TEXAS A&M UNIVERSITY
Engineering

ECEN 758 Data Mining and Analysis: Lecture 12, Bayesian and Nearest Neighbor Classification

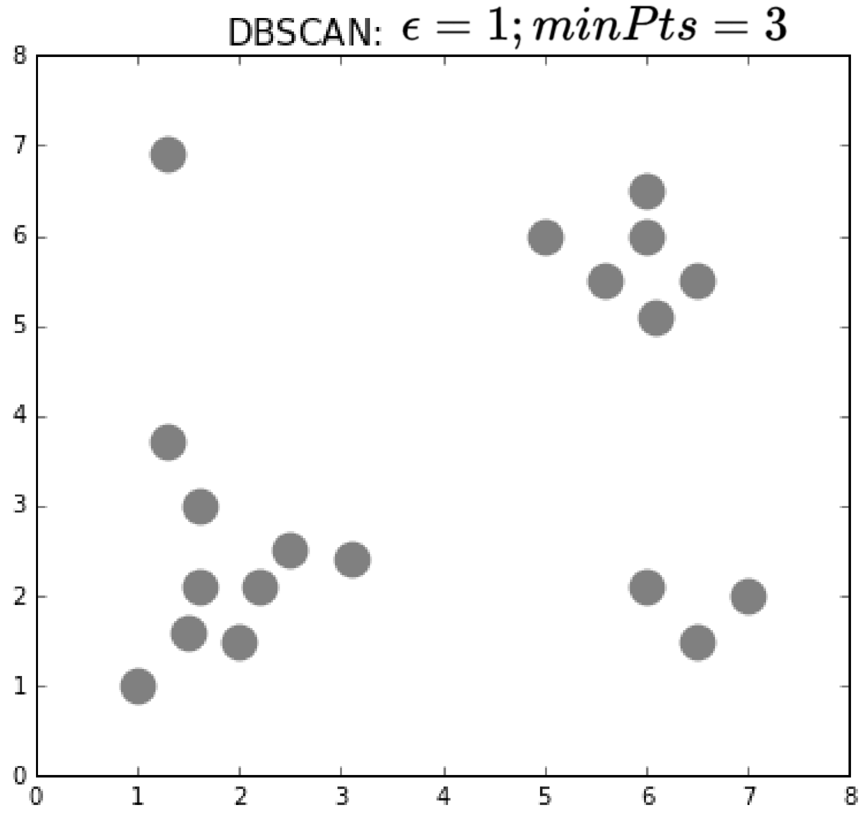
Joshua Peeples, Ph.D.
Assistant Professor
Department of Electrical and Computer Engineering

Announcements

- Assignment #3 will be released next Wednesday (10/09)
- No class next Monday (10/07): Fall Break
- Guest lecture next Wednesday (10/09)
  - +1 on Midterm exam for attendance (must sign-in to attendance sheet)
  - +3 for one paragraph (5 – 7 sentences) summary of presentation (must attend lecture)
  - Section 700: Watch recording and submit 1 paragraph (+1) or 2 paragraphs (+3)
- Exam I in two weeks on Monday, 10/14

Last Lecture

- Density-based Clustering



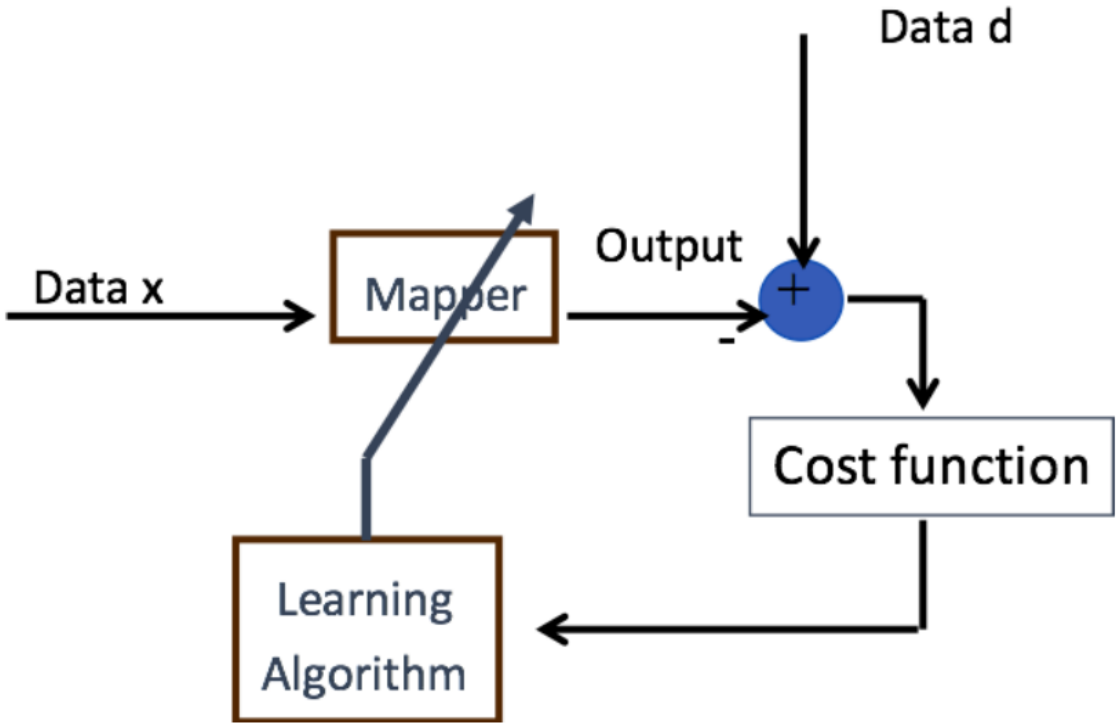
Today

TEXAS A&M UNIVERSITY Engineering

- Introduction to Classification I
  - Definition
  - Model types
  - Hyperparameters vs Parameters
- Bayesian and Nearest Neighbor Classification
- Reading: ZM Chapter 18

Machine Learning Model

- In machine learning the model is derived from the data (observations)
- As a learning machine, the model can be modified over time, with additional data (observations), with the goal of improving outcomes



Many Sub-areas in Machine Learning

TEXAS A&M UNIVERSITY Engineering

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Semi-supervised Learning
- Self-supervised Learning
- Multiple Instance Learning
- Active Learning
- Transfer Learning
- ….

Types of Learning

- Supervised learning
  - We “coach” the computer
  - Uses knowledge already learned
- Unsupervised learning
  - “We’re free!!”

Supervised Learning: Classification

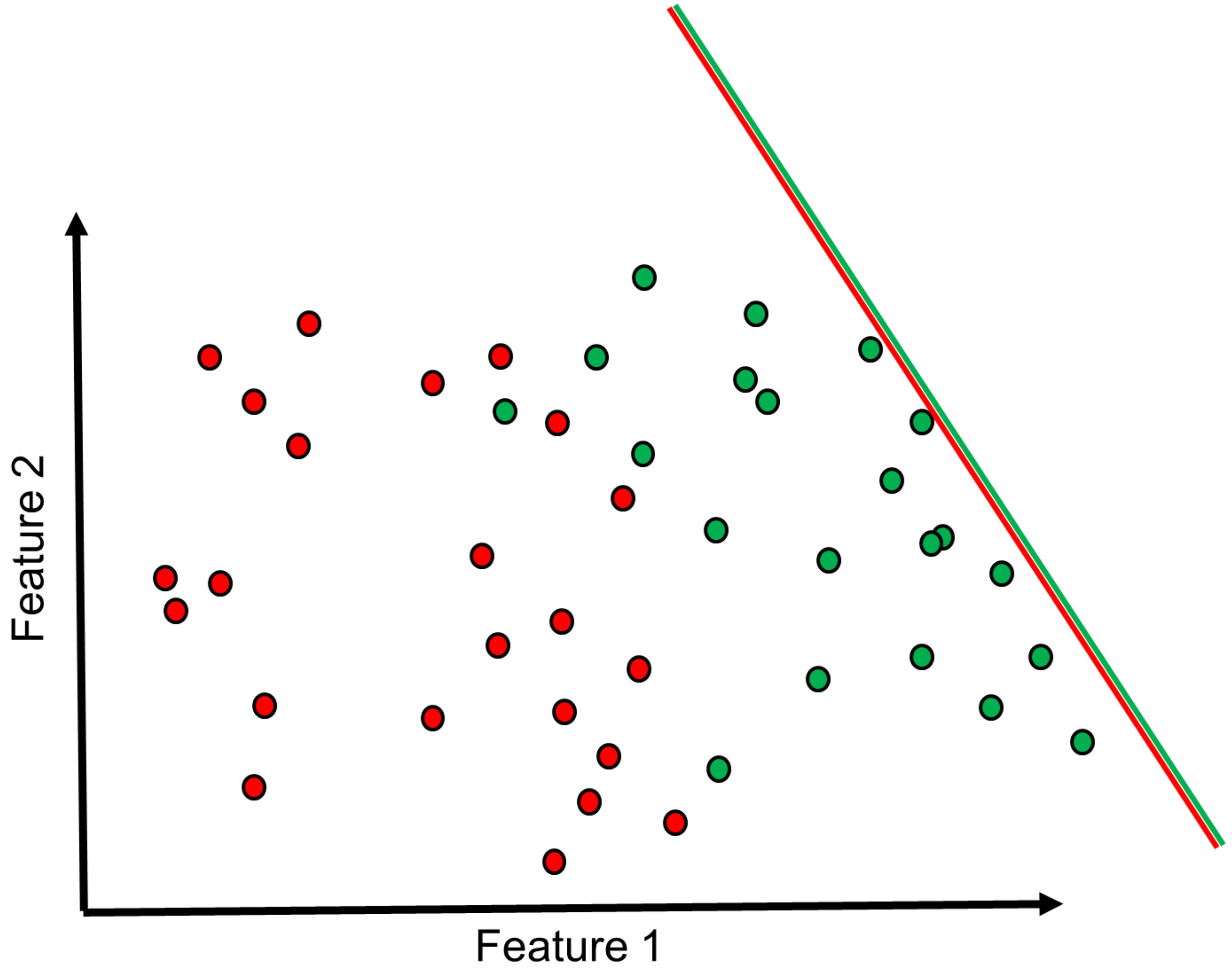
Supervised Learning

Learning from experience

0: Macaw 1: Conure

Classification Example

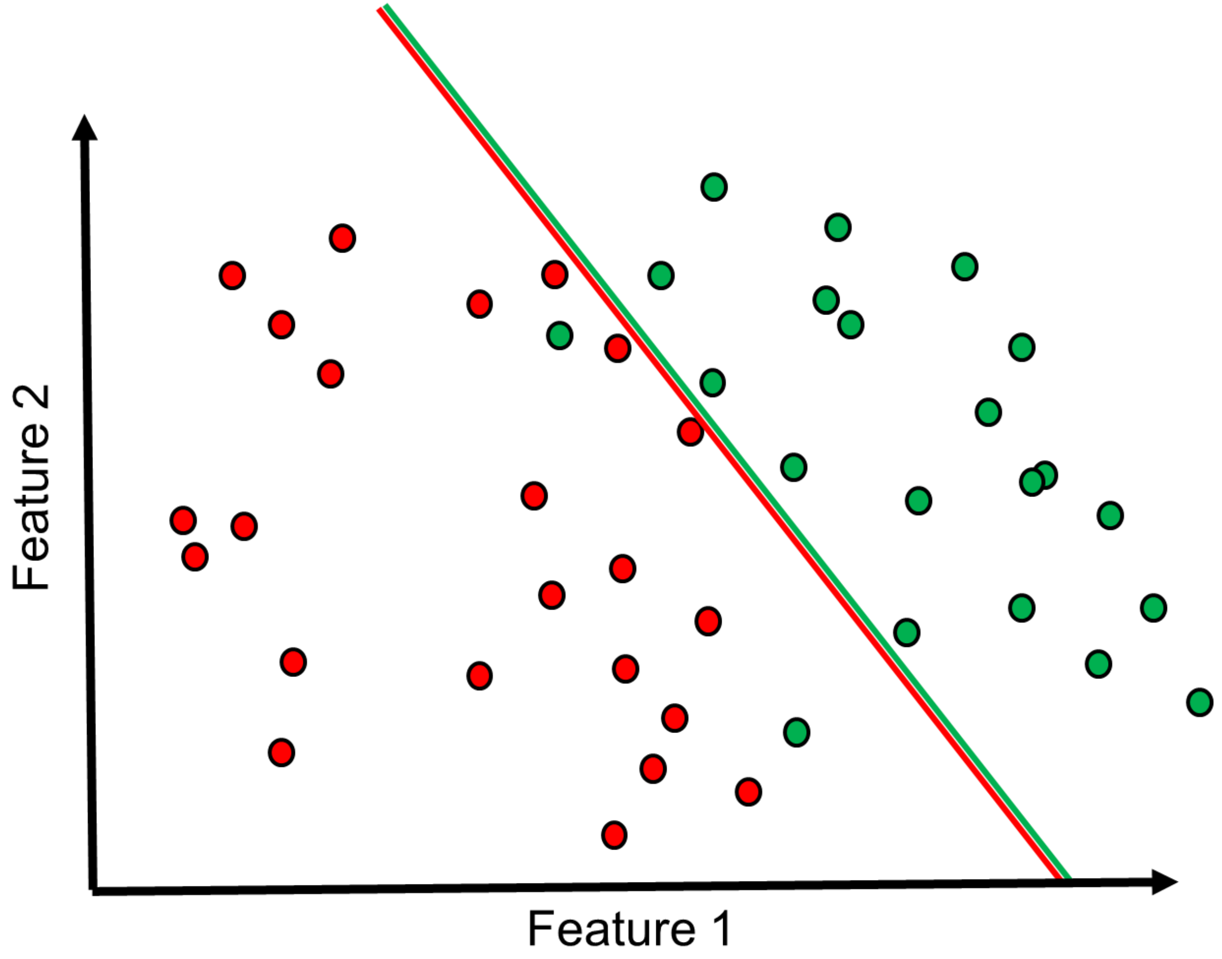
- Given a set of labeled (training) instances, learn a model
  - E.g., Find parameters for linear classifier
- Accurately predict new (test) samples



