CS 4342 - Machine Learning

WPI

Fall 2019

Lecture 1: *Introduction to Machine Learning & Probability*

Instructor: Ali Yousefi

Machine Learning: What it is and why it matters

- **Machine learning** is a method of data analysis that automates analytical model building. It is a branch of **artificial intelligence (AI)** based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.
- Examples of machine learning applications
    1. The heavily hyped, self-driving Google car? The essence of machine learning.
    2. Online recommendation offers such as those from Amazon and Netflix? Machine learning applications for everyday life.
    3. Knowing what customers are saying about you on Twitter? Machine learning combined with linguistic rule creation.
    4. Fraud detection? One of the more obvious, important uses in our world today.
- **Machine learning** is an exciting and alarming area of research within AI. Endowing machines with the ability to learn certain tasks could be extremely useful, could increase productivity, and help expedite all sorts of activities, from search algorithms to data mining.

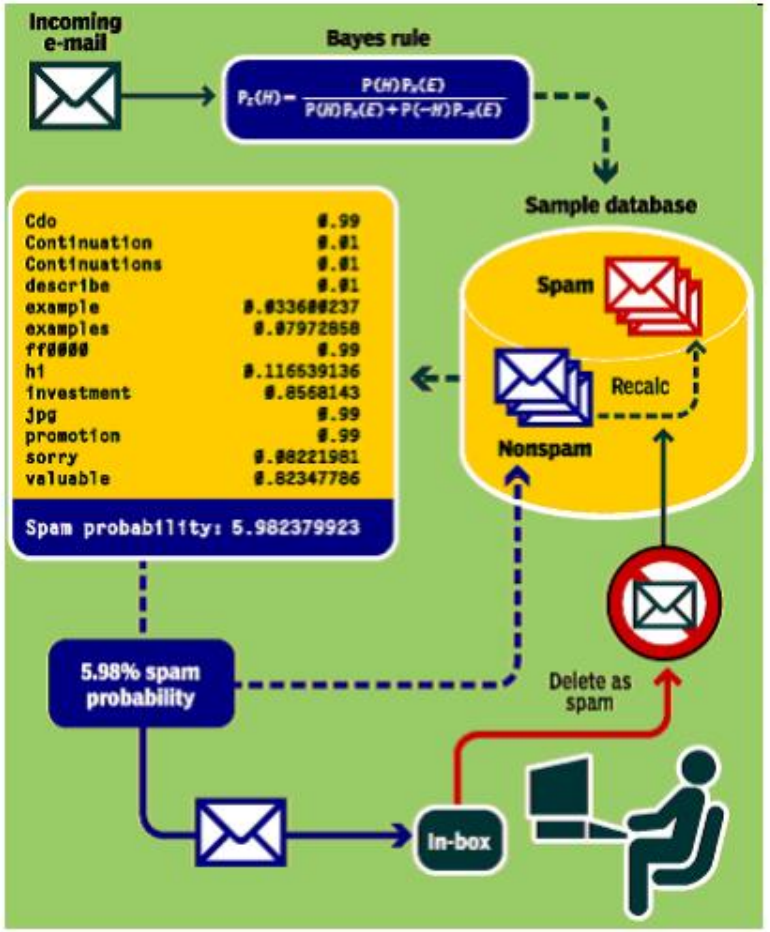
Examples of Machine Learning (ML) Problems

Classification, visual object recognition

Examples of Machine Learning (ML) Problems

- **Binary classification problem**

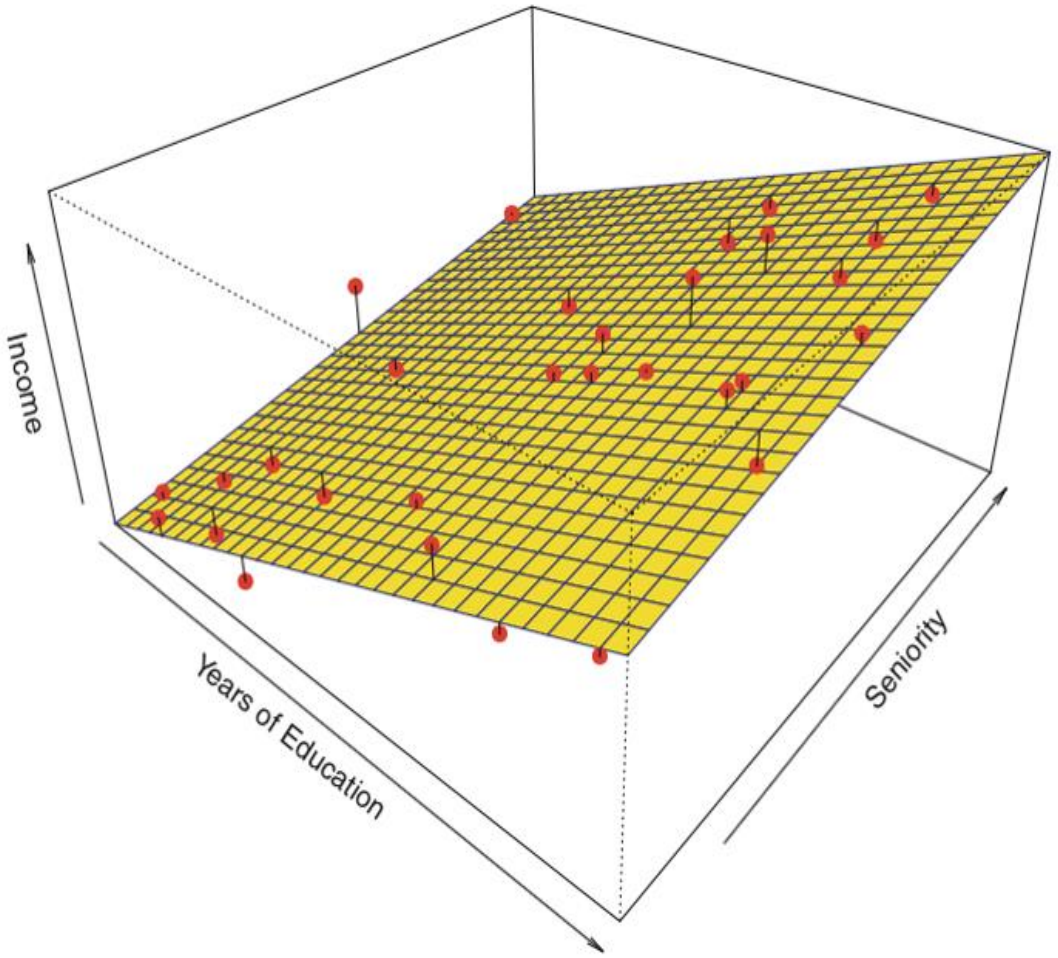
  Is this e-mail spam or useful (ham)?



