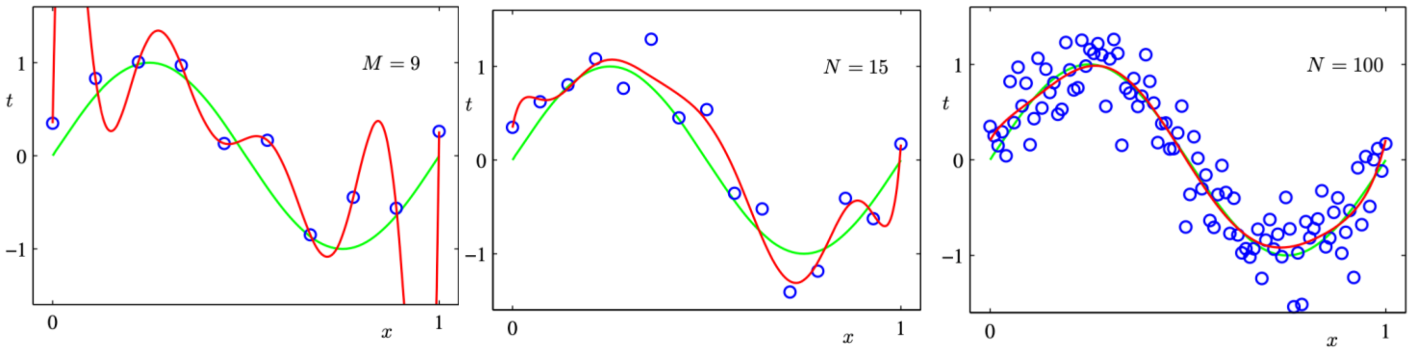
Over-fitting

- What's happening when over-fit?

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M$$

	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

When M=9



N=10 N=15 N=100

Over-fitting

- Solutions
  - More data (the more, the better)
  - Regularization

Keep the coefficients **SMALL**!

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

Regularization

- Ridge regression
  - Penalize large values of parameters

Model:

$$f(x;\boldsymbol{w}) = w_0 + w_1 x_1 + \cdots + w_d x_d$$

Loss function: Residual Sum of Squares + penalty term

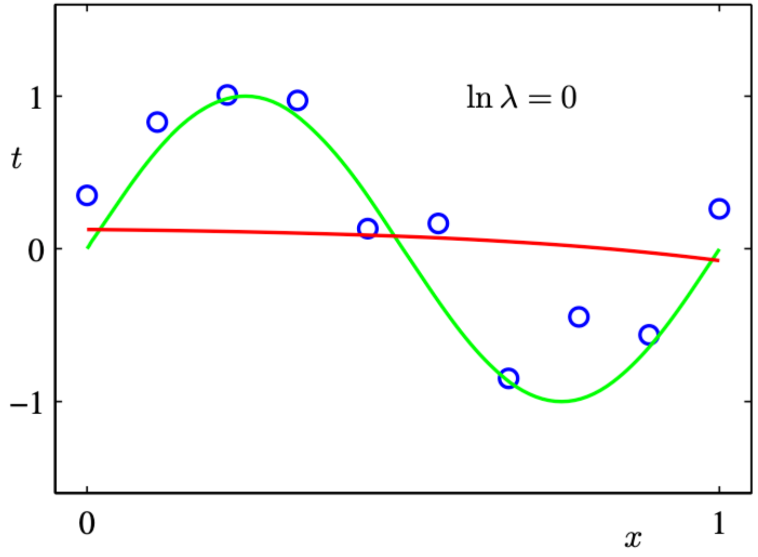
$$RSS(\boldsymbol{w}) = \sum_{i=1}^{n} \left(y_i - f(x_i;\boldsymbol{w})\right)^2 + \lambda \sum_{j=0}^{d} w_j^2$$

Over-fitting

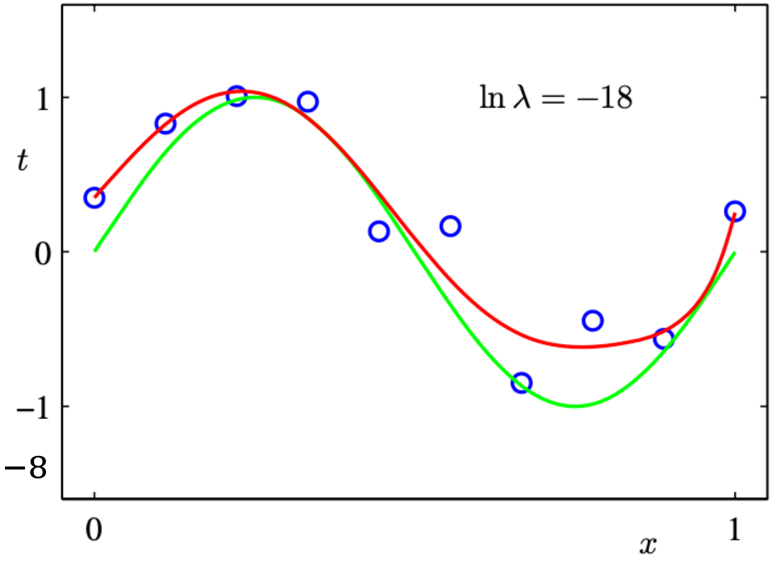
- Ridge Regression

$$RSS(\boldsymbol{w}) = \sum_{i=1}^{n}\left(y_i - f(x_i;\boldsymbol{w})\right)^2 + \lambda\sum_{j=0}^{d} w_j^2$$

	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01



$\lambda = 1$



$\lambda = 1.52 \times 10^{-8}$

Regularization

- Lasso regression
  - Penalize large values of parameters

Model:

$$f(x;\boldsymbol{\beta}) = \beta_0 + \beta_1 x_1 + \cdots + \beta_d x_d$$

Loss function: Residual Sum of Squares + penalty term

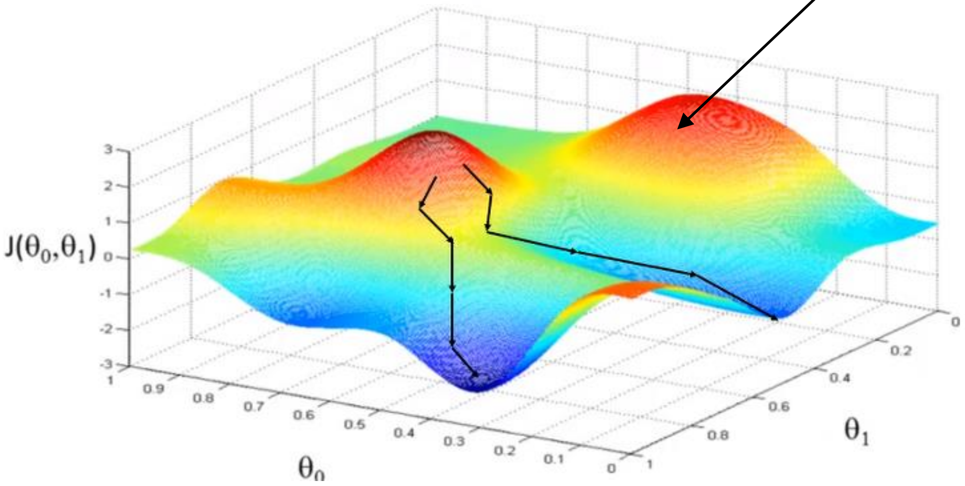
$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(y_i - f(x_i;\boldsymbol{\beta})\right)^2 + \boldsymbol{\lambda} \sum_{j=0}^{d} |\beta_j|$$

Gradient Descent

Gradient Descent for ML

- The most used learning algorithm especially for high dimensional data.

$$MSE = \frac{1}{2n}\sum(y_i - \hat{y})^2$$



Repeat until convergence {

$$\theta_{j+1} \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} \mathrm{MSE}(\theta)$$

}

$$\frac{\partial}{\partial \theta_j} \mathrm{MSE}(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (\theta^T \cdot x_i - y_i)[x_i]_j$$

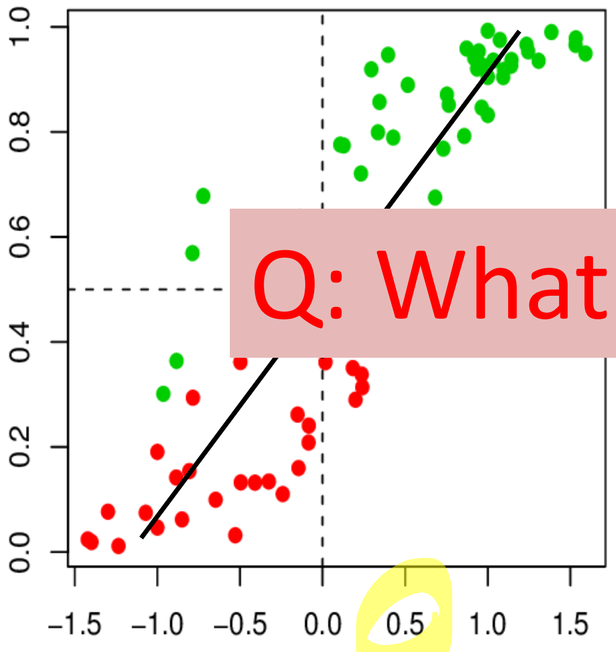
Why Gradient Descent?

- For linear regression, even if we can get the direct solution, sometimes Gradient Descent is preferred.
- Computational Efficiency:
  - A huge difference is observed when there is more than one dimension in the input space.
  - Complexity of matrix inversion in direct solution: $\mathcal{O}(D^3)$
  - It can be applied to a variety of models.