Classification Example

TEXAS A&M UNIVERSITY Engineering

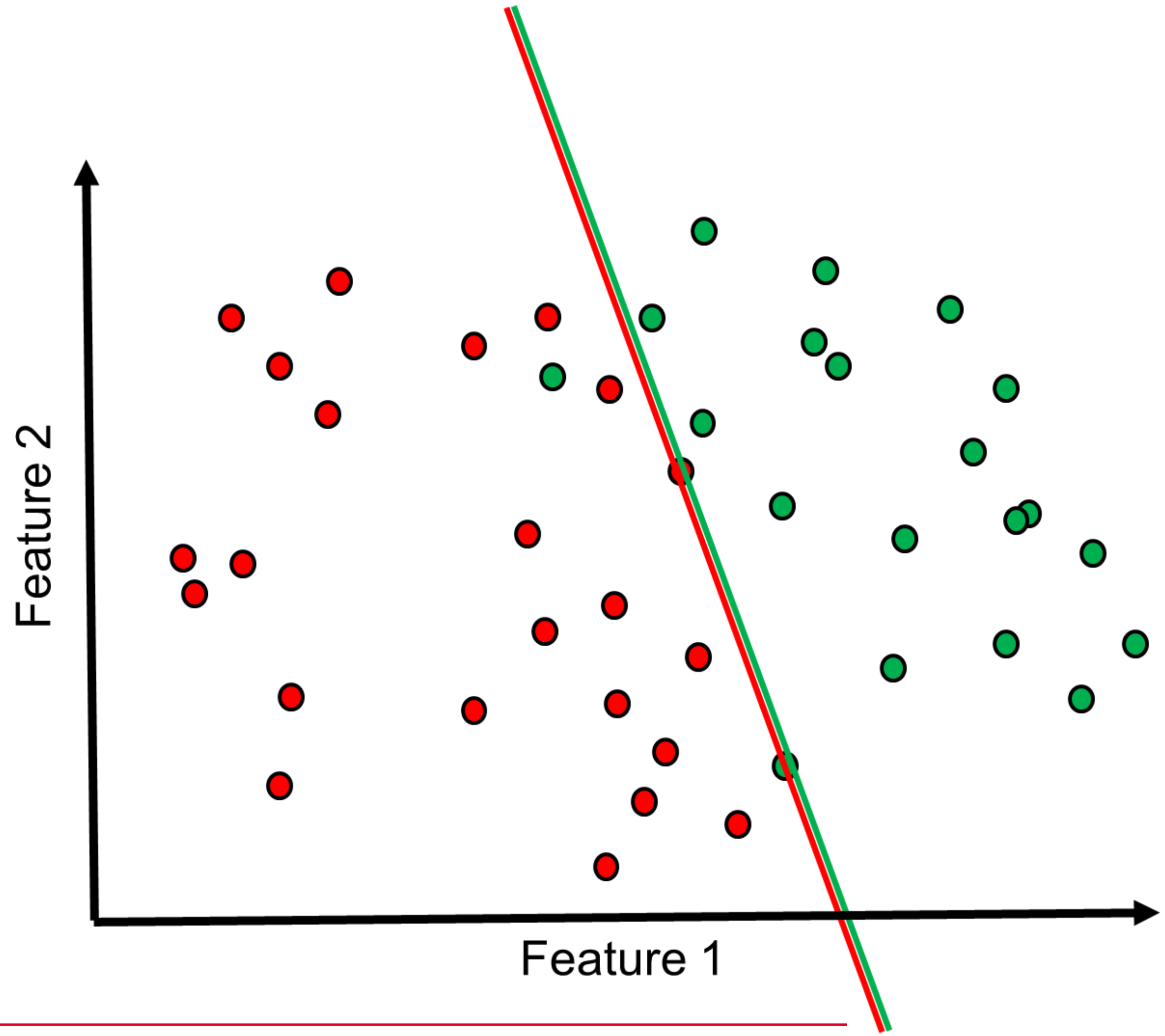
- Given a set of labeled (training) instances, learn a model
  - E.g., Find parameters for linear classifier
- Accurately predict new (test) samples



Classification Example

TEXAS A&M UNIVERSITY Engineering

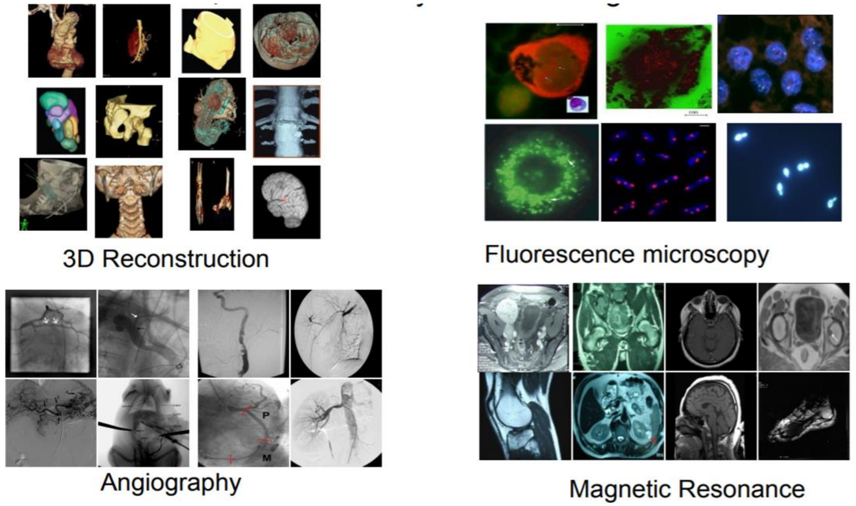
- Given a set of labeled (training) instances, learn a model
  - E.g., Find parameters for linear classifier
- Accurately predict new (test) samples



Classification Overview

Classification Tasks

- Classification: given features **X**, predict label (class) y
- Examples:
  - Communication symbol recognition (input signal w/source & channel induced noise, classes: valid communication symbols)
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Character recognition (input: handwritten characters, classes: {a..z})
  - Fraud detection (input: account activity, classes: fraud / no fraud)
  - … many more



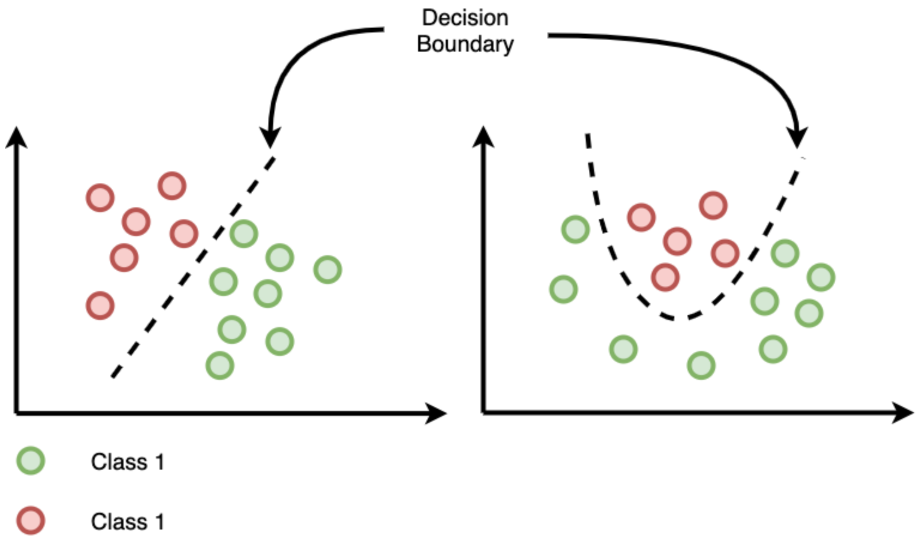
Src: Nowka 2015, IBM CLEF 2013

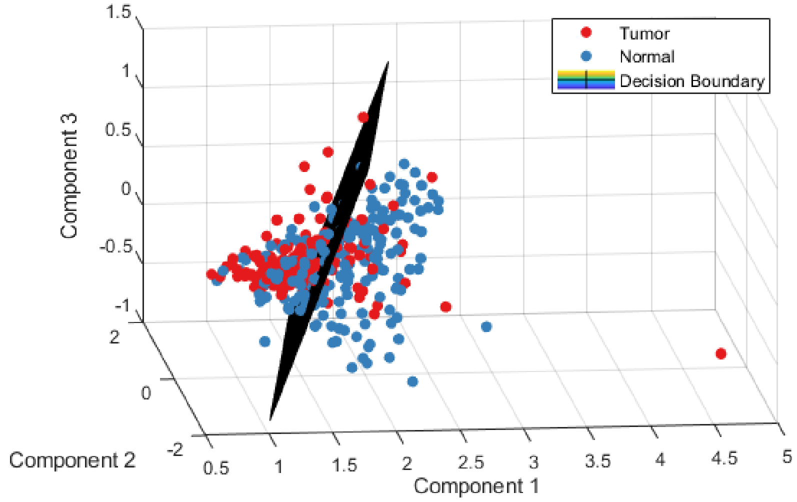
Image from: Madhu Ramiah- Medium

Decision Boundaries

TEXAS A&M UNIVERSITY Engineering

- Distinguish between classes
- In higher dimensions (>2), learning hyperplanes





Images from: Anuradha Wickramarachchi, Logistic Regression and Decision Boundary; Ming-Jer Jeng, Raman Spectroscopy Analysis for Optical Diagnosis of Oral Cancer Detection

Types of Classification

- **Binary**
- **Multi-class**
- Multi-label (binary)
- Multi-class, multi-output