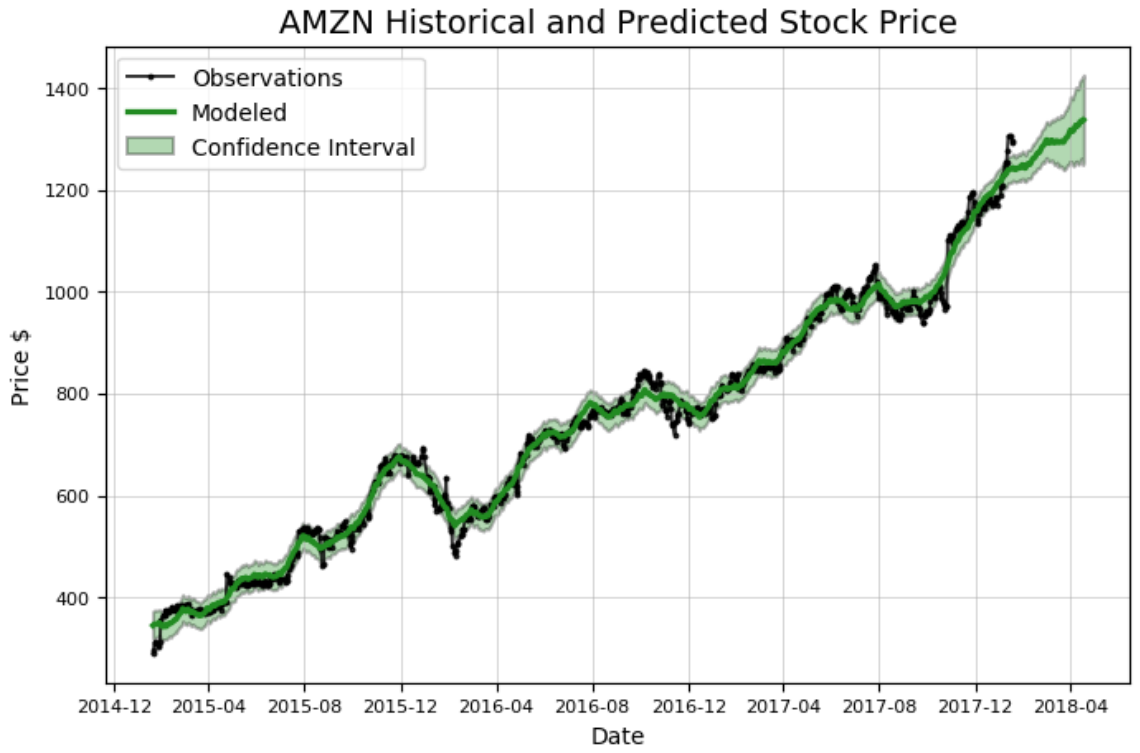
Spam Filter Express
http://www.spam-filter-express.com/