Classification with Linear Models

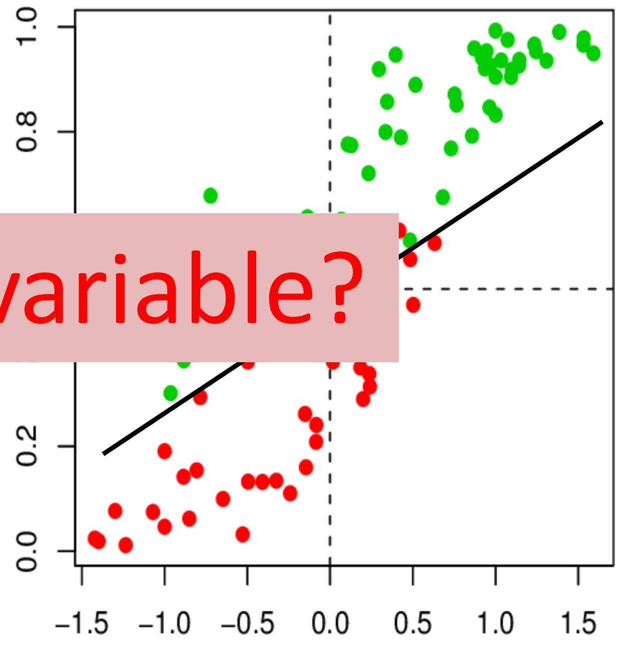
Logistic Regression

❖ From Regression to Classification



Q: What is the target variable?

Regression



Classification

Linear Regression

- What was the loss function used in regression models?

- Can we use the same in this case?

$$z = \mathrm{X}w + b$$

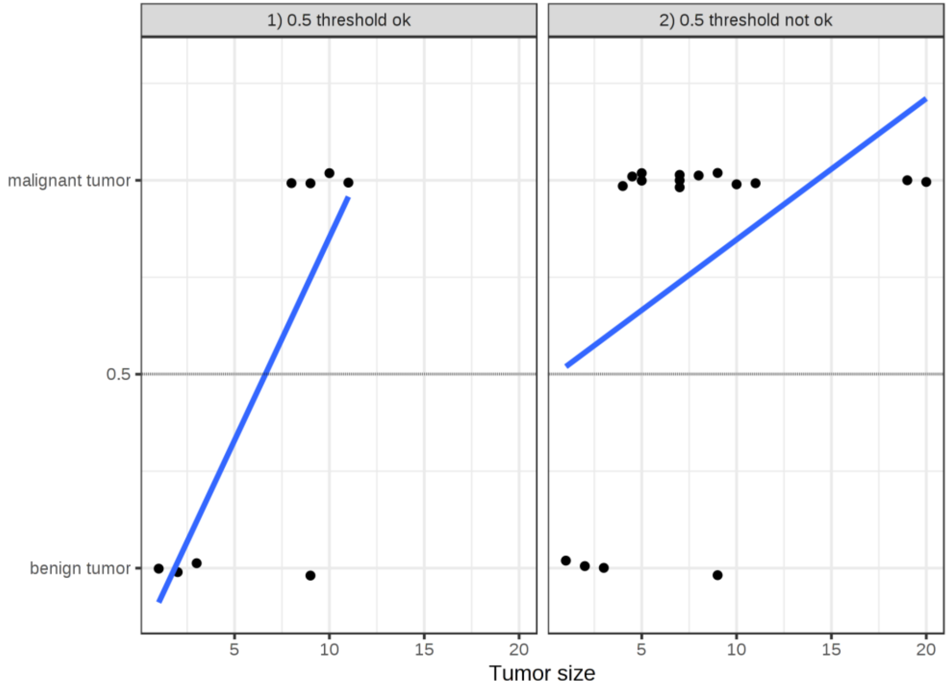
$$\mathcal{L}_{SE} = \frac{1}{2}(z - t)^2$$

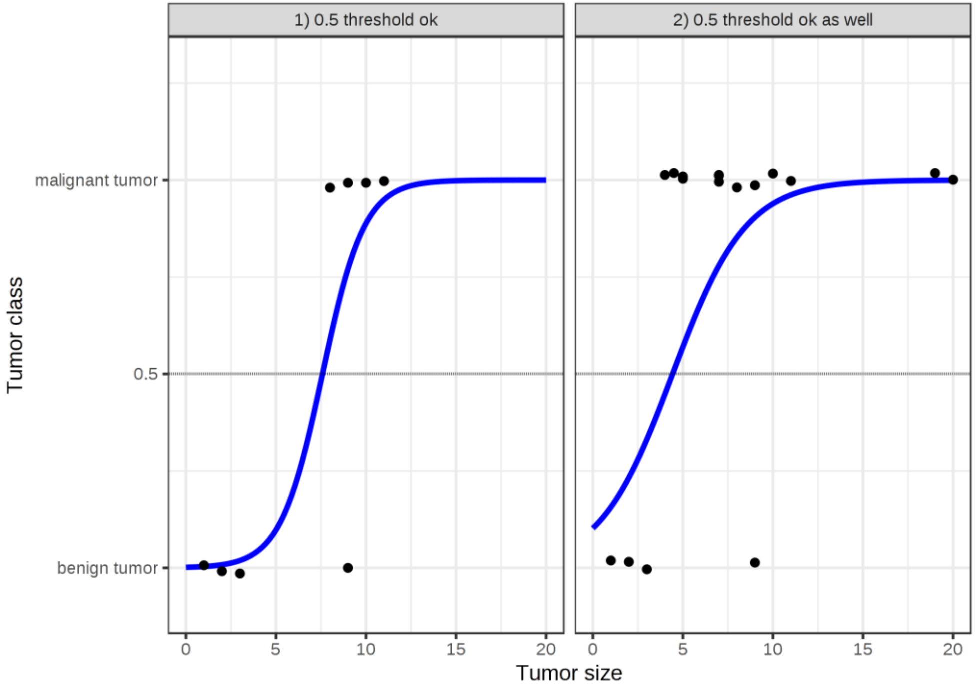
Linear Regression

- We need to output either 0 or 1 (i.e., class labels).
- How can we convert the continuous output to our desired output?
- What is the problem with squared error loss and linear function?

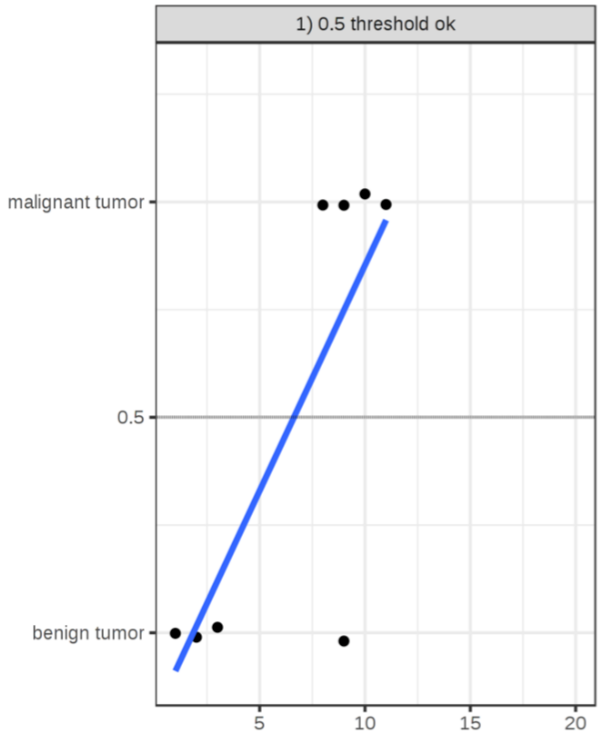
Logistic Regression

- Binary Label: True (1) or False (0)

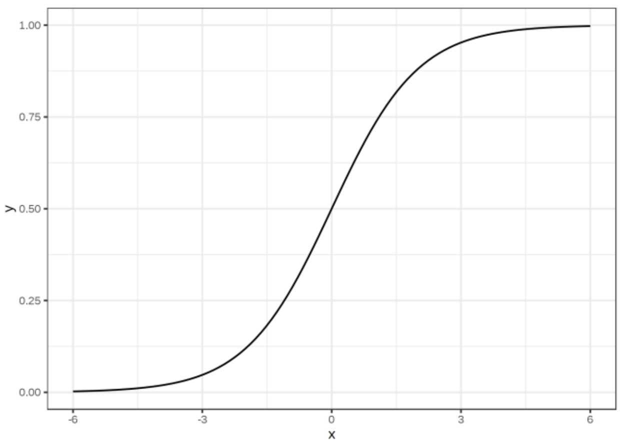




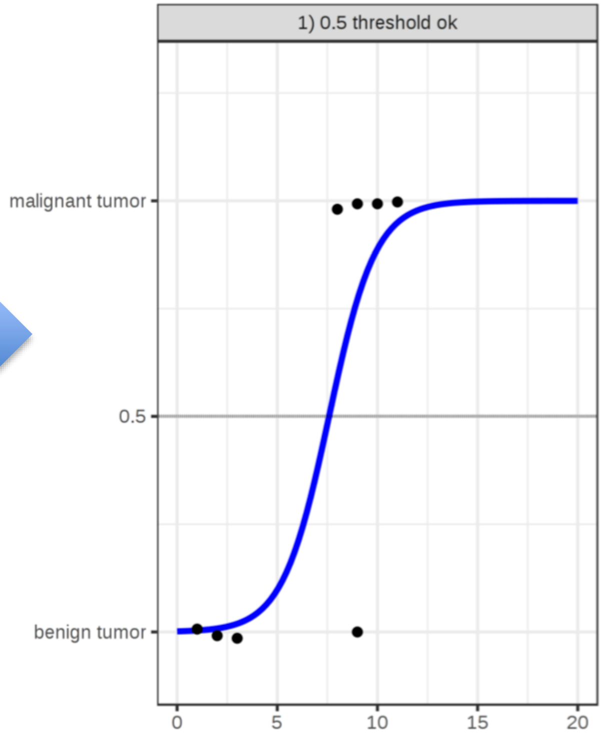
Logistic Regression



Logistic Function



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



$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b = w^T x + b$$

$$P(y = 1) = \frac{1}{1 + e^{-w^T x + b}}$$

Logistic Function

- Linear Model $z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b = w^T x + b$