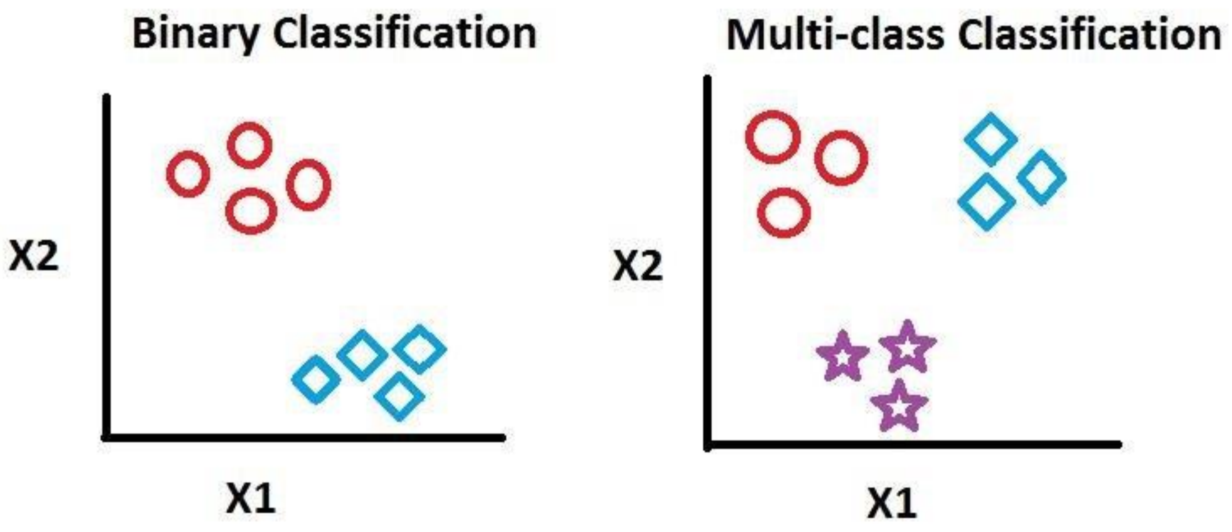
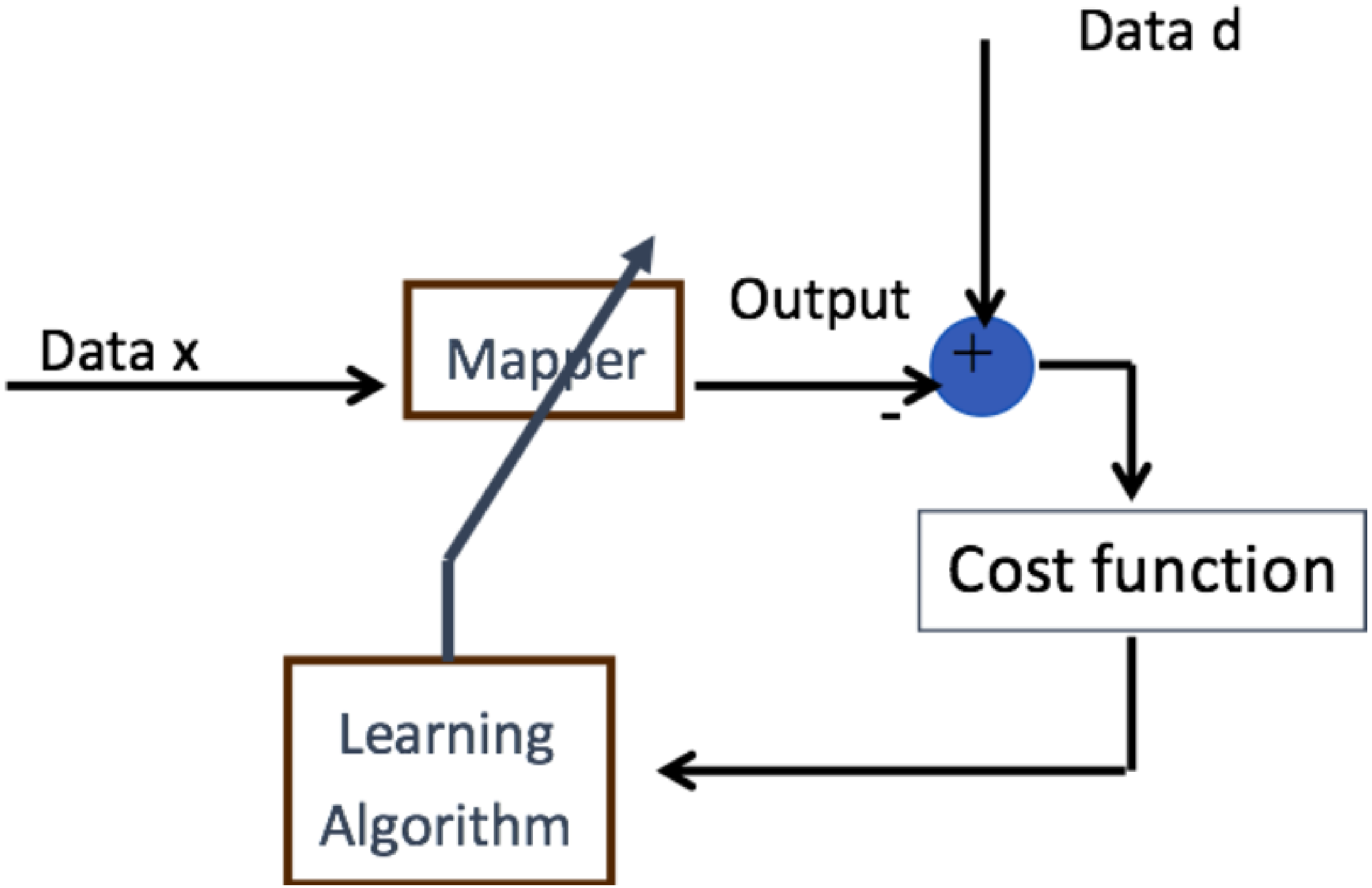


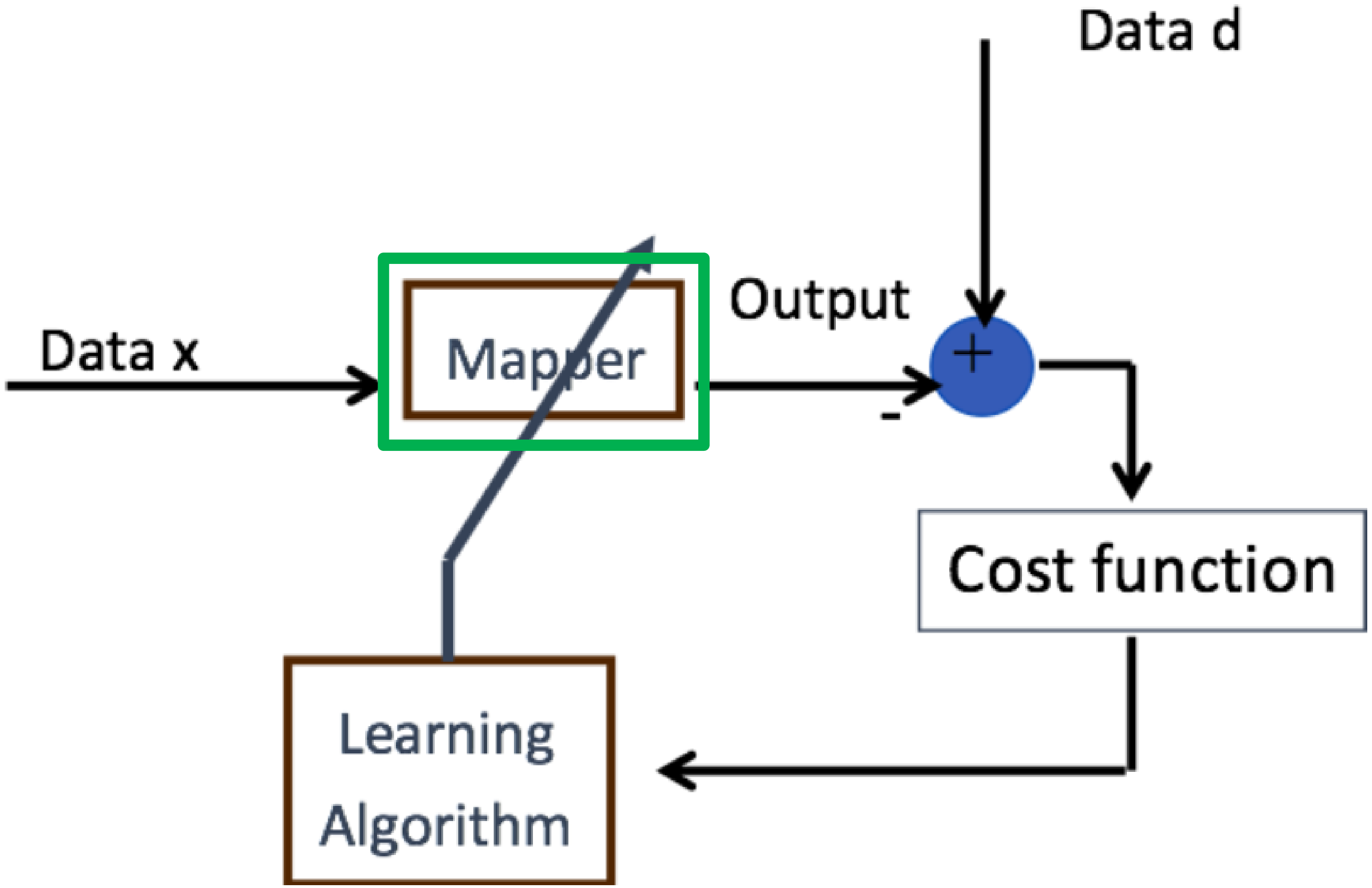
Image from: Emerson Nithiyaraj, FPGA Based Real Time Classification With Support Vector Machine

Model Types

TEXAS A&M UNIVERSITY Engineering



Model Types



Data d
Data x
Mapper
Output
+
-
Cost function
Learning
Algorithm

Parameters and Hyperparameters

- Parameters ($\boldsymbol{\theta}$) – (e.g. weights, means, covariances) are derived/modified (learned!) in the process of training the model
- Hyperparameters – (e.g. k in K-means) are set outside of the training/fit cycle and are used to direct characteristics of the resulting model

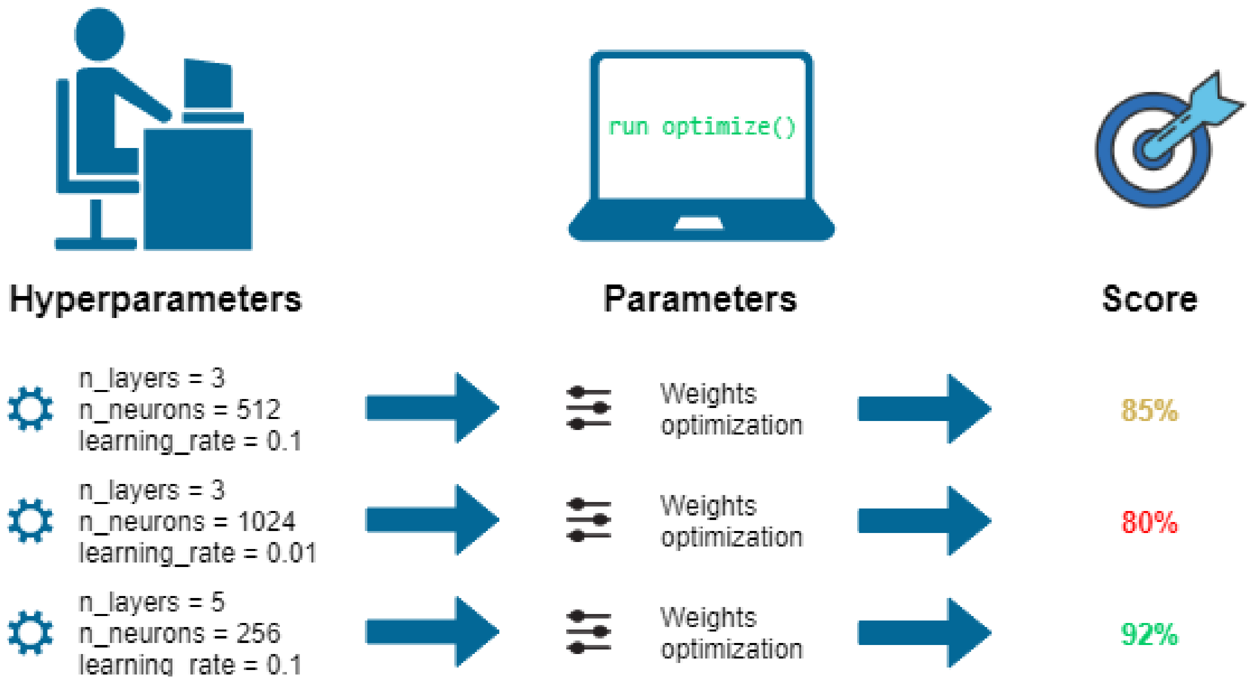


Image from: What is a hyperparameter? DeepAI

Discriminative vs Generative

- Discriminative
  - Focus on the decision boundary between classes
- Generative
  - Focus on distribution of classes