Classification, spam filtering

Examples of Machine Learning (ML) Problems

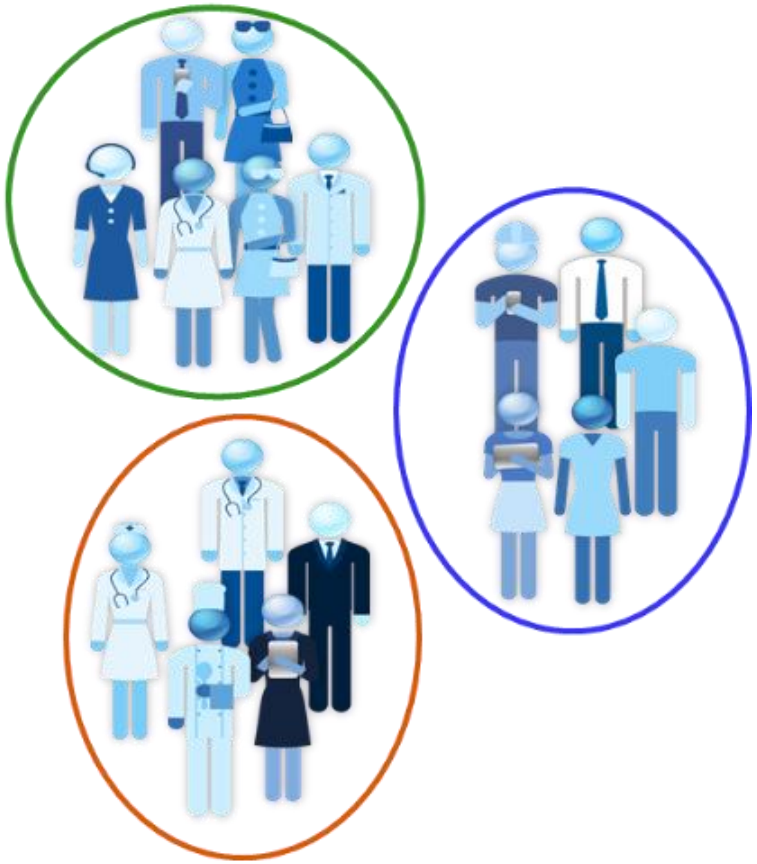


Linear Regression Graph, prediction of income



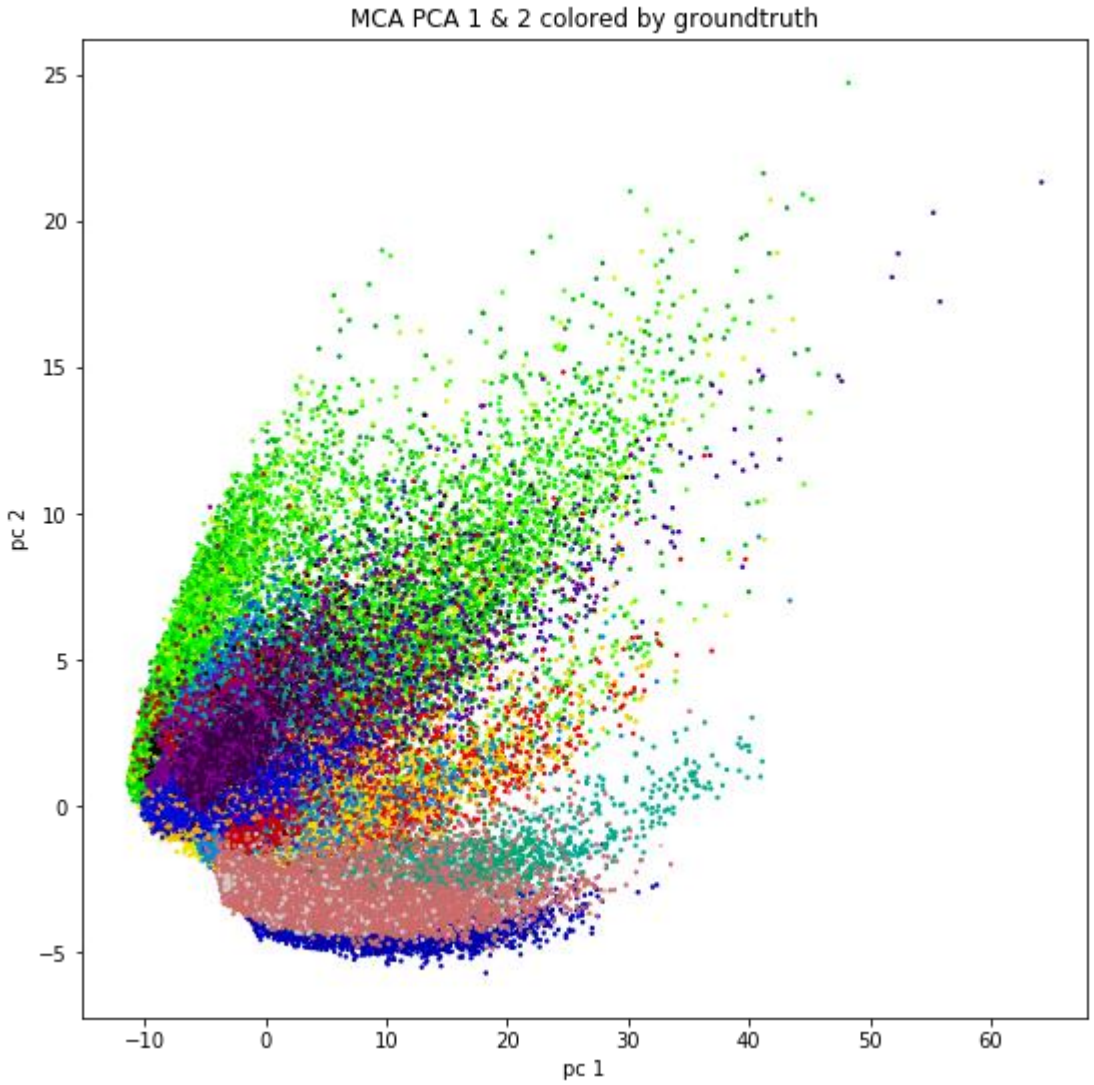
Regression, stock price prediction

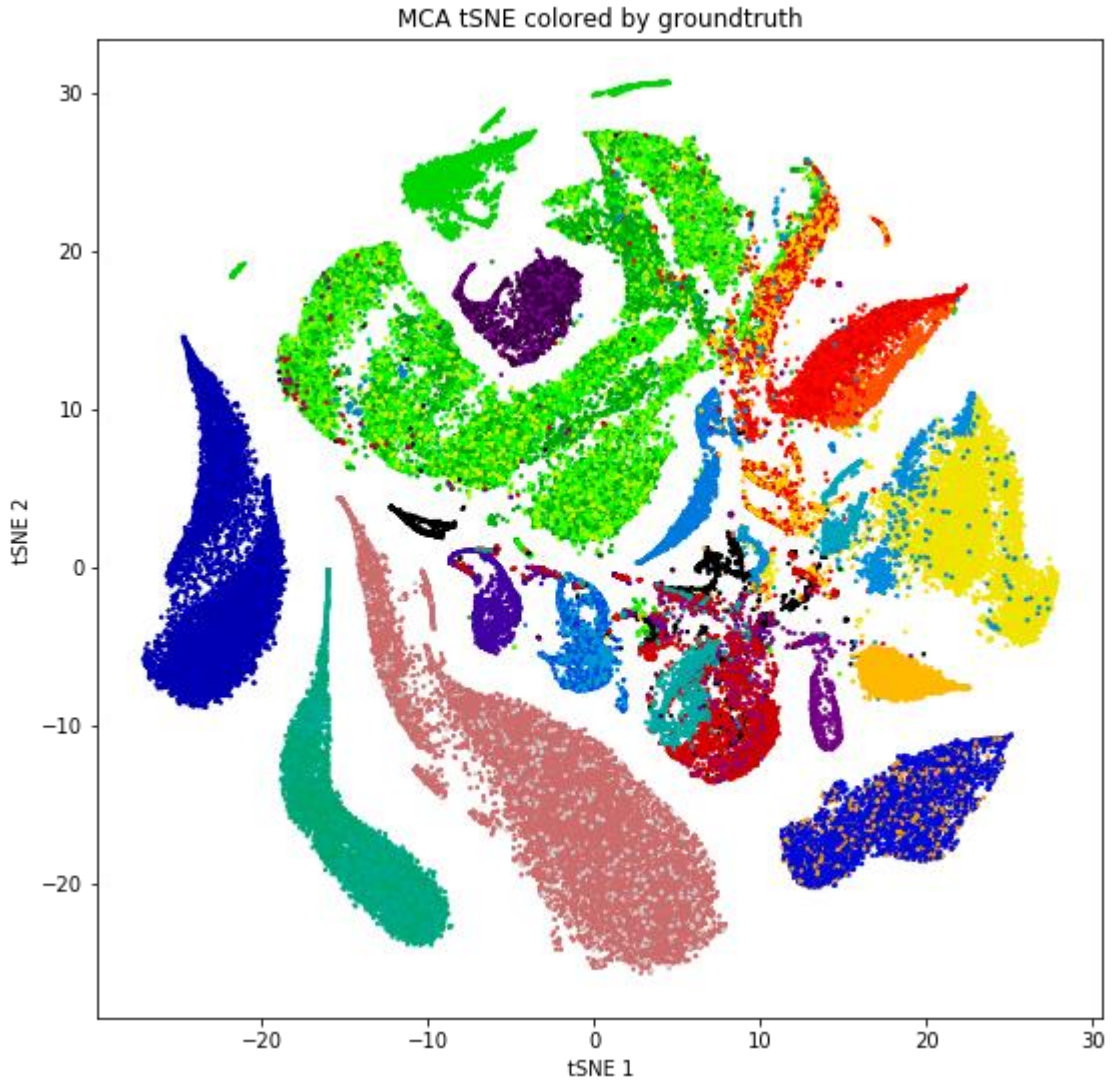
Examples of Machine Learning (ML) Problems