- Logistic Function $\sigma(z) = \dfrac{1}{1 + e^{-z}}$

Positive: $\hat{y} = 1$ $P(\text{positive}|z) = \sigma(z) = \dfrac{1}{1 + e^{-z}}$

Negative: $\hat{y} = 0$ $P(\text{negative}|z) = 1 - \sigma(z) = \dfrac{e^{-z}}{1 + e^{-z}}$

- Decision Boundary $\hat{y} = \begin{cases} 1 & \text{if } P(y = 1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$

Logistic Function

- Example: $w = (1.2, 0.023, -2.4); b = -205$

Sensor	Value
Temperature	125
Vibration	2450
Pressure	1.05

$$P(\text{fail}|x) = \sigma(w^T x + b)$$
$$= \sigma((\_\_\_\_\_\_\_\_\_\_\_\_) \cdot (\_\_\_\_\_\_\_\_\_\_\_\_) - 205)$$
$$= \sigma(-1.17)$$
$$= 0.2369$$

Sensor	Value
Temperature	125
Vibration	2550
Pressure	0.85

$$P(\text{fail}|x) = \sigma(w^T x + b)$$
$$= \sigma(1.61)$$
$$= 0.8334$$

Logistic Function with Squared Error Loss

- Loss Function definition:

$$z = \mathrm{X}w + b$$

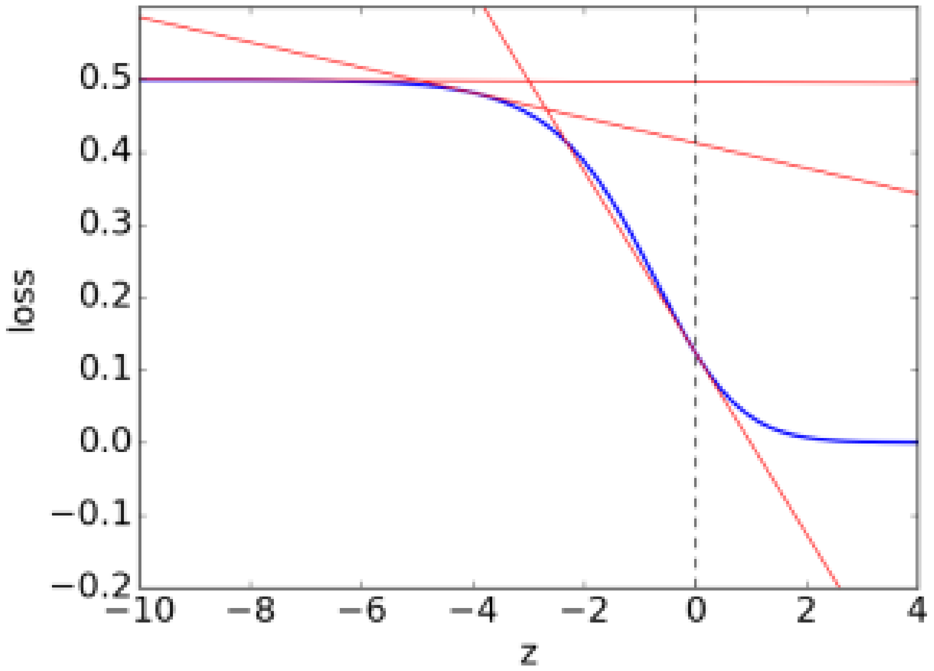
$$y = \sigma(z)$$

$$\mathcal{L}_{SE} = \frac{1}{2}(y - t)^2$$

- What is the problem with squared error loss and logistic regression?

Logistic Function with Squared Error Loss

- Let's plot the loss function, assuming t = 1.
- What problem do you observe?



Logistic Regression

- Logistic Function

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

- Probability of "positive" vs "negative" given $\hat{y}$

$$P(\text{positive}|z) = \sigma(z) = \frac{1}{1+e^{-z}} \qquad P(\text{negative}|z) = 1-\sigma(z) = \frac{e^{-z}}{1+e^{-z}}$$

- Log odds ratio of $P(\text{positive}|z)$ over $P(\text{negative}|z)$, easier to understand.

$$\log\frac{P(\text{positive}|z)}{P(\text{negative}|z)} = \log\frac{\frac{1}{1+e^{-z}}}{\frac{e^{-z}}{1+e^{-z}}} = \log\frac{1}{e^{-z}} = \log(1) - \log(e^{-z}) = z = b + w_1x_1 + \cdots w_nx_n$$

Learning Logistic Function

- Cross Entropy Loss

Given a prediction $\hat{y} = \sigma(w^T x + b)$ and the correct target y (which is 0 or 1)

$$L(\hat{y}, y) = \text{How much } \hat{y} \text{ differs from the true } y$$

Conditional Likelihood

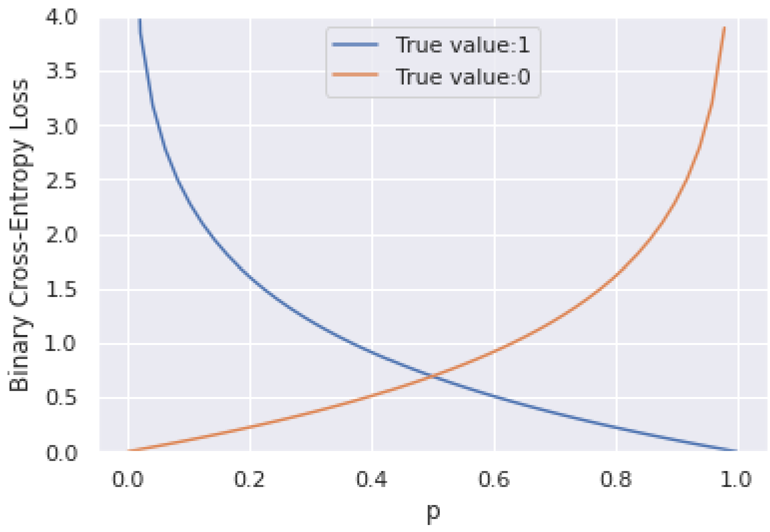
$$P(y|x) = \hat{y}^y (1 - \hat{y})^{1-y}$$

Taking Log

$$\begin{aligned} \log P(y|x) &= \log \hat{y}^y (1 - \hat{y})^{1-y} \\ &= y \log \hat{y} + (1 - y) \log(1 - \hat{y}) \end{aligned}$$

Therefore, we have the cross $-$ entropy loss

$$L(\hat{y}, y) = \log P(y|x) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$



Logistic Function

- Example: $w = (1.2, 0.023, -2.4); b = -205$

Sensor	Value
Temperature	125
Vibration	2550
Pressure	0.85

If the true label $y = 1$

$$L(\hat{y}, y) = -y\log\hat{y} - (1 - y)\log(1 - \hat{y})$$

If the true label $y = 0$

$$L(\hat{y}, y) = -y\log\hat{y} - (1 - y)\log(1 - \hat{y})$$

Learning Logistic Function

- Gradient Descent

$$w^t = \mathrm{w}^{\mathrm{t}} - \underbrace{\alpha}_{\text{Step size}} \underbrace{\frac{d}{dw} L(w, b; x)}_{\text{Improving direction}}$$

$$L(\sigma(z), y) = -y\log\sigma(z) - (1-y)\log(1-\sigma(z)) = -y\log\sigma(wx+b) - (1-y)\log(1-\sigma(wx+b))$$

$$\frac{\partial L(\sigma(z), y)}{\partial w_j} = [\sigma(wx+b) - y] \cdot x_j$$

Learning Logistic Function

function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ
    # where: L is the loss function
    #   f is a function parameterized by θ
    #   x is the set of training inputs $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$
    #   y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, \ldots, y^{(m)}$

$\theta \leftarrow 0$
repeat til done   # see caption
  For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)
    1. Optional (for reporting):   # How are we doing on this tuple?
       Compute $\hat{y}^{(i)} = f(x^{(i)};\theta)$ # What is our estimated output $\hat{y}$?
       Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$) from the true output $y^{(i)}$?
    2. $g \leftarrow \nabla_\theta L(f(x^{(i)};\theta), y^{(i)})$ # How should we move θ to maximize loss?
    3. $\theta \leftarrow \theta - \eta\, g$ # Go the other way instead
return θ

Example

- An equipment showing anomalies with temperature (x_1) and vibration (x_2)

$$x_1 = 3; x_2 = 2 \text{ in the past month, when the equipment failed } (i.e., y = 1)$$

- Want to predict failure | temperature, vibration

Initialization $w_1 = w_2 = b = 0; \;\; \alpha = 0.1$

Model update $w^{t+1} = w^t - \alpha \nabla_w L(\sigma(wx + b; x), y) \qquad b^{t+1} = b^t - \alpha \nabla_b L(\sigma(wx + b; x), y)$

$$\nabla_{w,b} = \begin{bmatrix} \frac{\partial L(\sigma, y)}{\partial w_1} \\ \frac{\partial L(\sigma, y)}{\partial w_2} \\ \frac{\partial L(\sigma, y)}{\partial b} \end{bmatrix}$$