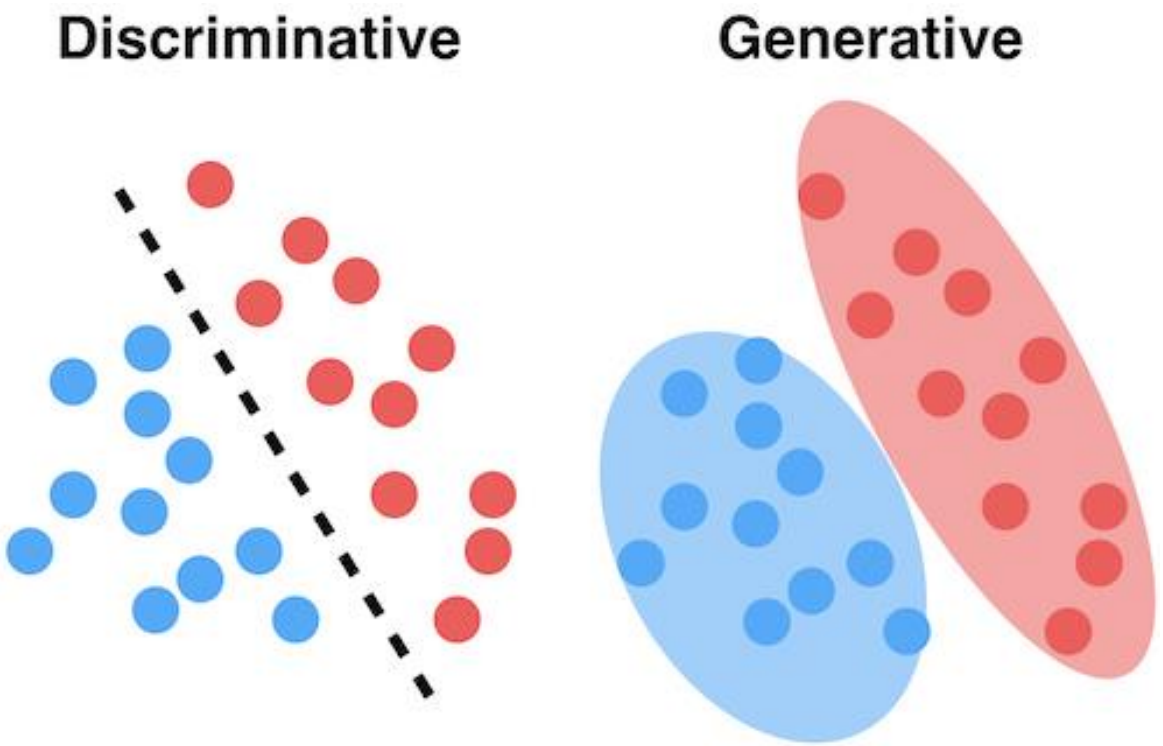


Image from: betterprogramming.pub

Parametric vs Non-parametric

- Parametric
  - Use known functional form
  - Makes assumptions of data
- Non-parametric
  - Free to learn any functional form
  - No strong assumptions of mapping function

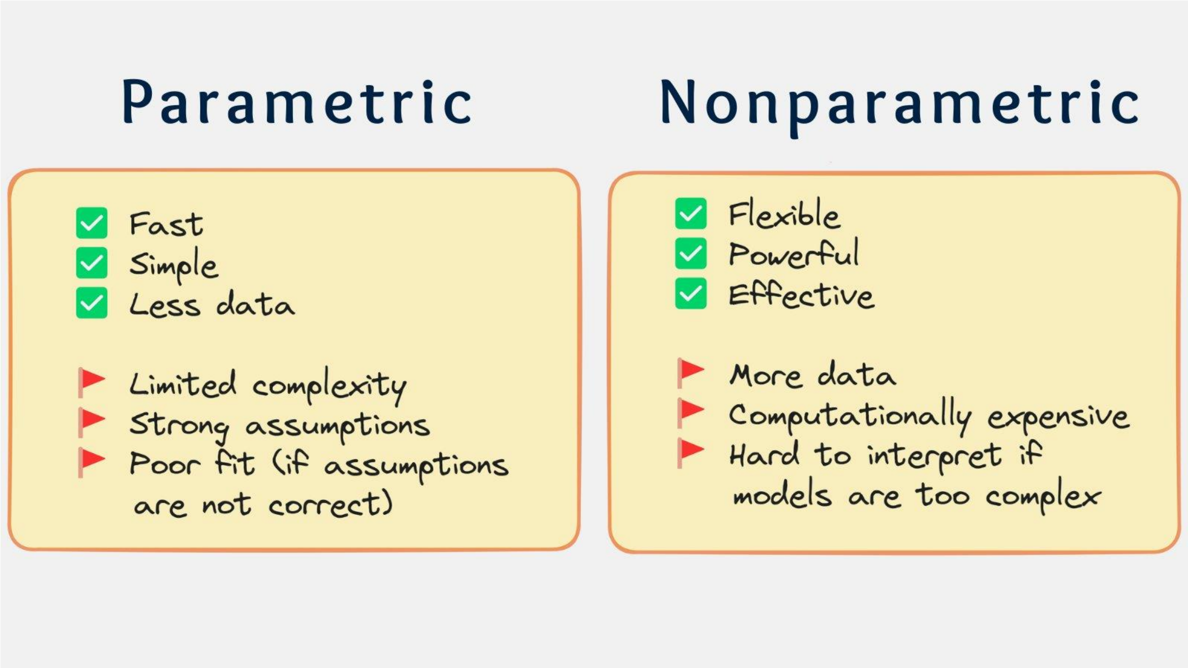
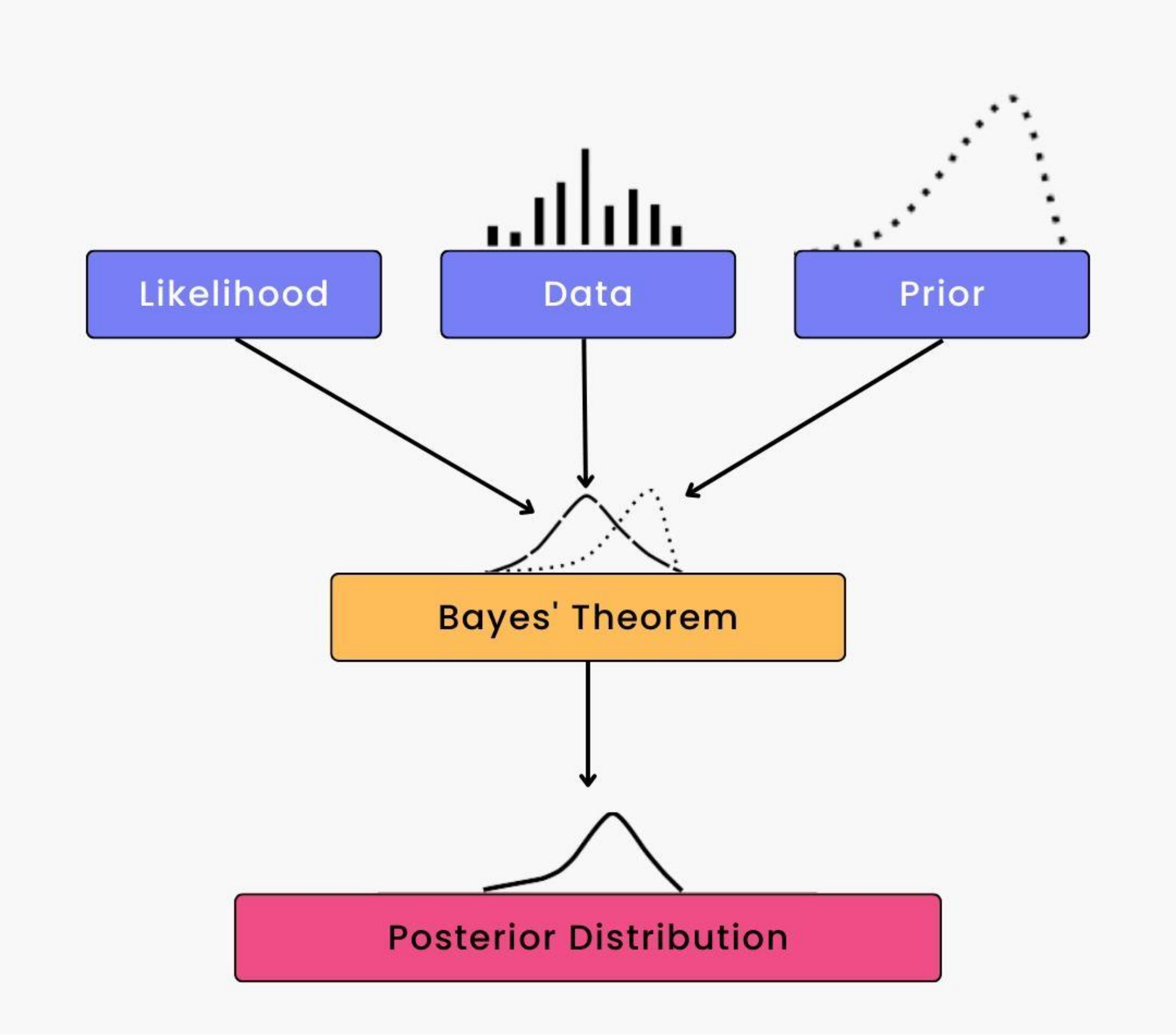


Image from: Levi, X.

Bayes Classifier

Bayes' Theorem



"Posterior" "Likelihood" "Prior"

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

"Evidence"

Image from: Turing, An Introduction to Naive Bayes Algorithm for Beginners.

Bayes Classifier

- Given training dataset **D**
  - n points $\mathbf{x}_i$ in d-dimensions
  - Each point has class label y_i

$$\mathbf{D} = \begin{pmatrix} X_1 & X_2 & \cdots & X_d \\ \hline x_{11} & x_{12} & \cdots & x_{1d} \\ x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nd} \end{pmatrix} \quad y_i \in \{c_1, c_2, \ldots, c_k\}$$

- Estimate posterior probability for each class c_i
- Predicted class has highest probability ($\hat{y}$)

$$\hat{y} = \arg\max_{c_i} \{P(c_i|\mathbf{x})\}$$

$$\boxed{P(c_i|\mathbf{x}) = \frac{P(\mathbf{x}|c_i) \cdot P(c_i)}{P(\mathbf{x})}}$$

Bayes Classifier

- Given training dataset **D**
  - n points $\mathbf{x}_i$ in d-dimensions
  - Each point has class label y_i
- P(x) is fixed for given point

$$P(c_i|\mathbf{x}) = \frac{P(\mathbf{x}|c_i)\cdot P(c_i)}{P(\mathbf{x})}$$

$$P(\mathbf{x}) = \sum_{j=1}^{k} P(\mathbf{x}|c_j)\cdot P(c_j)$$

$$\arg\max_{c_i}\{P(\mathbf{x}|c_i)P(c_i)\}$$

Bayes Classifier: Parameter Estimation

Bayes Classifier: Prior

- Can be given prior
  - E.g., (fair) coin toss
- Can find by counting observations
  - Number point data points in class (n_i) divided by total number of data points (n)

$$\boldsymbol{D}_i = \{\boldsymbol{x}_j \in \boldsymbol{D} \mid \boldsymbol{x}_j \text{ has class } y_j = c_i\}$$

$$\boxed{\hat{P}(c_i) = \frac{n_i}{n}}$$

Bayes Classifier: Likelihood

- Need to estimate joint probability of **x** across all dimensions
- Assume each class is normally distributed

$$P(\mathbf{x}|c_i) = P(\mathbf{x} = (x_1, x_2, \ldots, x_d)|c_i)$$

$$\hat{f}_i(\mathbf{x}) = \hat{f}(\mathbf{x}|\hat{\boldsymbol{\mu}}_i, \widehat{\Sigma}_i) = \frac{1}{(\sqrt{2\pi})^d \sqrt{|\widehat{\Sigma}_i|}} \exp\left\{ -\frac{(\mathbf{x} - \hat{\boldsymbol{\mu}}_i)^T \widehat{\Sigma}_i^{-1} (\mathbf{x} - \hat{\boldsymbol{\mu}}_i)}{2} \right\}$$

Bayes Classifier: Likelihood

- Find posterior probability, $P(c_i|\mathbf{x})$
- Predict class using highest posterior probability

$$P(c_i|\mathbf{x}) = \frac{\hat{f}_i(\mathbf{x})P(c_i)}{\sum_{j=1}^{k} \hat{f}_j(\mathbf{x})P(c_j)}$$

$$\hat{y} = \arg\max_{c_i} \left\{ \hat{f}_i(\mathbf{x})P(c_i) \right\}$$

$$\hat{f}_i(\mathbf{x}) = \hat{f}(\mathbf{x}|\hat{\boldsymbol{\mu}}_i, \widehat{\Sigma}_i) = \frac{1}{(\sqrt{2\pi})^d \sqrt{|\widehat{\Sigma}_i|}} \exp\left\{ -\frac{(\mathbf{x}-\hat{\boldsymbol{\mu}}_i)^T \widehat{\Sigma}_i^{-1} (\mathbf{x}-\hat{\boldsymbol{\mu}}_i)}{2} \right\}$$

Bayes Classifier Pseudocode

BayesClassifier $(\boldsymbol{D} = \{(\boldsymbol{x}_j, y_j)\}_{j=1}^{n})$:

for $i = 1, \ldots, k$ **do**

$\boldsymbol{D}_i \leftarrow \{\boldsymbol{x}_j \mid y_j = c_i, j = 1, \ldots, n\}$ `// class-specific subsets`

$n_i \leftarrow |\boldsymbol{D}_i|$ `// cardinality`

$\hat{P}(c_i) \leftarrow n_i / n$ `// prior probability`

$\hat{\boldsymbol{\mu}}_i \leftarrow \frac{1}{n_i} \sum_{\boldsymbol{x}_j \in \boldsymbol{D}_i} \boldsymbol{x}_j$ `// mean`