Clustering, group the data based on the similarities

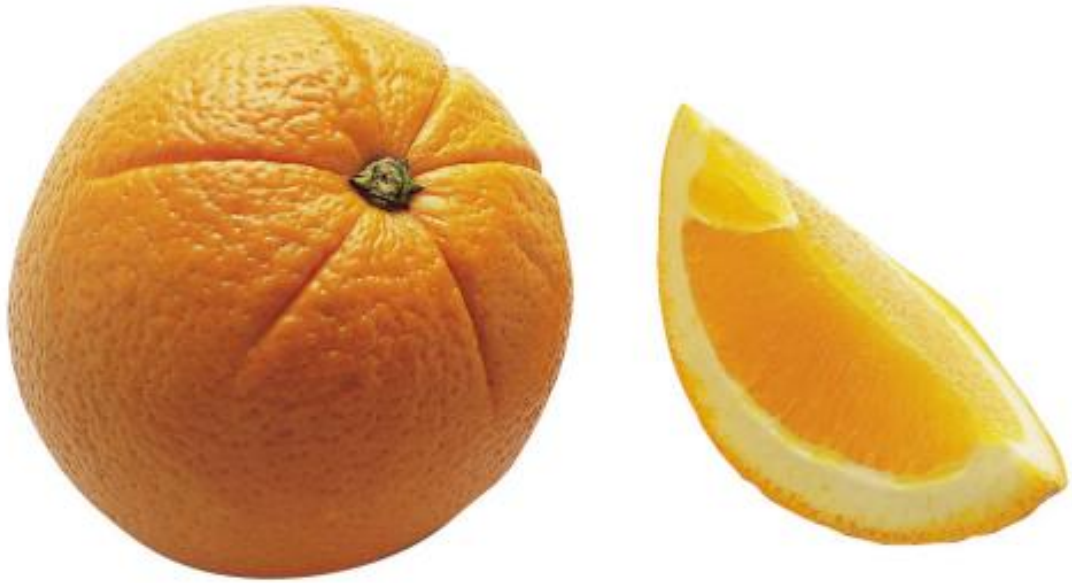
Examples of Machine Learning (ML) Problems