Example

- Update the model

Initialization $w_1^0 = w_2^0 = b^0 = 0; \;\; \alpha = 0.1$

Model update $w^{t+1} = w^t - \alpha \nabla_w L(\sigma(wx + b; x), y) \qquad b^{t+1} = b^t - \alpha \nabla_b L(\sigma(wx + b; x), y)$

$$\nabla_{w,b} = \begin{bmatrix} -1.5 \\ -1.0 \\ -0.5 \end{bmatrix}$$

$$\begin{bmatrix} w_1^1 \\ w_2^1 \\ b^1 \end{bmatrix}$$

Gradient of Cross Entropy Loss

Derivative of log(x)

$$\frac{d}{dx}\ln(x) = \frac{1}{x}$$

Derivative of the logistic function

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1-\sigma(z))$$

Chain Rule of Derivatives

$$\frac{df}{dx} = \frac{du}{dv}\cdot\frac{dv}{dx}$$

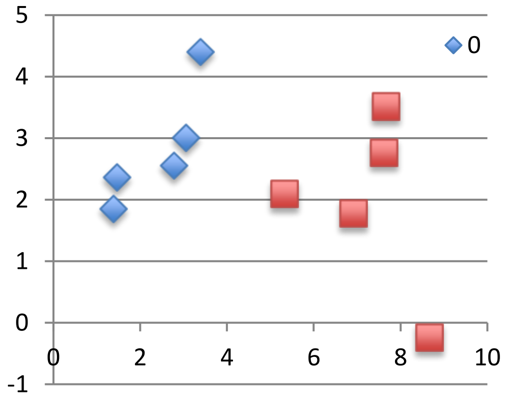
Derivative of Cross-Entropy Loss

$$\begin{aligned}
\frac{\partial L_{\mathrm{CE}}}{\partial w_j} &= \frac{\partial}{\partial w_j} - [y\log\sigma(w\cdot x+b)+(1-y)\log(1-\sigma(w\cdot x+b))] \\
&= -\left[\frac{\partial}{\partial w_j} y\log\sigma(w\cdot x+b) + \frac{\partial}{\partial w_j}(1-y)\log[1-\sigma(w\cdot x+b)]\right] \\
&= -\frac{y}{\sigma(w\cdot x+b)}\frac{\partial}{\partial w_j}\sigma(w\cdot x+b) - \frac{1-y}{1-\sigma(w\cdot x+b)}\frac{\partial}{\partial w_j}1-\sigma(w\cdot x+b) \\
&= -\left[\frac{y}{\sigma(w\cdot x+b)} - \frac{1-y}{1-\sigma(w\cdot x+b)}\right]\frac{\partial}{\partial w_j}\sigma(w\cdot x+b) \\
&= -\left[\frac{y-\sigma(w\cdot x+b)}{\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]}\right]\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]\frac{\partial(w\cdot x+b)}{\partial w_j} \\
&= -\left[\frac{y-\sigma(w\cdot x+b)}{\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]}\right]\sigma(w\cdot x+b)[1-\sigma(w\cdot x+b)]x_j \\
&= -[y-\sigma(w\cdot x+b)]x_j \\
&= [\sigma(w\cdot x+b)-y]x_j
\end{aligned}$$

Logistic Regression by Hand

- Data

X1	X2	Y
2.7810836	2.550537	0
1.46548937	2.36212508	0
3.39656169	4.40029353	0
1.38807019	1.85022032	0
3.06407232	3.00530597	0
7.62753121	2.75926224	1
5.33244125	2.08862678	1
6.92259672	1.77106367	1
8.67541865	-0.2420687	1
7.67375647	3.50856301	1



- Learning rate: 0.1
- Initial model: (w1,w2,b)=(0,0,0)
- Updating Equations:

$$w^{t+1} = w^t - \alpha \nabla_w L(\sigma(wx + b; x), y)$$

$$b^{t+1} = b^t - \alpha \nabla_b L(\sigma(wx + b; x), y)$$

Logistic Regression by Hand

- Iteration #1

$$w_1^0 = w_2^0 = b^0 = 0; \;\; \alpha = 0.1$$

$$\text{Data: } x_1^0 = 2.7810836; \; x_2^0 = 2.550537; \; y^0 = 0$$

$$w^{t+1} = w^t - \alpha \nabla_w L(\sigma(wx + b; x), y) \qquad b^{t+1} = b^t - \alpha \nabla_b L(\sigma(wx + b; x), y)$$

$$\nabla_{w,b} = \begin{bmatrix} (\sigma(wx + b; x) - y)x_1 \\ (\sigma(wx + b; x) - y)x_2 \\ \sigma(wx + b; x) - y \end{bmatrix}$$

$$\begin{bmatrix} w_1^1 \\ w_2^1 \\ b^1 \end{bmatrix}$$

Logistic Regression by Hand

- Iteration #2 $w_1^1 = -0.14;\ w_2^1 = -0.13;\ b^1 = -0.05$

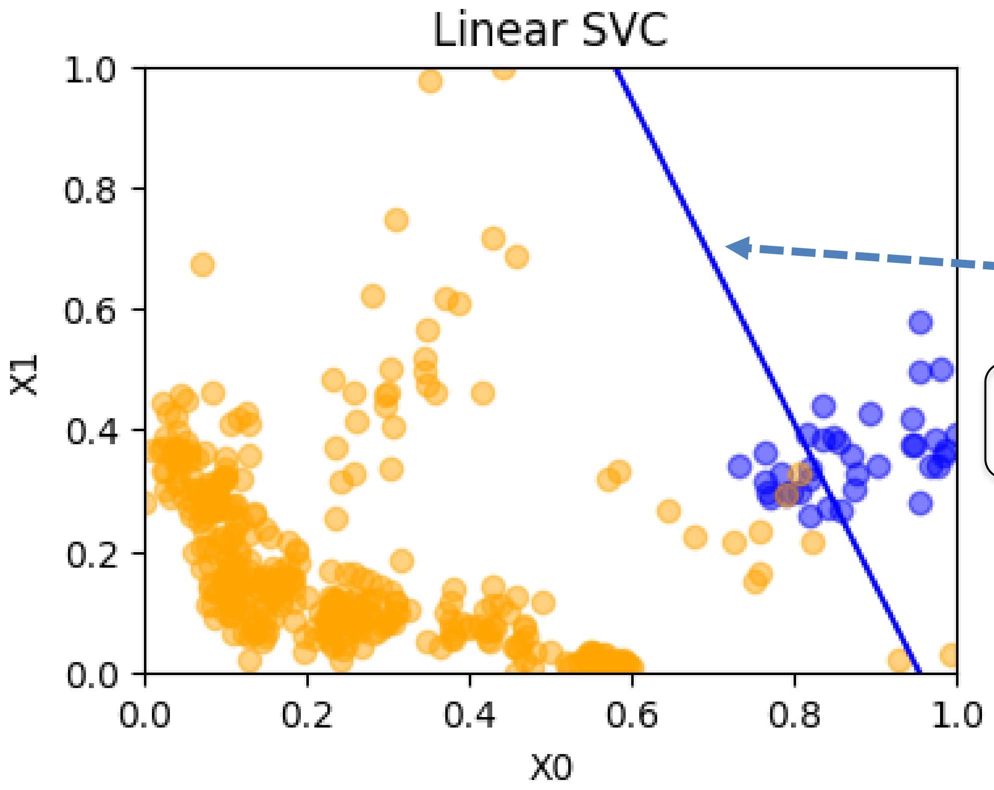
Data: $x_1^0 = 1.4654894;\ x_2^0 = 2.3621251;\ y^0 = 0$

$$\nabla_{w,b} = \begin{bmatrix} (\sigma(wx+b;x)-y)x_1 \\ (\sigma(wx+b;x)-y)x_2 \\ \sigma(wx+b;x)-y \end{bmatrix} = \begin{bmatrix} (\sigma(-0.14\times 1.47 - 0.13\times 2.36 - 0.05) - 0)\times 1.47 \\ (\sigma(-0.14\times 1.47 - 0.13\times 2.36 - 0.05) - 0)\times 2.36 \\ (\sigma(-0.14\times 1.47 - 0.13\times 2.36 - 0.05) - 0) \end{bmatrix}$$

$$\begin{bmatrix} w_1^2 \\ w_2^2 \\ b^2 \end{bmatrix} = \begin{bmatrix} w_1^1 \\ w_2^1 \\ b^1 \end{bmatrix} - \alpha \cdot \begin{bmatrix} 0.53 \\ 0.86 \\ 0.36 \end{bmatrix} = \begin{bmatrix} -0.14 \\ -0.13 \\ -0.05 \end{bmatrix} - 0.1 \times \begin{bmatrix} 0.53 \\ 0.86 \\ 0.36 \end{bmatrix} = \begin{bmatrix} -0.19 \\ -0.21 \\ -0.09 \end{bmatrix}$$

- Iteration #120 $w_1^{120} = 1.14;\ w_2^{120} = -1.54;\ b^{120} = -0.58$

Analyzing: Binary Decision Boundary



$z = Xw + b = 0$

Is this a decision boundary?

$\boldsymbol{\sigma(z = 0) = ?}$

Multiclass Classification

- In this case the shape of the output vector should have $N \times K$ dimensions with one-hot vectors.
- Vectorized:

$$z = \mathrm{W}\mathrm{x} + \mathrm{b}$$

where W is of shape $K \times D$ and b is a K-dimensional vector.