$\boldsymbol{Z}_i \leftarrow \boldsymbol{D}_i - \boldsymbol{1}_{n_i} \hat{\boldsymbol{\mu}}_i^T$ `// centered data`

$\widehat{\Sigma}_i \leftarrow \frac{1}{n_i} \boldsymbol{Z}_i^T \boldsymbol{Z}_i$ `// covariance matrix`

return $\hat{P}(c_i), \hat{\boldsymbol{\mu}}_i, \widehat{\Sigma}_i$ *for all* $i = 1, \ldots, k$

Testing $(\boldsymbol{x}$ **and** $\hat{P}(c_i)$, $\hat{\boldsymbol{\mu}}_i$, $\widehat{\Sigma}_i$, **for all** $i \in [1, k])$:

$\hat{y} \leftarrow \arg\max_{c_i} \{ f(\boldsymbol{x} | \hat{\boldsymbol{\mu}}_i, \widehat{\Sigma}_i) \cdot P(c_i) \}$

return $\hat{y}$

Bayes Classifier Example: Iris Data

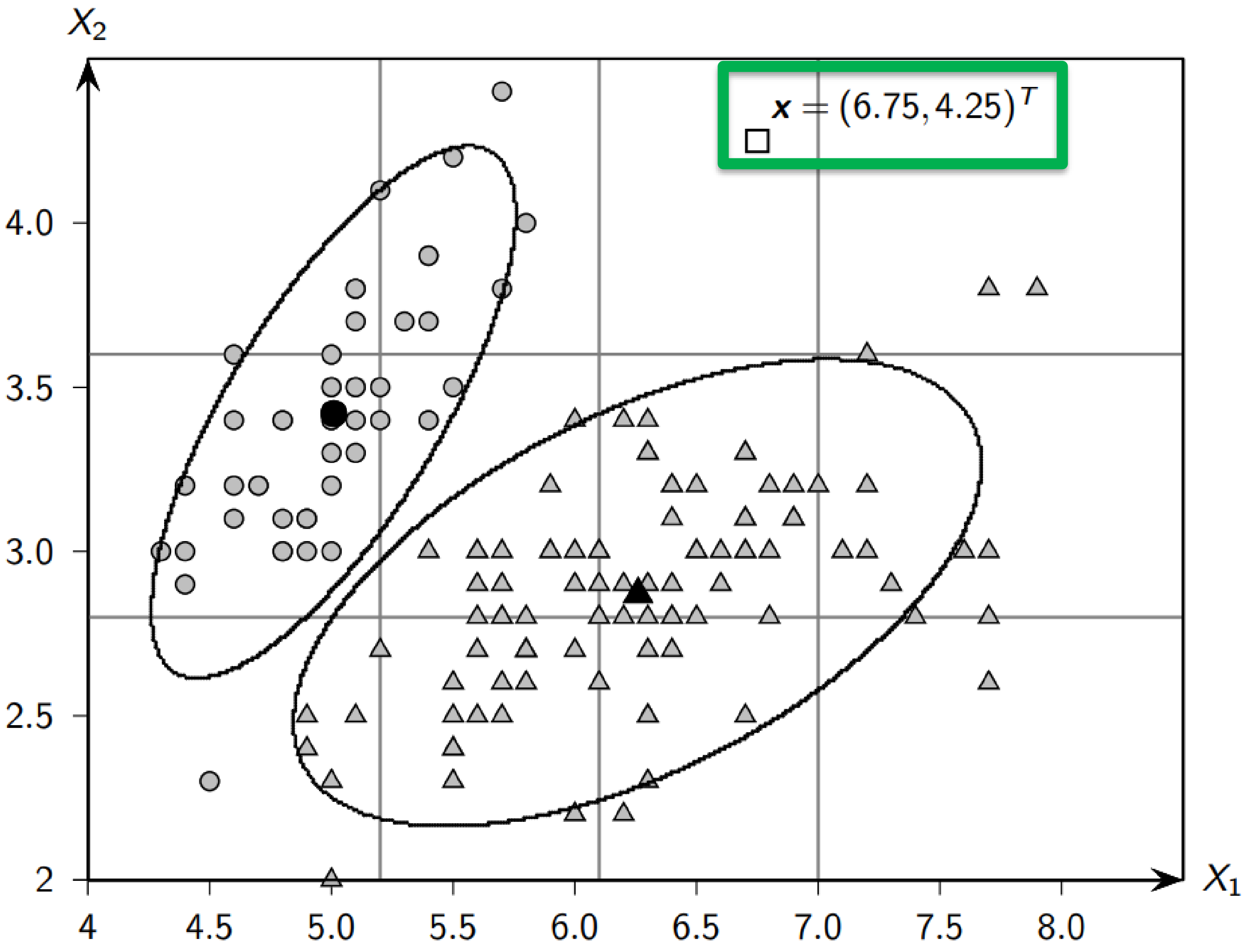
- Class 1: Iris-setosa
- Class 2: Other classes
- Steps:
  - **Compute priors**
  - **Estimate parameters**
  - Compute posterior for new test points

$$\hat{P}(c_1) = \frac{n_1}{n} = \frac{50}{150} = 0.33 \qquad \hat{P}(c_2) = \frac{n_2}{n} = \frac{100}{150} = 0.67$$

$$\hat{\mu}_1 = \begin{pmatrix} 5.006 \\ 3.418 \end{pmatrix} \qquad \hat{\mu}_2 = \begin{pmatrix} 6.262 \\ 2.872 \end{pmatrix}$$

$$\widehat{\Sigma}_1 = \begin{pmatrix} 0.1218 & 0.0983 \\ 0.0983 & 0.1423 \end{pmatrix} \qquad \widehat{\Sigma}_2 = \begin{pmatrix} 0.435 & 0.1209 \\ 0.1209 & 0.1096 \end{pmatrix}$$

Bayes Classifier Example: Iris Data



Bayes Classifier Example: Iris Data

- Class 1: Iris-setosa
- Class 2: Other classes
- Steps:
  - Compute priors
  - Estimate parameters
  - **Compute posterior for new test points**

$$\hat{P}(c_2|\mathbf{x}) > \hat{P}(c_1|\mathbf{x})$$

$$\hat{P}(c_1|\mathbf{x}) \propto \hat{f}(\mathbf{x}|\hat{\boldsymbol{\mu}}_1, \widehat{\Sigma}_1)\hat{P}(c_1) = (4.951 \times 10^{-7}) \times 0.33 = 1.634 \times 10^{-7}$$

$$\hat{P}(c_2|\mathbf{x}) \propto \hat{f}(\mathbf{x}|\hat{\boldsymbol{\mu}}_2, \widehat{\Sigma}_2)\hat{P}(c_2) = (2.589 \times 10^{-5}) \times 0.67 = 1.735 \times 10^{-5}$$

Bayes Classifier: Categorical

Bayes Classifier: Categorical

- Compute joint probability mass function (PMF) from **X**
- Compute joint PMF for each class by counting the number of times an attribute occurs

$$dom(X_j) = \{a_{j1}, a_{j2}, \ldots, a_{jm_j}\}$$

$$\boxed{P(\mathbf{x}|c_i) = f(\mathbf{v}|c_i) = f\left(\mathbf{X}_1 = \mathbf{e}_{1r_1}, \ldots, \mathbf{X}_d = \mathbf{e}_{dr_d} \mid c_i\right)}$$

$$\hat{f}(\mathbf{v}|c_i) = \frac{n_i(\mathbf{v})}{n_i}$$

Bayes Classifier: Unforeseen Events

- Avoid zero probabilities
- Use Laplace Smoothing
- Introduce smoothing parameter: α
- Larger values of α move data likelihood to uniform distribution
- m_j is domain for each attribute

$$\hat{f}(\mathbf{v}|c_i) = \frac{n_i(\mathbf{v})+1}{n_i + \prod_{j=1}^{d} m_j}$$

$$\hat{f}(\mathbf{v}_j|c_i) = \frac{n_i(\mathbf{v})+\alpha}{n_i + \alpha * \prod_{j=1}^{d} m_j}$$

Bayes Classifier Example: Iris Data

Assume that the `sepal length` and `sepal width` attributes in the Iris dataset have been discretized as shown below.

Bins	Domain
$[4.3, 5.2]$	Very Short (a_{11})
$(5.2, 6.1]$	Short (a_{12})
$(6.1, 7.0]$	Long (a_{13})
$(7.0, 7.9]$	Very Long (a_{14})

(a) Discretized `sepal length`

Bins	Domain
$[2.0, 2.8]$	Short (a_{21})
$(2.8, 3.6]$	Medium (a_{22})
$(3.6, 4.4]$	Long (a_{23})

(b) Discretized `sepal width`

We have $|dom(X_1)| = m_1 = 4$ and $|dom(X_2)| = m_2 = 3$.

Bayes Classifier Example: Iris Data

	Class: c_1	X_2 Short (e_{21})	X_2 Medium (e_{22})	X_2 Long (e_{23})	$\hat{f}_{X_1}$
X_1	Very Short (e_{11})	1/50	33/50	5/50	39/50
	Short (e_{12})	0	3/50	8/50	13/50
	Long (e_{13})	0	0	0	0
	Very Long (e_{14})	0	0	0	0
	$\hat{f}_{X_2}$	1/50	36/50	13/50	

	Class: c_2	X_2 Short (e_{21})	X_2 Medium (e_{22})	X_2 Long (e_{23})	$\hat{f}_{X_1}$
X_1	Very Short (e_{11})	6/100	0	0	6/100
	Short (e_{12})	24/100	15/100	0	39/100
	Long (e_{13})	13/100	30/100	0	43/100
	Very Long (e_{14})	3/100	7/100	2/100	12/100
	$\hat{f}_{X_2}$	46/100	52/100	2/100	

Bayes Classifier Example: Iris Data

Consider a test point $\mathbf{x} = (5.3, 3.0)^T$ corresponding to the categorical point (`Short`, `Medium`), which is represented as $\mathbf{v} = \begin{pmatrix} \mathbf{e}_{12}^T & \mathbf{e}_{22}^T \end{pmatrix}^T$.

The prior probabilities of the classes are $\hat{P}(c_1) = 0.33$ and $\hat{P}(c_2) = 0.67$. The likelihood and posterior probability for each class is given as

$$\hat{P}(\mathbf{x}|c_1) = \hat{f}(\mathbf{v}|c_1) = 3/50 = 0.06$$
$$\hat{P}(\mathbf{x}|c_2) = \hat{f}(\mathbf{v}|c_2) = 15/100 = 0.15$$
$$\hat{P}(c_1|\mathbf{x}) \propto 0.06 \times 0.33 = 0.0198$$
$$\hat{P}(c_2|\mathbf{x}) \propto 0.15 \times 0.67 = 0.1005$$

In this case the predicted class is $\hat{y} = c_2$.