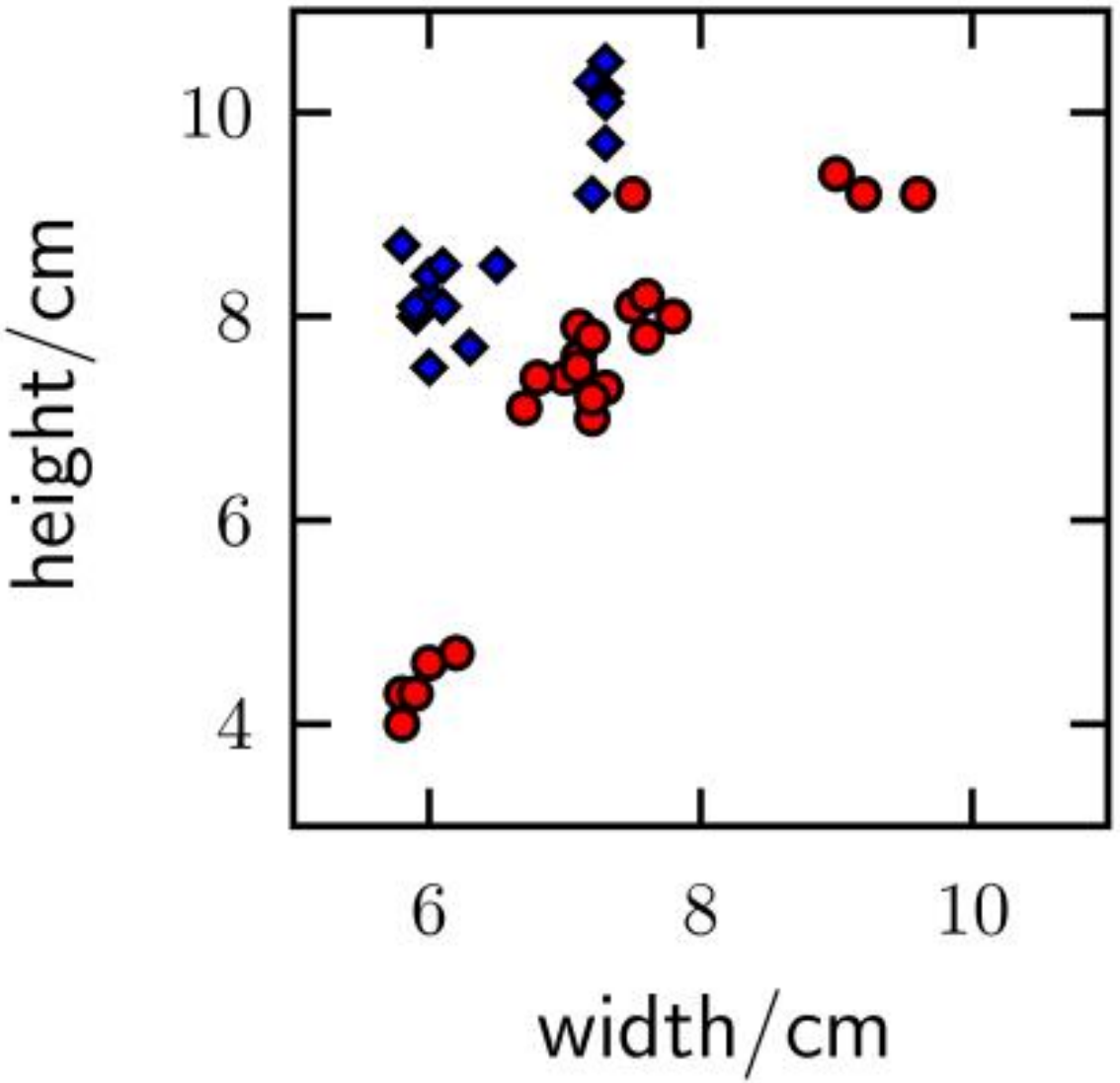


Dimensionality Reduction, cell RNA-sequence (20-dimension), PCA and TSNE

An Example ML Problem: Oranges and Lemons

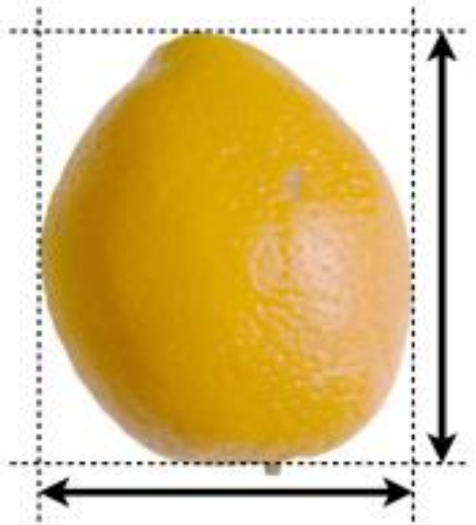


A Two-dimensional Dataset



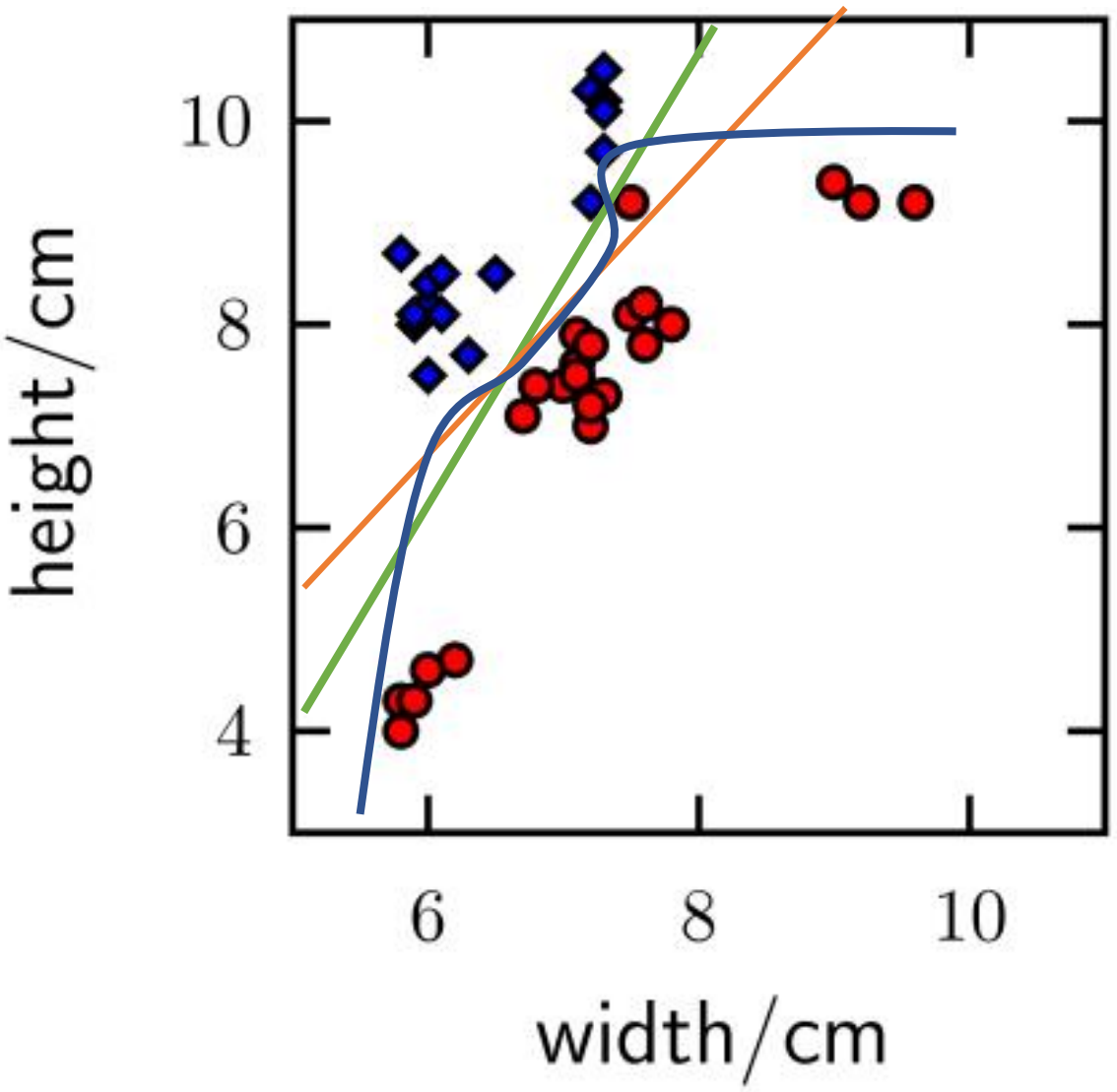
Oranges:

Lemons:



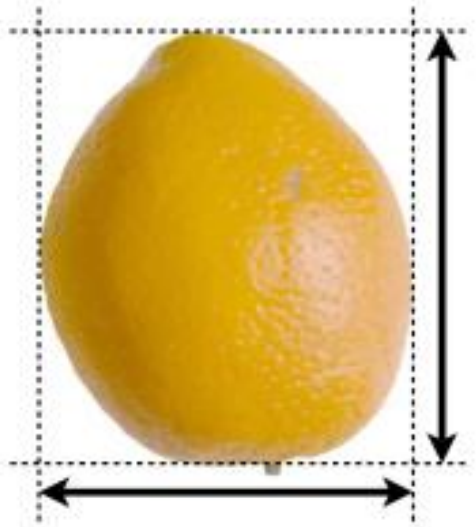
The Odd-ball Orange

Which Classifier?

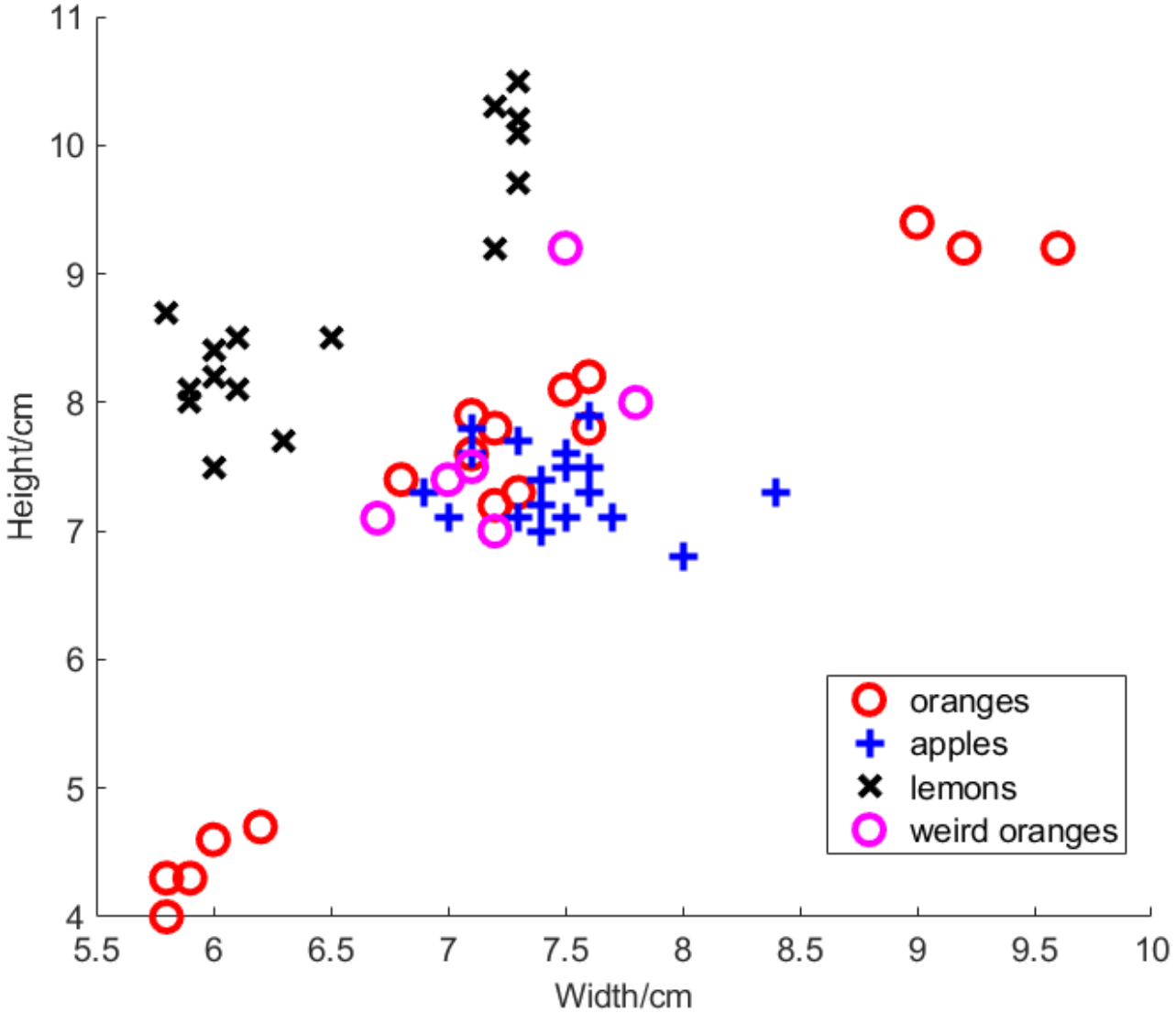


Oranges:

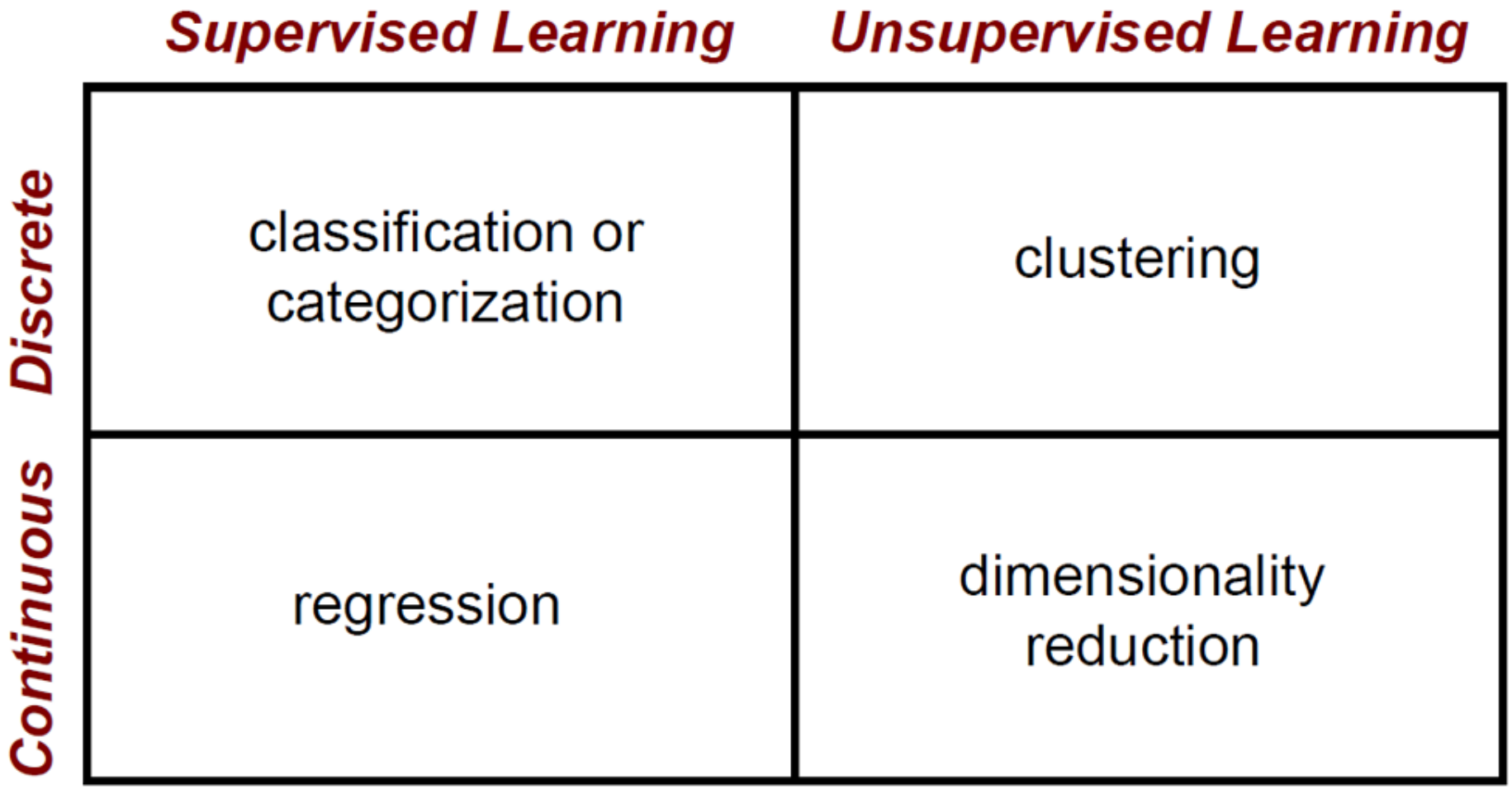
Lemons:



Oranges, Lemons and Apples



Machine Learning Problems



- Semi-supervised learning
- Reinforcement Learning (RL)

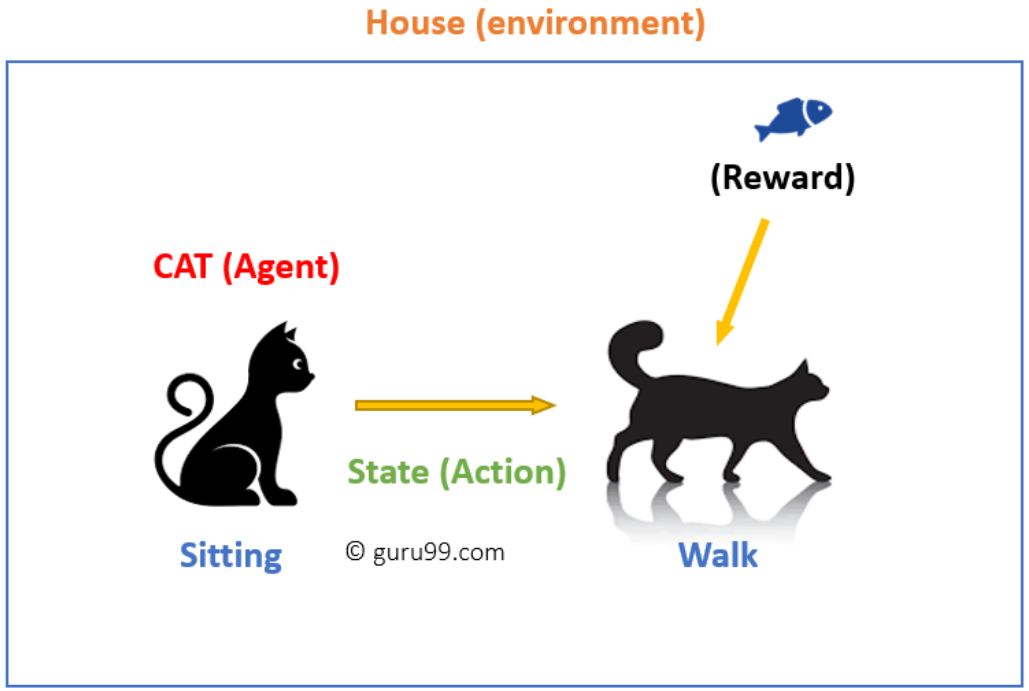
Types of Machine Learning Problems

- **Predictive or supervised learning approach:** The goal is to learn a mapping from inputs x to outputs y, given a labeled set of input-output pairs $\boldsymbol{D} = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{N}$
- $\boldsymbol{x_i}$, **input**, is a D-dimensional vector of numbers (**features**, **attributes** or **covariates**)
- y_i, **response variable** or **output**, can be anything. Most methods assume y_i is a **categorical or nominal** variable from some finite set, $\mathrm{y}_i \in \{1, \ldots, C\}$, or y_i is real valued scalar.
- y_i is **categorical**, then we have a **classification or pattern recognition** problem
- y_i is **real-valued**, then we have a **regression problem**