- Two most popular approaches: One-vs-rest, Multinomial

Multiclass Classification

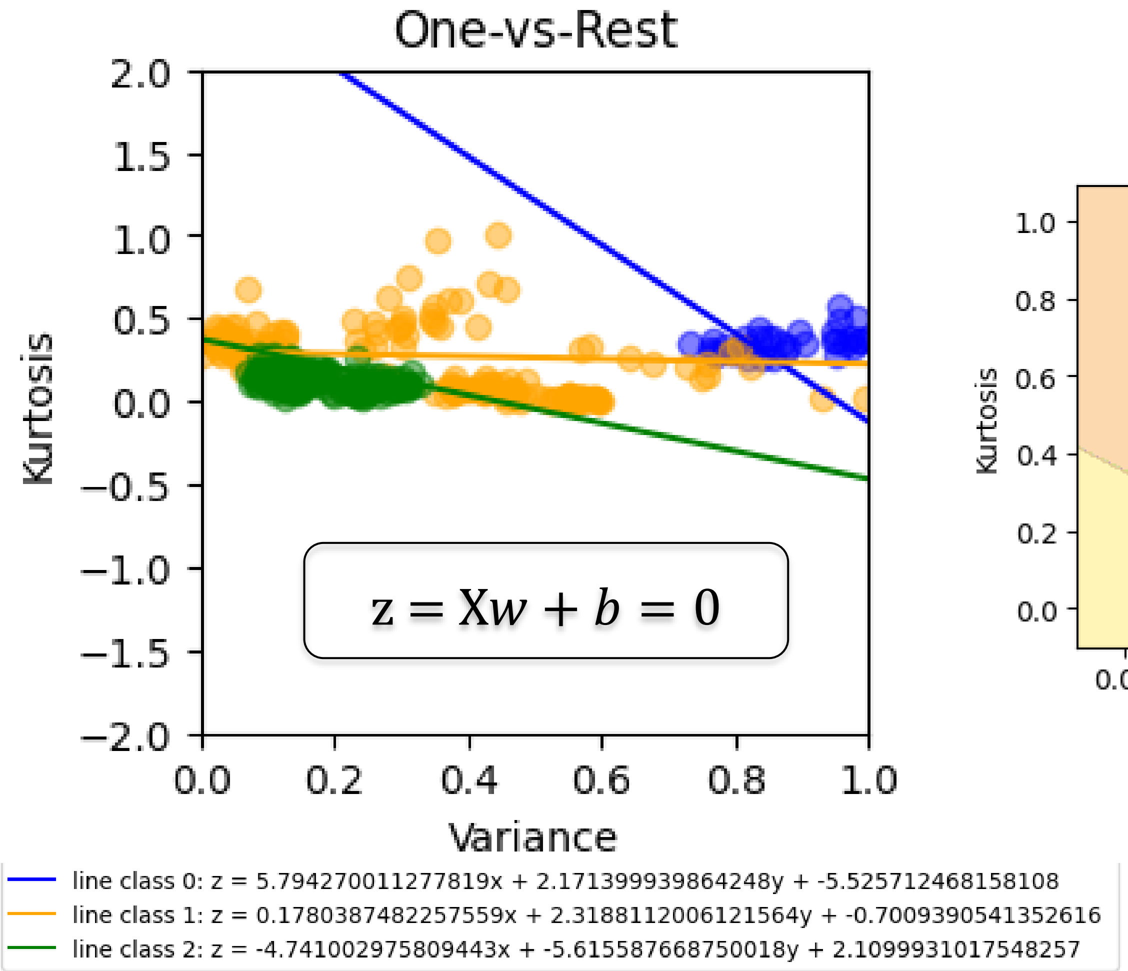
- Activation function in this case will be the softmax function.

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

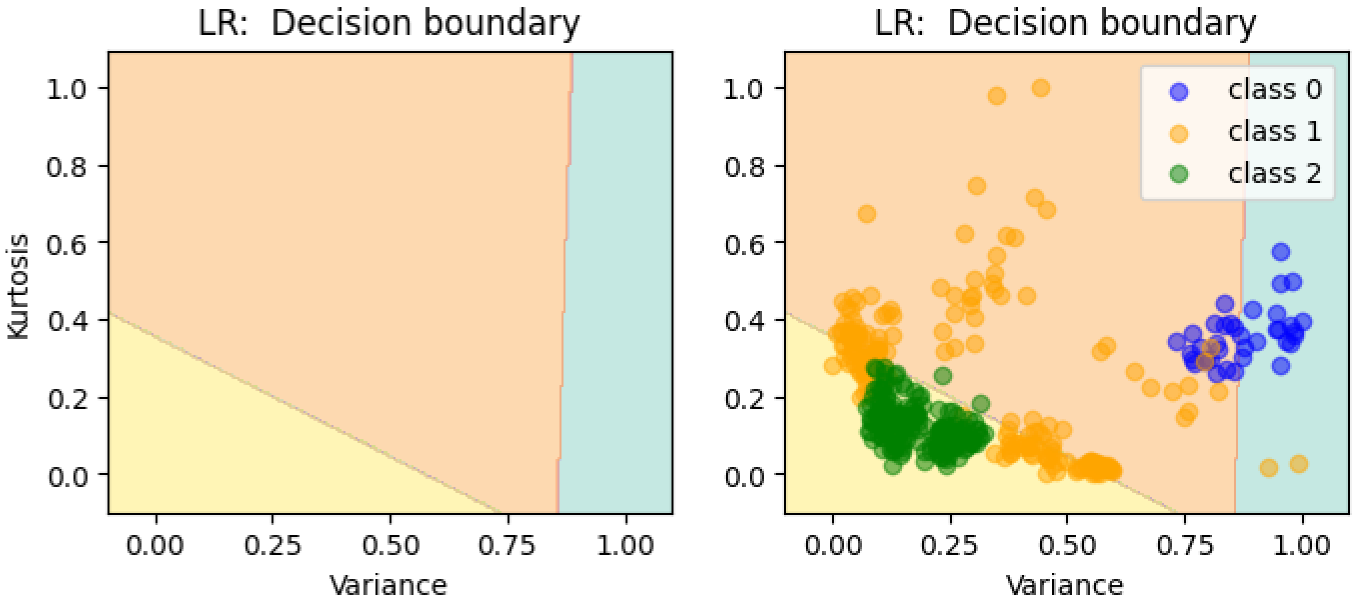
- As the model outputs a vector of class probabilities, the loss function is as follows where the log is applied element wise.

$$\mathcal{L}_{\text{CE}}(\mathbf{y}, \mathbf{t}) = -\sum_{k=1}^{K} t_k \log y_k$$

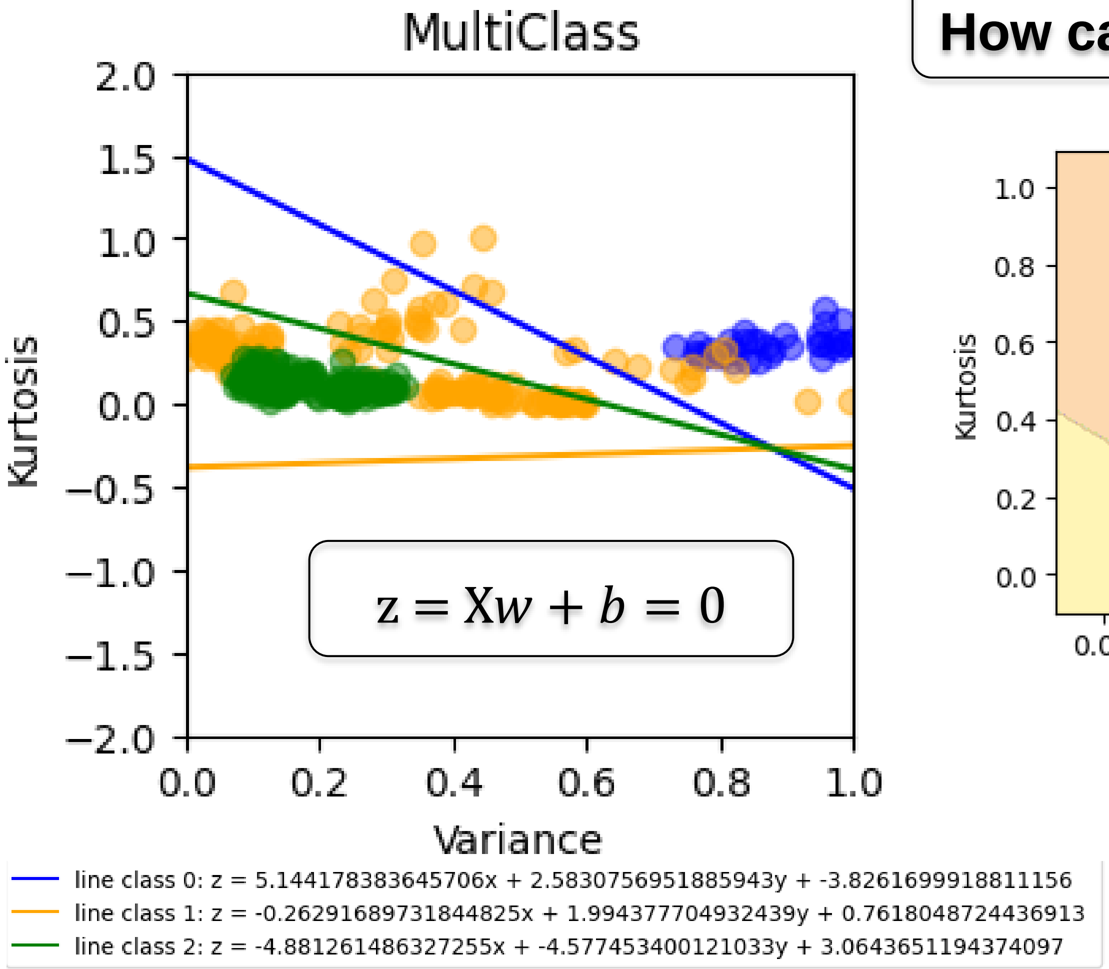
One-vs-Rest Decision Boundary



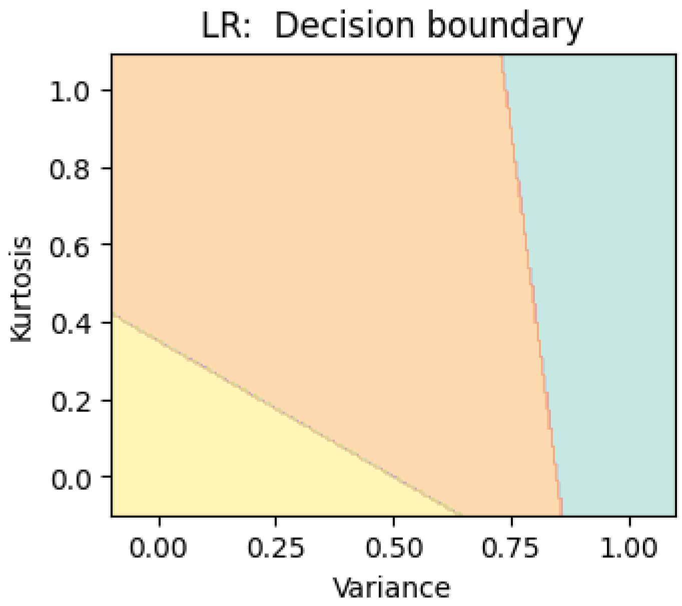
How is the decision boundary defined?