Bayes Classifier

- May lack enough data to estimate joint pdf or pmf
  - Especially with many features
- Numeric attributes need to estimate covariances (d^2)
- Categorical attributes need to estimate all possible values
  - If attribute is binary, 2^d possibilities

Naive Bayes Classifier

Naive Bayes Classifier

- Assume attributes are independent

$$P(\mathbf{x}|c_i) = P(x_1, x_2, \ldots, x_d|c_i) = \prod_{j=1}^{d} P(x_j|c_i)$$

- Likelihood for class c_i for dimension x_j

$$P(x_j|c_i) \propto f(x_j|\hat{\mu}_{ij}, \hat{\sigma}_{ij}^2) = \frac{1}{\sqrt{2\pi}\hat{\sigma}_{ij}} \exp\left\{-\frac{(x_j - \hat{\mu}_{ij})^2}{2\hat{\sigma}_{ij}^2}\right\}$$

Naive Bayes Classifier

- Assumption leads to diagonal covariance
- Each class has mean vector and covariance matrix
  - 2d parameters to estimate per dimension x_j

$$\Sigma_i = \begin{pmatrix} \sigma_{i1}^2 & 0 & \dots & 0 \\ 0 & \sigma_{i2}^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & \dots & \sigma_{id}^2 \end{pmatrix}$$

Naive Bayes Classifier Pseudocode

NaiveBayes ($\boldsymbol{D} = \{(\boldsymbol{x}_j, y_j)\}_{j=1}^{n}$)**:**

for $i = 1, \ldots, k$ **do**

  $\boldsymbol{D}_i \leftarrow \left\{\boldsymbol{x}_j^T \mid y_j = c_i, j = 1, \ldots, n\right\}$ `// class-specific subsets`

  $n_i \leftarrow |\boldsymbol{D}_i|$ `// cardinality`

  $\hat{P}(c_i) \leftarrow n_i / n$ `// prior probability`

  $\hat{\boldsymbol{\mu}}_i \leftarrow \frac{1}{n_i} \sum_{\boldsymbol{x}_j \in \boldsymbol{D}_i} \boldsymbol{x}_j$ `// mean`

  $\overline{\boldsymbol{D}}_i = \boldsymbol{D}_i - \boldsymbol{1} \cdot \hat{\boldsymbol{\mu}}_i^T$ `// centered data for class` c_i

  for $j = 1, .., d$ **do** `// class-specific var for` j`th attribute`

    $\hat{\sigma}_{ij}^2 \leftarrow \frac{1}{n_i} (\overline{X}_j^{\,i})^T \, (\overline{X}_j^{\,i})$ `// variance`

  $\hat{\boldsymbol{\sigma}}_i \leftarrow \left(\hat{\sigma}_{i1}^2, \ldots, \hat{\sigma}_{id}^2\right)^T$ `// class-specific attribute variances`

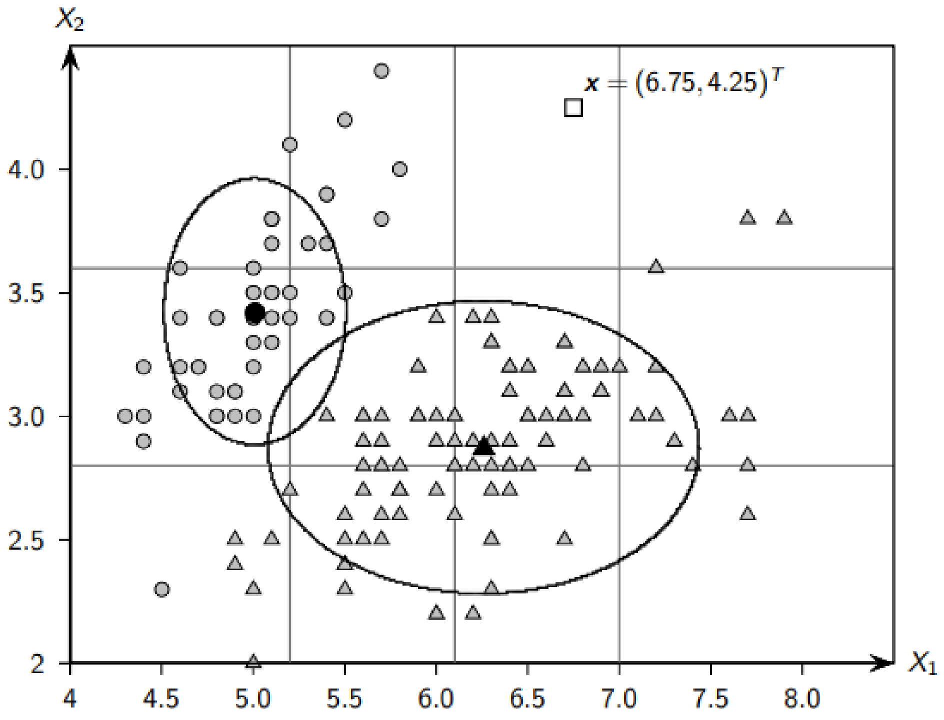
return $\hat{P}(c_i), \hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{\sigma}}_i$ for all $i = 1, \ldots, k$

Testing ($\boldsymbol{x}$ **and** $\hat{P}(c_i)$, $\hat{\boldsymbol{\mu}}_i$, $\hat{\boldsymbol{\sigma}}_i$, **for all** $i \in [1, k]$)**:**

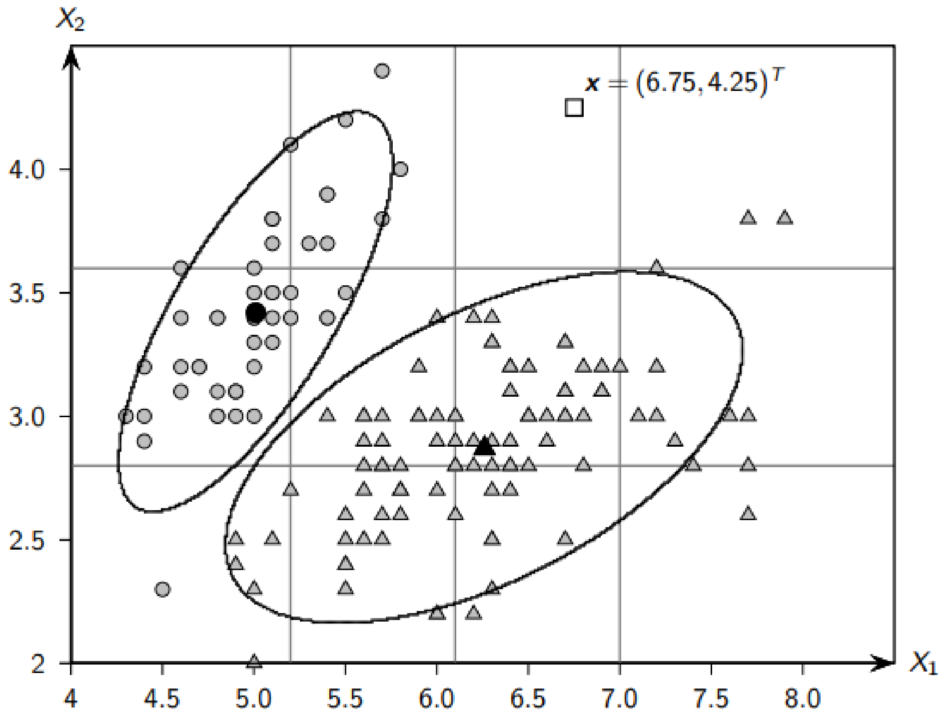
$\hat{y} \leftarrow \arg\max_{c_i} \left\{ \hat{P}(c_i) \prod_{j=1}^{d} f(x_j | \hat{\mu}_{ij}, \hat{\sigma}_{ij}^2) \right\}$

return $\hat{y}$

Naive Bayes vs Full Bayes



(a) Naive Bayes



(b) Full Bayes

Naive Bayes Classifier: Categorical

Bayes Classifier: Categorical

- Compute joint probability mass function (PMF) from **X**
- Compute joint PMF for each class by counting the number of times an attribute occurs

$$P(\mathbf{x}|c_i) = \prod_{j=1}^{d} P(x_j|c_i) = \prod_{j=1}^{d} f\left(\mathbf{X}_j = \mathbf{e}_{jr_j} \mid c_i\right)$$

$$\hat{f}(\mathbf{v}_j|c_i) = \frac{n_i(\mathbf{v}_j)}{n_i}$$

Naive Bayes Classifier: Unforeseen Events

- Avoid zero probabilities
- Use Laplace Smoothing
- Introduce smoothing parameter: α
- Larger values of α move data likelihood to uniform distribution
- m_j is dimensionality

$$\boxed{\hat{f}(\mathbf{v}_j|c_i) = \frac{n_i(\mathbf{v}_j)+1}{n_i+m_j}}$$

$$\hat{f}(\mathbf{v}_j|c_i) = \frac{n_i(\mathbf{v}_j)+\alpha}{n_i+\alpha * m_j}$$

Naive Bayes Classifier: Parameter Estimation

Naive Bayes Classifier

- Estimate posterior probability for each class c_i
- Predicted class has highest probability ($\hat{y}$)
- Two approaches:
  - Maximum Likelihood Estimation (MLE)
  - Maximum a posteriori (MAP)

$$P(c_i|\mathbf{x}) = \frac{P(\mathbf{x}|c_i) \cdot P(c_i)}{P(\mathbf{x})}$$

$$\arg\max_{c_i} \{ P(\mathbf{x}|c_i) P(c_i) \}$$

Maximum Likelihood Estimation (MLE)

- Maximize posterior by maximizing data likelihood, P($\mathbf{x}$|c_i)
- Similar to EM for GMM
  - Not "weighted" but only use samples from class to compute mean and variance for each feature and class

$$P(\mathbf{x}|c_i) = P(x_1, x_2, \ldots, x_d|c_i) = \prod_{j=1}^{d} P(x_j|c_i)$$

$$P(x_j|c_i) \propto f(x_j|\hat{\mu}_{ij}, \hat{\sigma}_{ij}^2) = \frac{1}{\sqrt{2\pi}\hat{\sigma}_{ij}} \exp\left\{-\frac{(x_j - \hat{\mu}_{ij})^2}{2\hat{\sigma}_{ij}^2}\right\}$$

Maximum a Posteriori (MAP)

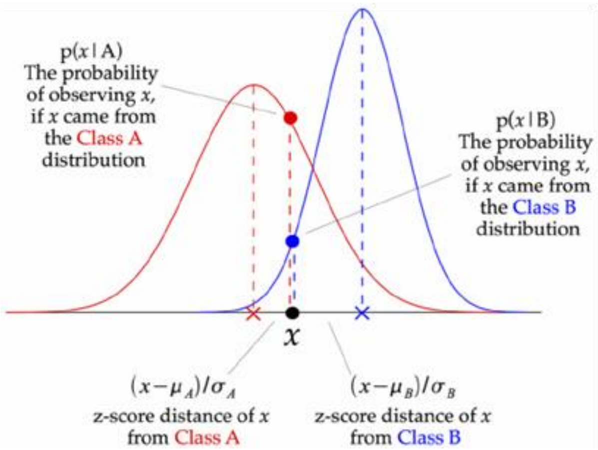
TEXAS A&M UNIVERSITY Engineering

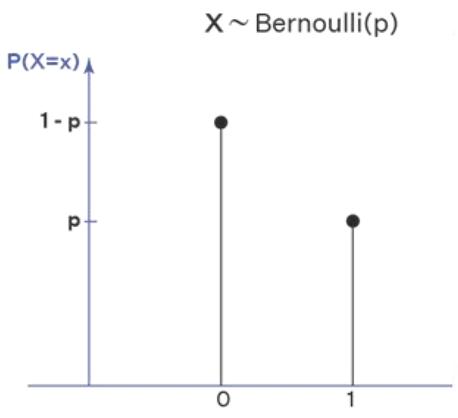
- Another option is to consider the most likely parameter value given the data, “maximum a posteriori” or MAP
- Now include prior with estimation

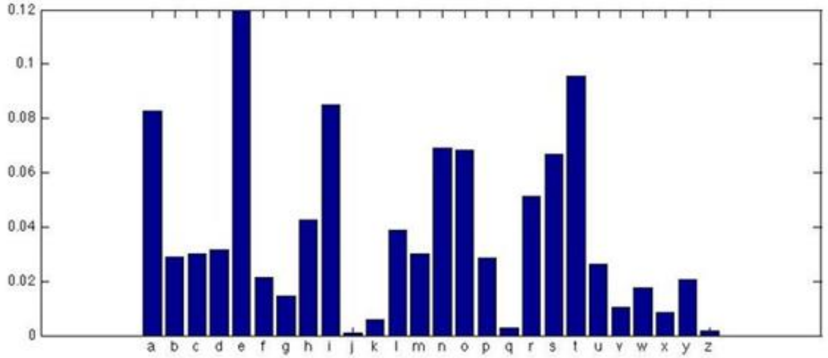
$$\arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)$$