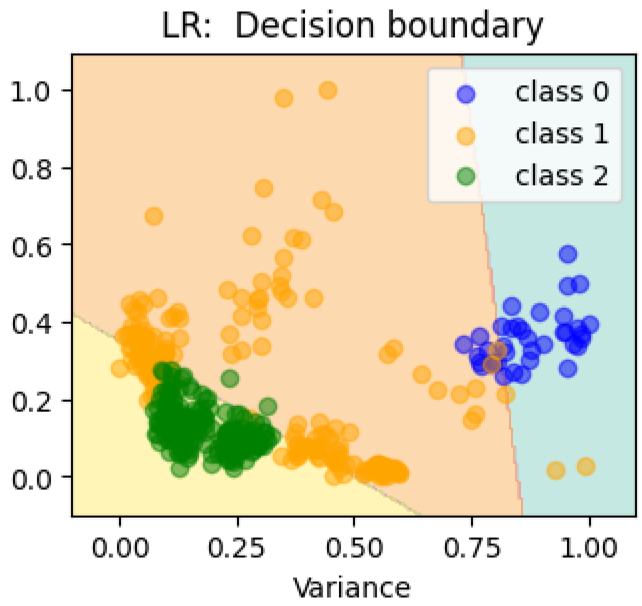


Multinomial Decision Boundary



How can you interpret these boundaries? (hint: log-odds)





Summary: Linear Models

- Regression with Linear Models:
  - Optimization: Direct Solution, Gradient Descent
  - Cost Function and Regularization
- Classification with Linear Models:
  - Activation Functions: Logistic and Softmax
  - No Direct Solution

Naïve Bayes Method

Naive Bayes Method

- Probabilistic ML method based on the Bayes Theorem
- Prob {hypothesis y is true given evidence X}

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

- Given

$$X = (x_1, x_2, x_3, ..., x_n)$$

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

$$\propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

#	Outlook	Temp	Humidity	Windy	Mtnc Op
1	Rainy	Hot	High	False	No
2	Rainy	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Sunny	Mild	High	False	Yes
5	Sunny	Cool	Normal	False	Yes
6	Sunny	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Rainy	Mild	High	False	No
9	Rainy	Cool	Normal	False	Yes
10	Sunny	Mild	Normal	False	Yes
:	:	:	:	:	:
:	:	:	:	:	:

Naive Bayes Method

#	Outlook	Temp	Humidity	Windy	Mtnc Op
1	Rainy	Hot	High	False	No
2	Rainy	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Sunny	Mild	High	False	Yes
5	Sunny	Cool	Normal	False	Yes
6	Sunny	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Rainy	Mild	High	False	No
9	Rainy	Cool	Normal	False	Yes
10	Sunny	Mild	Normal	False	Yes
11	Rainy	Mild	Normal	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Sunny	Mild	High	True	No

y	(x_1, x_2, x_3, x_4)	$P(y)\prod_{i=1}^{n} P(x_i \mid y)$
Y	Rain, Hot, Humid, False	0.64*(0.22*0.22*0.33*0.67) = 0.007
N		0.36*(0.60*0.40*0.80*0.40) = **0.027**

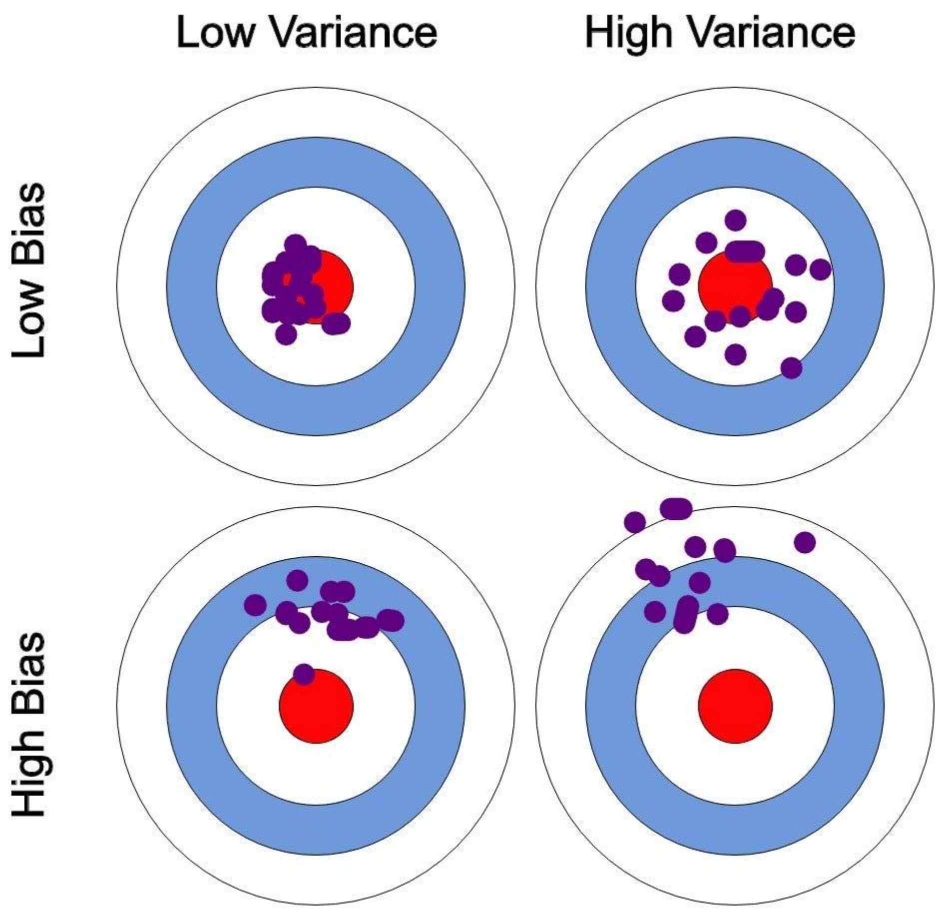
Rainy, hot, humid & not windy → No maintenance operation

Ensemble Methods

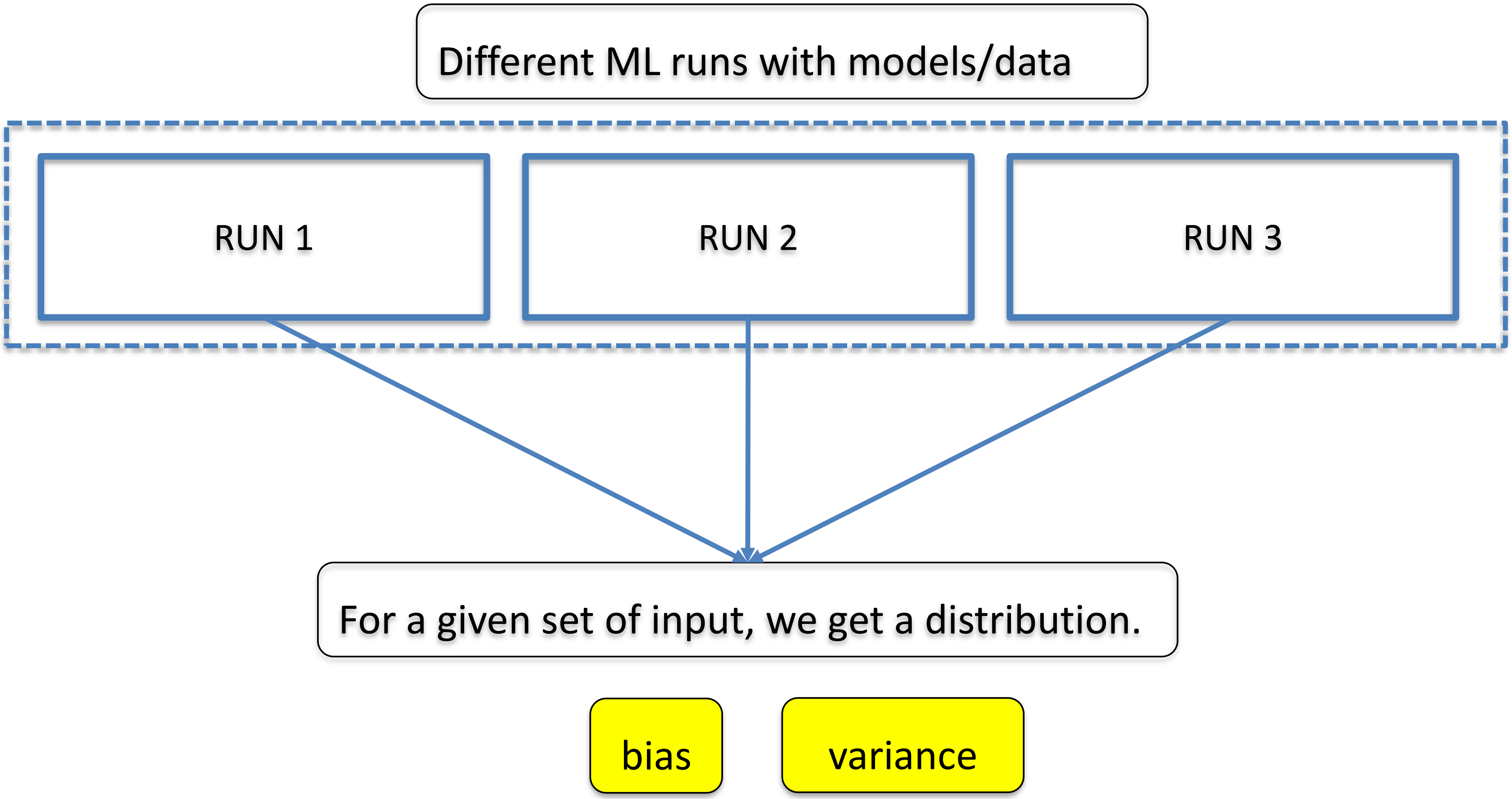
Overview

- Ensemble is a method where predictions from different models are combined to yield the final output.
- There are different ways in which this can be implemented:
  - Different types of machine learning models are trained on the same training set.
  - Same model type with a similar training set but different parameters.
  - Same model but trained on different subsets of the training set.
- Two major types: Bagging and Boosting

Bias-Variance in Machine Learning



Bias-Variance Setup



Bagging Method

- Reduces Variance
- Solves Overfitting

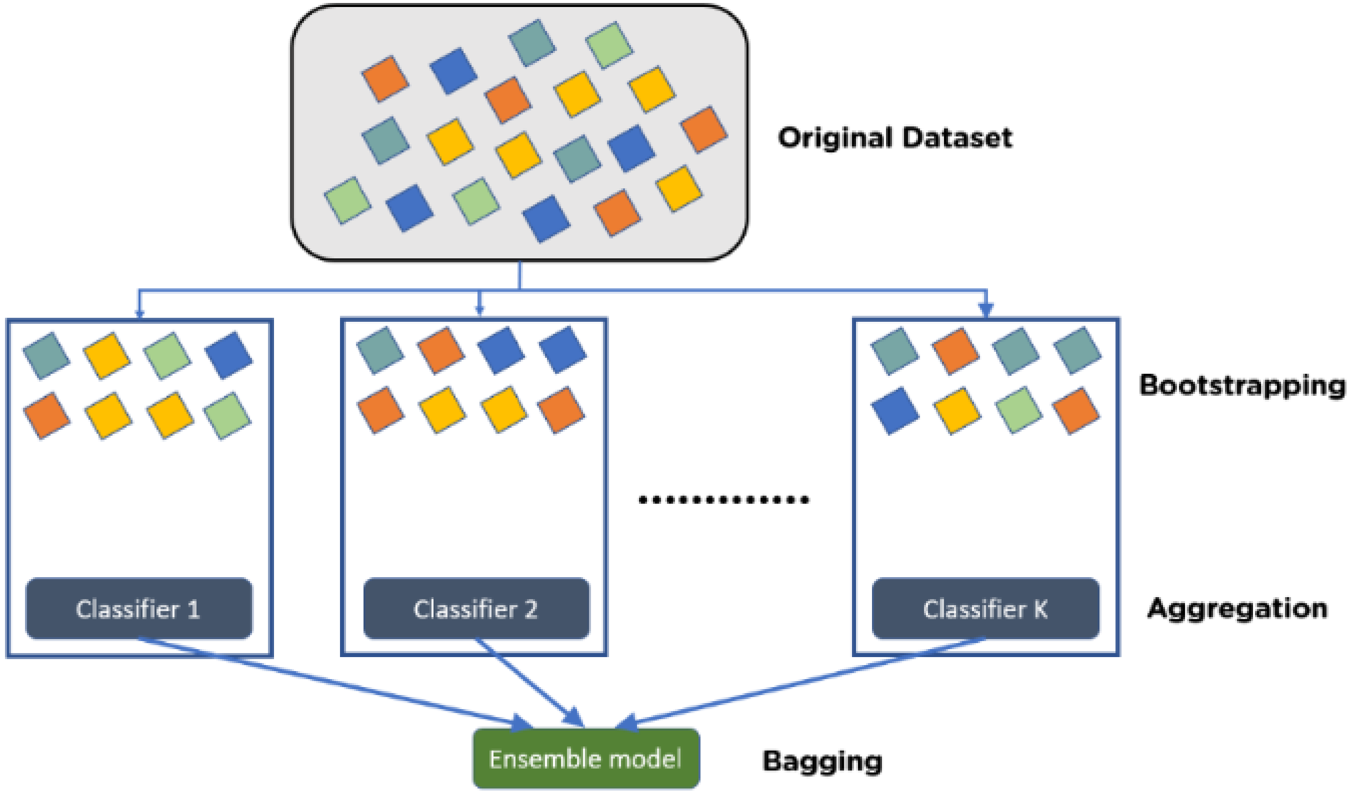


Image Source

Boosting Method

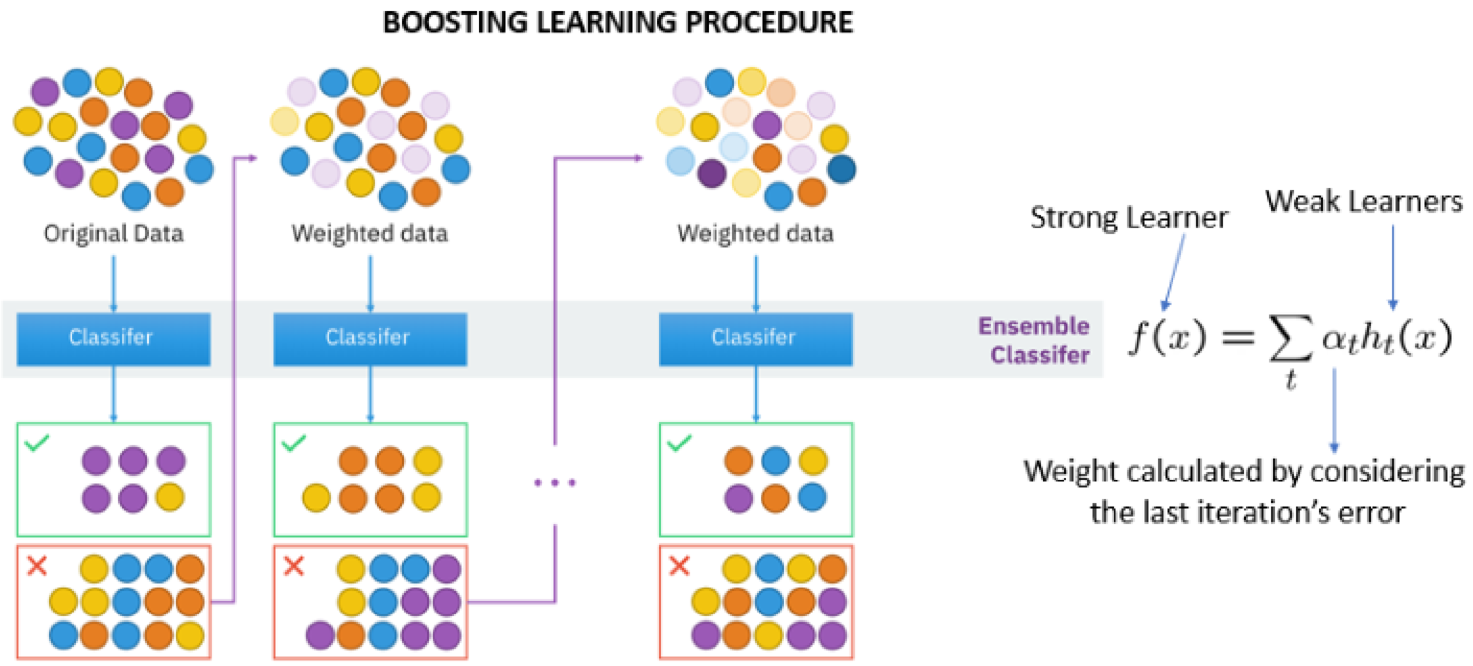


Image Source

- Reduces Bias
- Can Overfit

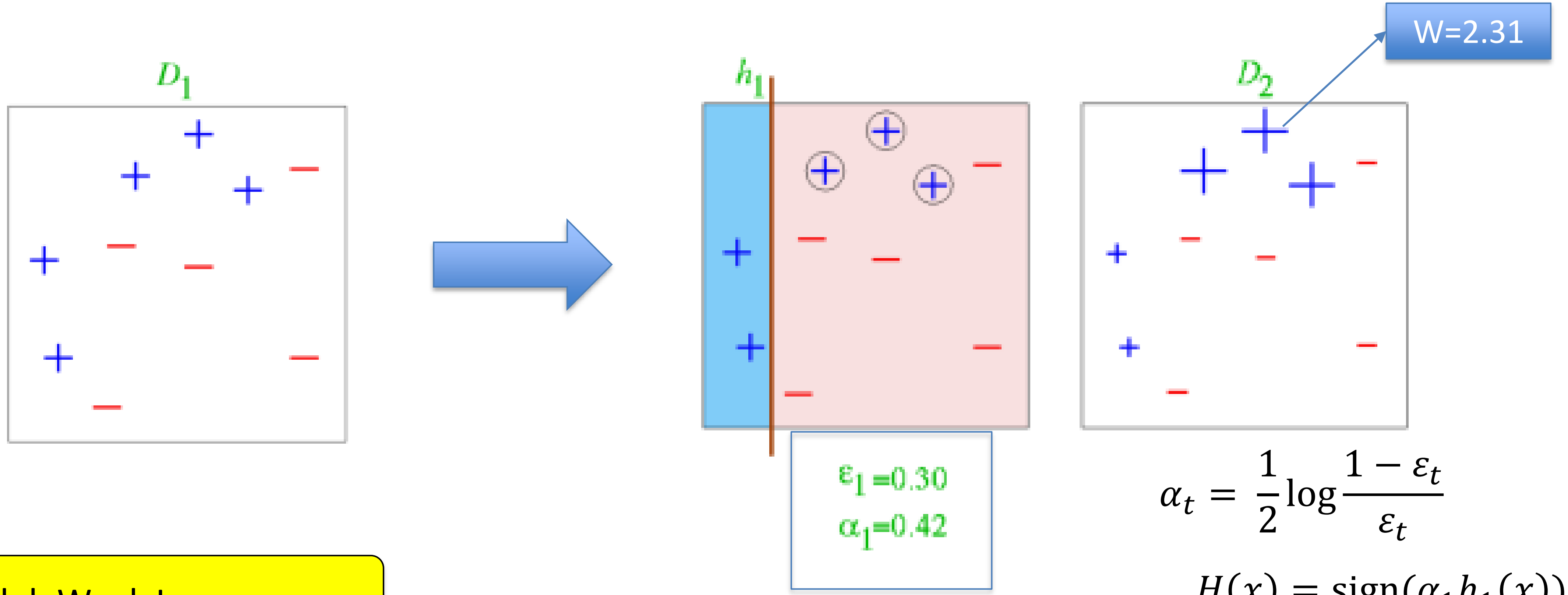
AdaBoost Algorithm

for i from 1 to N, $w_i^{(1)} = 1$
for $m = 1$ to M **do**
  Fit weak classifier m to minimize the objective function:
    $\epsilon_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I(f_m(\mathbf{x}_i) \neq y_i)}{\sum_i w_i^{(m)}}$
  where $I(f_m(\mathbf{x}_i) \neq y_i) = 1$ if $f_m(\mathbf{x}_i) \neq y_i$ and 0 otherwise
  $\alpha_m = \ln \frac{1-\epsilon_m}{\epsilon_m}$
  for all i **do**