Distributions Used with Naive Bayes

- Normal/Gaussian
- Bernoulli
- Multinomial







Distributions Used with Naive Bayes

Features

Classes

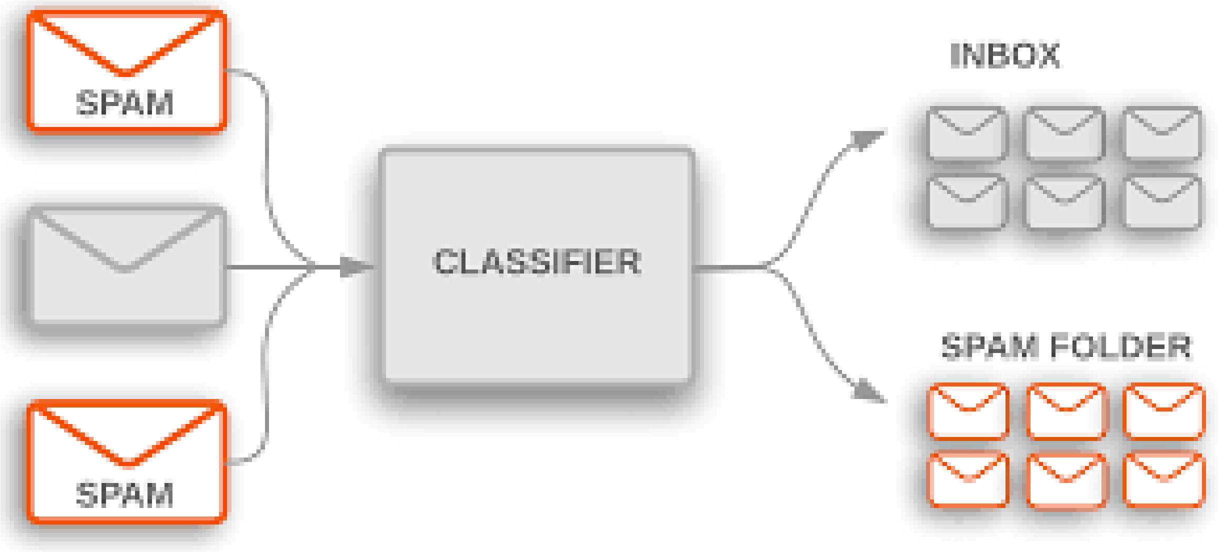
	Features are discrete Binary/Boolean functions	Features are discrete occurrence counts	Features are numerical values sampled from a continuous function with a Gaussian distribution
Binary Labels {T,F}, {Pass,Fail}; {0,1}, {-1,1)	**sklearn.naive_bayes. BernoulliNB()** with 2 classes	**sklearn.naive_bayes. MultinomialNB()** with 2 classes	**sklearn.naive_bayes. GaussianNB()** with 2 classes
Multinomial Labels (3 or more classes) {0-6}, {airplane, bicycle, car}, {pass, fail type1, fail type 2, fail type 3}	**sklearn.naive_bayes. BernoulliNB()**	**sklearn.naive_bayes. MultinomialNB()**	**sklearn.naive_bayes. GaussianNB()**

* Use variant ComplimentNB if unbalanced distribution and CategoricalNB if data is categorical {Monday, Tuesday, Wed….} rather than occurrences

Practical Applications of Naive Bayes

Naive Bayes Application: Spam Filter

- Jupyter Notebook

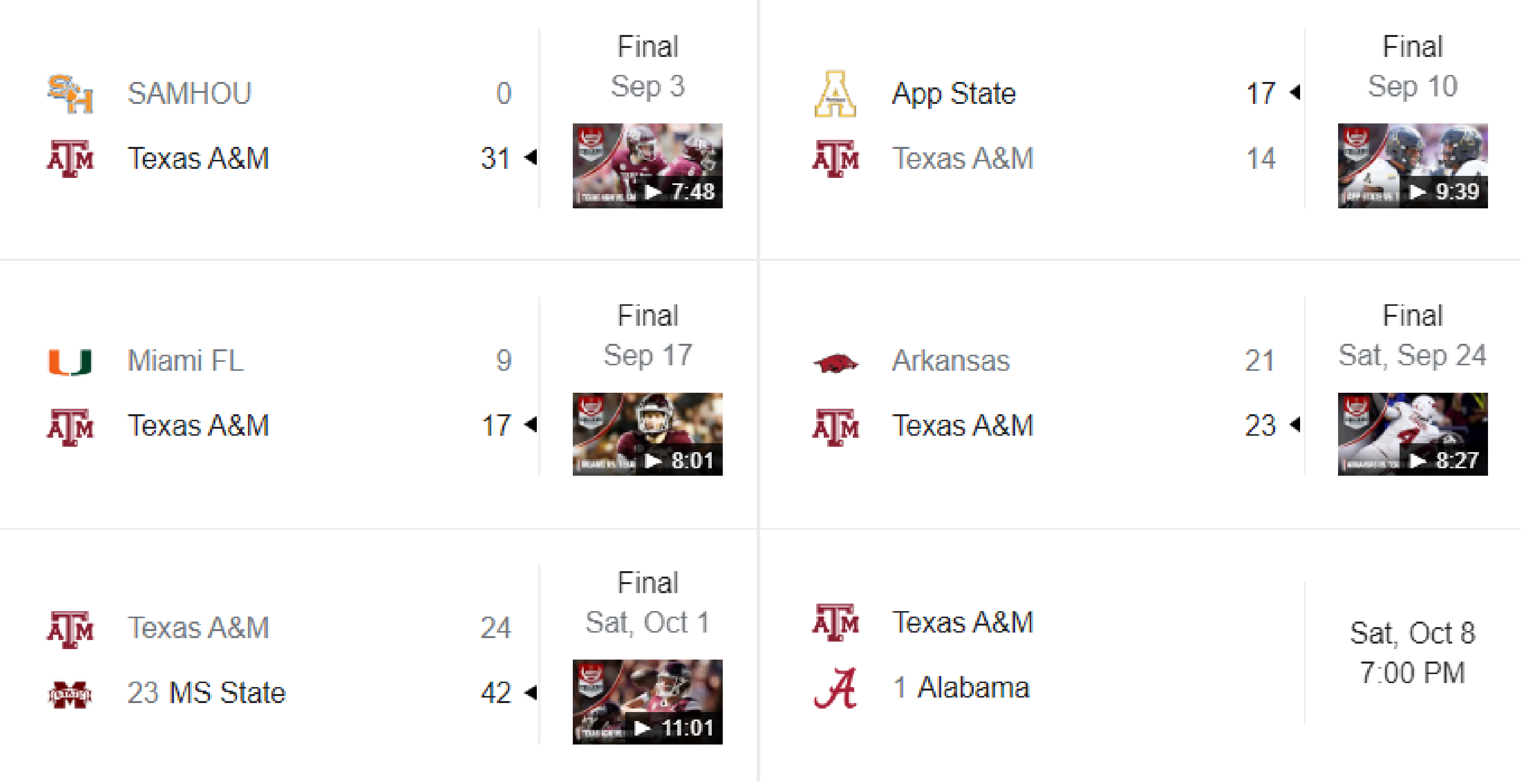


Naive Bayes Application: Predicting Football Games

SAMHOU	0	Texas A&M	31 ◂	Final	Sep 3	▶ 7:48
App State	17 ◂	Texas A&M	14	Final	Sep 10	▶ 9:39
Miami FL	9	Texas A&M	17 ◂	Final	Sep 17	▶ 8:01
Arkansas	21	Texas A&M	23 ◂	Final	Sat, Sep 24	▶ 8:27
Texas A&M	24	23 MS State	42 ◂	Final	Sat, Oct 1	▶ 11:01
Texas A&M		1 Alabama			Sat, Oct 8 7:00 PM	

Naive Bayes Application: Predicting Football Games

TEXAS A&M UNIVERSITY Engineering



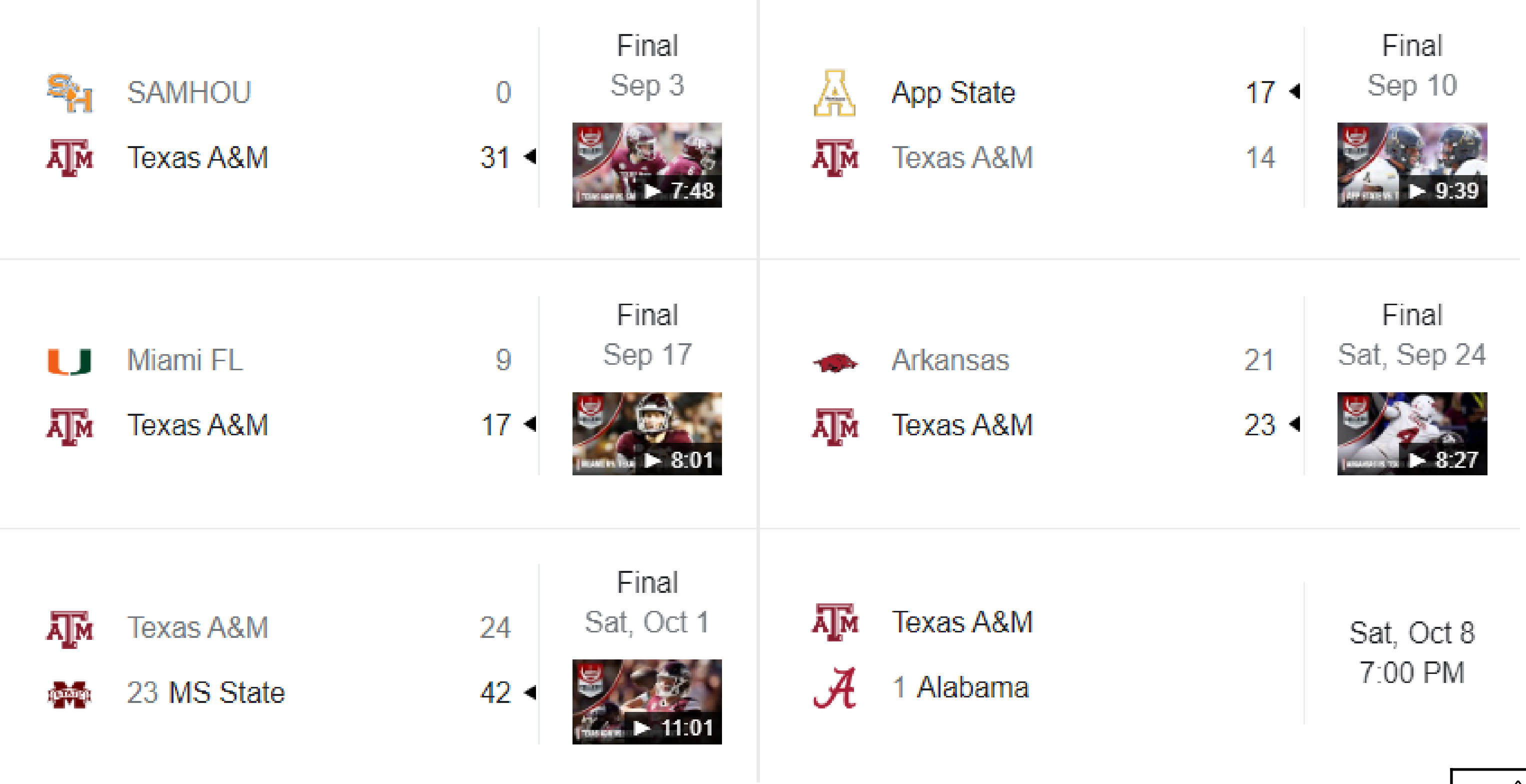
X= W, L, W, W, L

Likelihood = $p^X * (1-p)^{1-X}$

MLE: $\theta_{ML} = \underset{\theta}{argmax}\, L(x,\theta)$ $\quad \theta_{MLE}: \hat{p} = \sum_{i=1}^{D} X_i / n = 0.6$

MAP: $\theta_{MAP} = \underset{\theta}{argmax}\, L(x,\theta) * P(\theta)$ $\quad \theta_{MAP}: \hat{p} = 0.615$

Naive Bayes Application: Predicting Football Games



X= W, L, W, W, L

Likelihood = $p^X * (1-p)^{1-X}$

MLE: $\theta_{ML} = \underset{\theta}{argmax}\, L(x,\theta)$ $\quad \theta_{MLE}: \hat{p} = \sum_{i=1}^{D} X_i / n = 0.6$

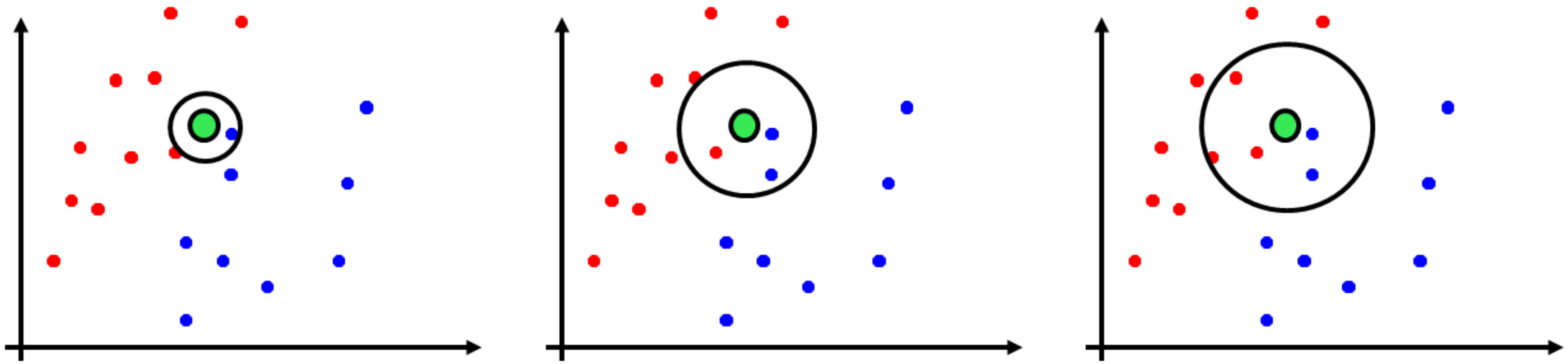
MAP: $\theta_{MAP} = \underset{\theta}{argmax}\, L(x,\theta) * P(\theta)$ $\quad \theta_{MAP}: \hat{p} = 0.615$

$\hat{p}$	prior	likelihood	posterior
0.538	0.181818	0.372044	0.067644
0.615	0.363636	0.380833	0.138485
0.666	0.090909	0.406964	0.036997
0.692	0.181818	0.426238	0.077498
0.844	0.090909	0.625548	0.056868
0.9	0.090909	0.739	0.067182

k-Nearest Neighbors Classifier

k-Nearest Neighbors Classification

- Non-parametric and discriminative classifier
- Assign class based on the majority vote of the k-closest neighbors



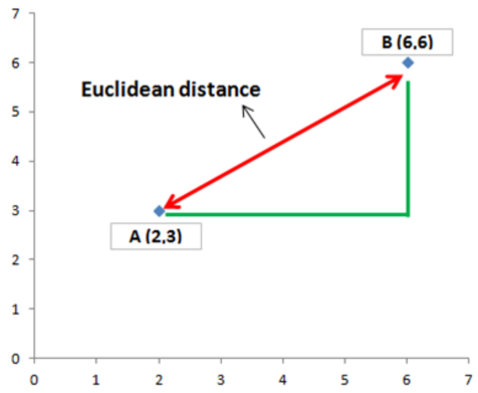
k=1;
New sample =>

k=3;
New sample =>

k=5;
New sample =>

k-Nearest Neighbors Classification

- Very simple nonparametric model
  - Select a distance function L(x, x’) (e.g., Euclidean)
  - Choose a hyperparameter K (usually odd)



$$\textit{Euclidean distance}\ (a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

Image from: Soner Yildirim, K-Nearest Neighbors (kNN) — Explained

k-Nearest Neighbors Classification

TEXAS A&M UNIVERSITY Engineering

- For each instance in the test set (new, unobserved) x,
- Select in D the k examples that are nearest to x according to $L(\mathbf{x}, \mathbf{x}^i)$ and keep their index in set $\{s_1, \cdots, s_K\}$, find:
- For classification: $\hat{y} = \underset{c_i}{\operatorname{argmax}} K_i$
- In scikit-learn we use KNeighborsClassifier()

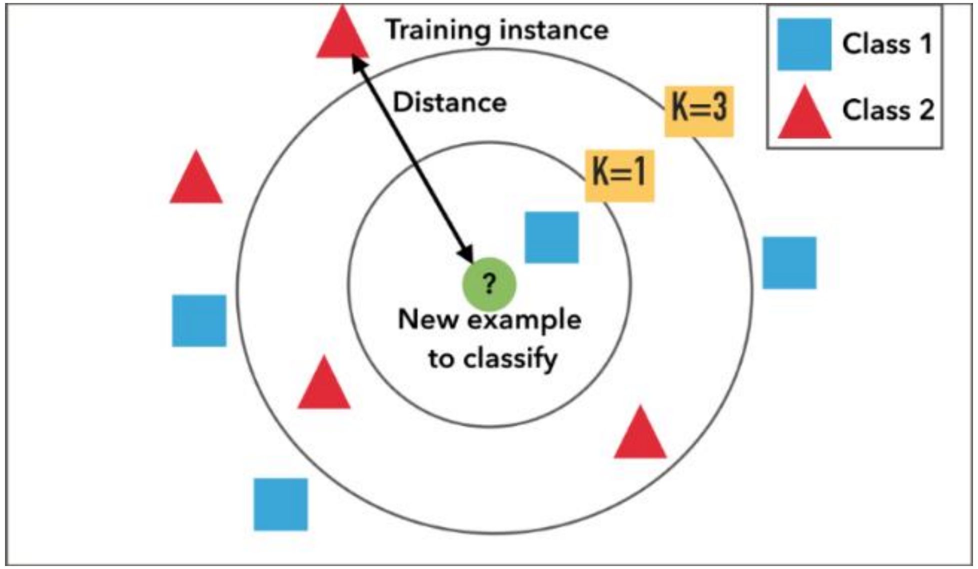
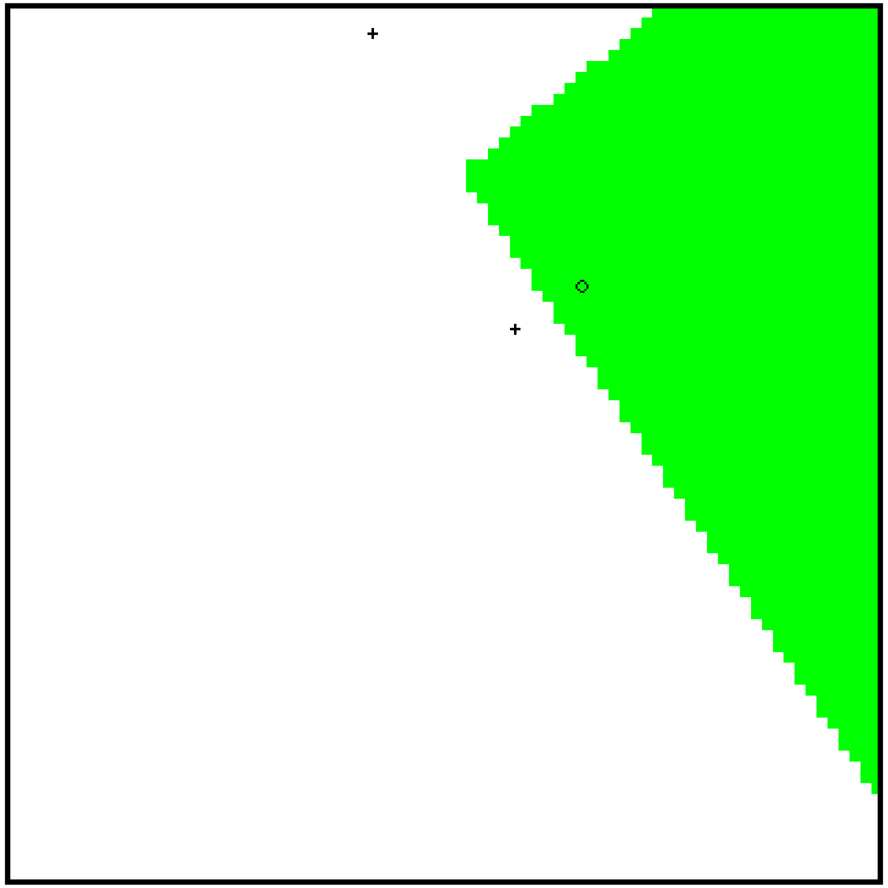


Image from: Martin Matta, k-Nearest Neighbors in Python From Scratch.

k-Nearest Neighbors Classification

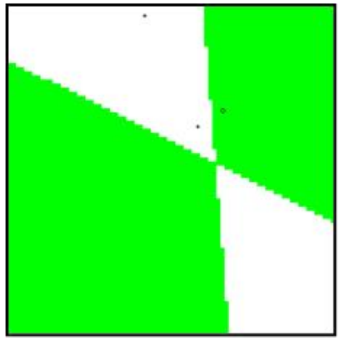
- Classification from similarity
  - Case-based reasoning
  - Predict an instance's label using similar instances
- Nearest-neighbor classification
  - 1-NN: copy the label of the most similar data point
  - K-NN: vote the k nearest neighbors
  - Key issue: how to define similarity and number of k



http://www.cs.cmu.edu/~zhuxj/courseproject/knndemo/KNN.html

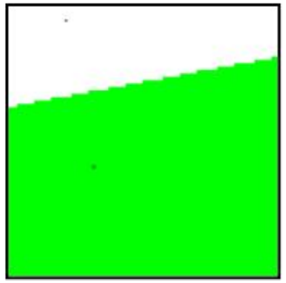
Image from: Martin Matta, k-Nearest Neighbors in Python From Scratch.

Decision Boundaries for Varying Number Training Samples

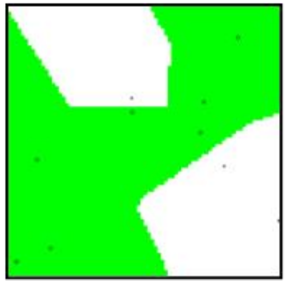


Truth

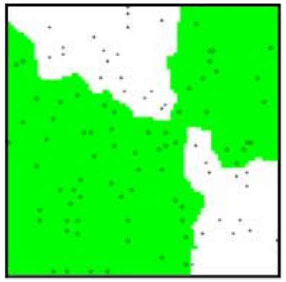
2 Observations



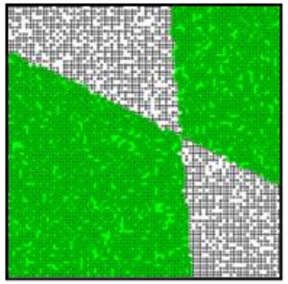
10 Observations



100 Observations



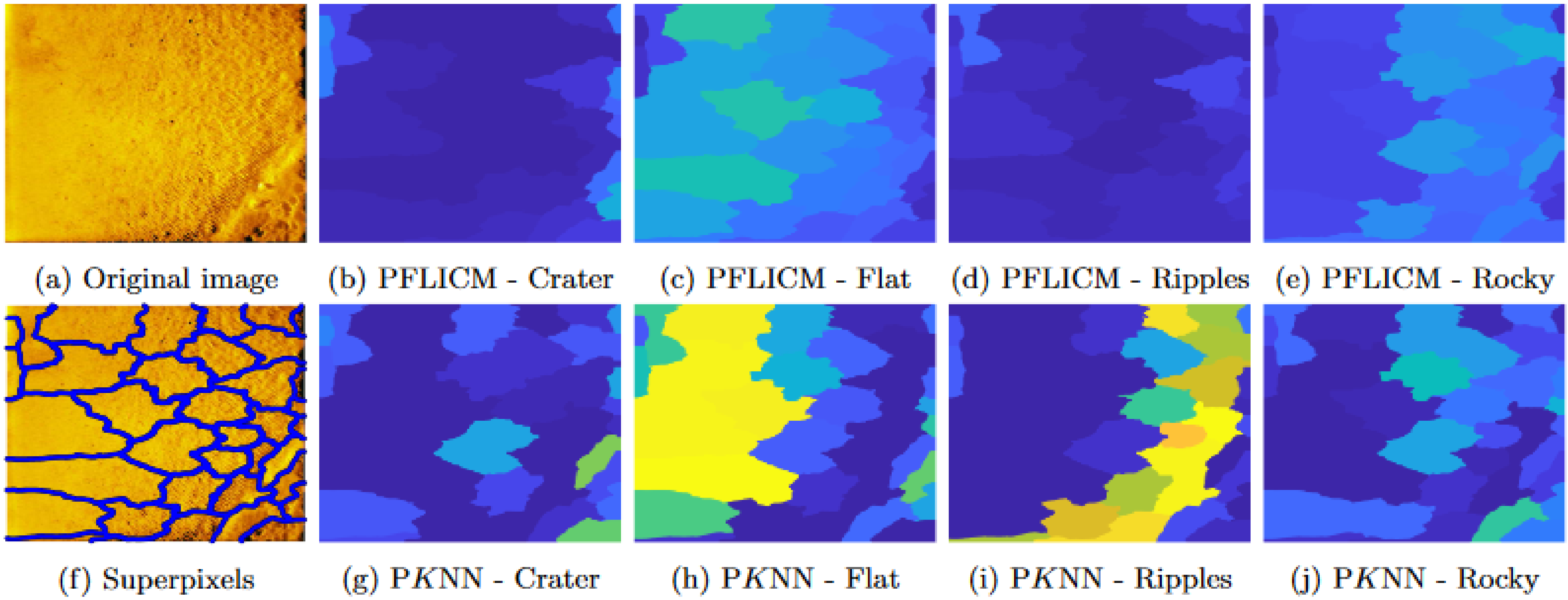
10000 Observations



http://www.cs.cmu.edu/~zhuxj/courseproject/knndemo/KNN.html

Practical Application of k-NN

Sonar Image Segmentation



J. Peeples, et al. Comparison of Possibilistic Fuzzy Local Information C-Means and Possiblisitic K-Nearest Neighbors for Synthetic Aperture Sonar Segmentation. SPIE, 2019.

Next class

- Introduction to Classification II
  - Methodology
  - Metrics
  - Overfitting and underfitting
- Decision tree classification

TEXAS A&M UNIVERSITY Engineering

Thank You! Questions?

Joshua Peeples, Ph.D.

https://www.joshpeeples.com/

jpeeples@tamu.edu

Supplemental Slides

Useful Links

- [StatQuest: Bayes' Theorem]
- [StatQuest: Naive Bayes]
- [StatQuest: Gaussian Naive Bayes]