    $w_i^{(m+1)} = w_i^{(m)} e^{\alpha_m I(f_m(\mathbf{x}_i) \neq y_i)}$
  end for
end for

$$g(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m f_m(\mathbf{x})\right)$$

Source

AdaBoost Example



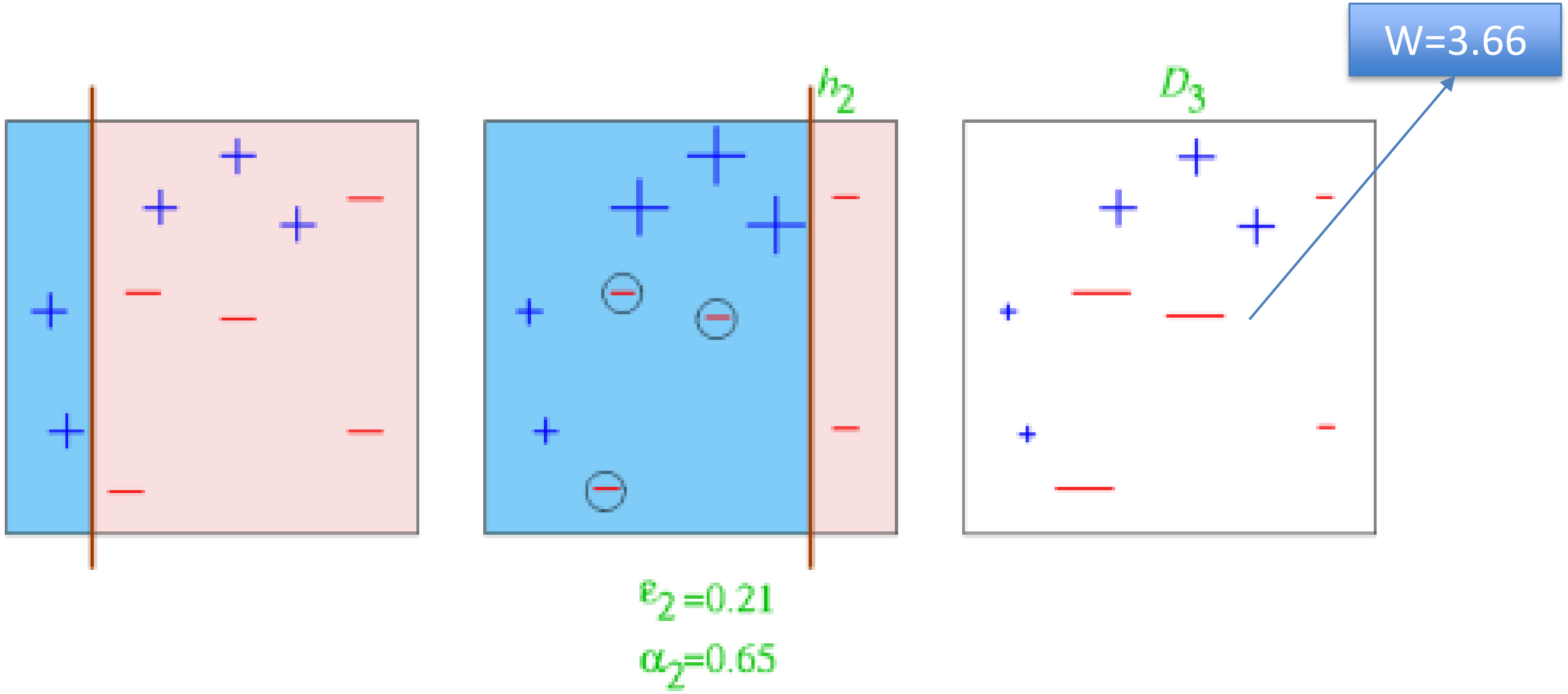
$$\alpha_t = \frac{1}{2}\log\frac{1-\varepsilon_t}{\varepsilon_t}$$

$$H(x) = \text{sign}(\alpha_1 h_1(x))$$

Model: Weak Learner

Slide Reference

AdaBoost Example

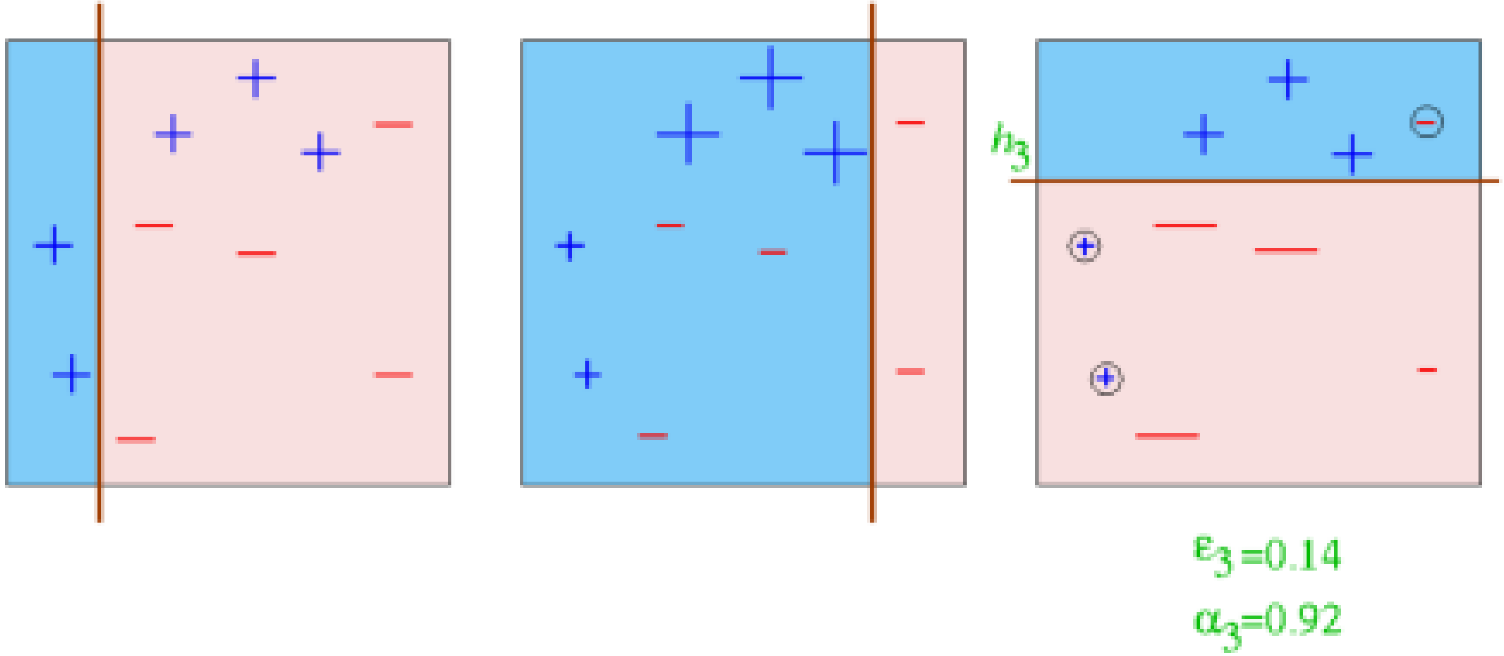


Model: Weak Learner

$$H(x) = \text{sign}\big(\alpha_1 h_1(x) + \alpha_2 h_2(x)\big)$$

Slide Reference

AdaBoost Example



Model: Weak Learner

$$H(x) = \text{sign}\big(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)\big)$$

Slide Reference

UNIVERSITY OF TORONTO | Engineering

Thank you.

cmore@mie.utoronto.ca
http://cmore.mie.utoronto.ca