Types of Machine Learning Problems

- **Descriptive or unsupervised learning approach:** The goal is to find "interesting patterns" in the data, given only inputs $\boldsymbol{D} = \{\boldsymbol{x_i}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$. This is sometimes called **knowledge discovery**.

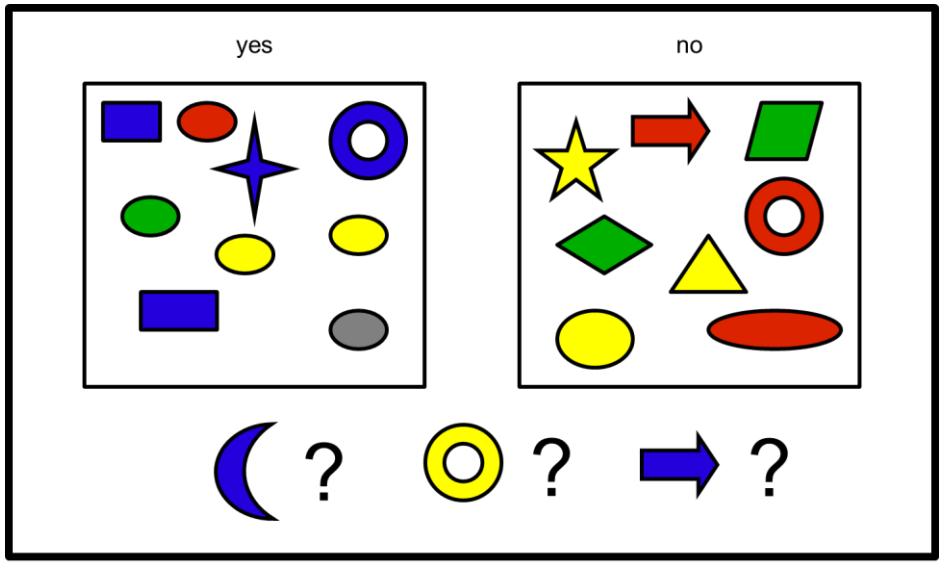
- **Reinforcement learning:** It is about taking suitable action to maximize reward in a particular situation.



Supervised Learning: Classification Problem

$$\hat{y} = \hat{f}(\mathbf{x})$$

- We use the hat symbol to denote an estimate. Our main goal is to make predictions on novel inputs, meaning ones that we have not seen before (this is called generalization), since predicting the response on the training set is easy (we can just look up the answer).
- Binary classification problem $y_i \in \{0, 1\}$



D features (attributes)

N cases

Color	Shape	Size (cm)
Blue	Square	10
Red	Ellipse	2.4
Red	Ellipse	20.7

Label
1
1
0

Probabilistic Predictions

- Probability distribution over possible labels

$$\boldsymbol{p(y|\mathbf{x}, \mathcal{D})}$$

- For a binary classifier, we only need $p(y = 1|x, \mathcal{D})$

$$p(y = 1|\mathrm{x}, \mathcal{D}) + p(y = 0|\mathrm{x}, \mathcal{D}) = 1$$

- If we have multiple models, we can make this assumption explicit by writing

$$\boldsymbol{p(y|\mathbf{x}, \mathcal{D}, M)}$$

- "Best guess" or "true label" – MAP estimate

$$\hat{\boldsymbol{y}} = \hat{\boldsymbol{f}}(\mathbf{x}) = \underset{c=1\cdots C}{\operatorname{argmax}}\, \boldsymbol{p(y = c|\mathbf{x}, \mathcal{D})}$$

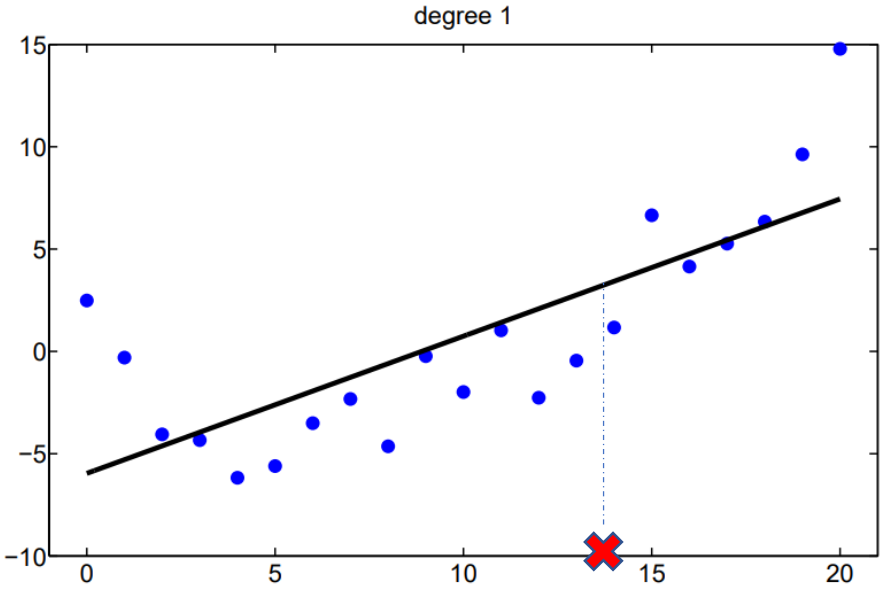
Supervised Learning: Regression Problem

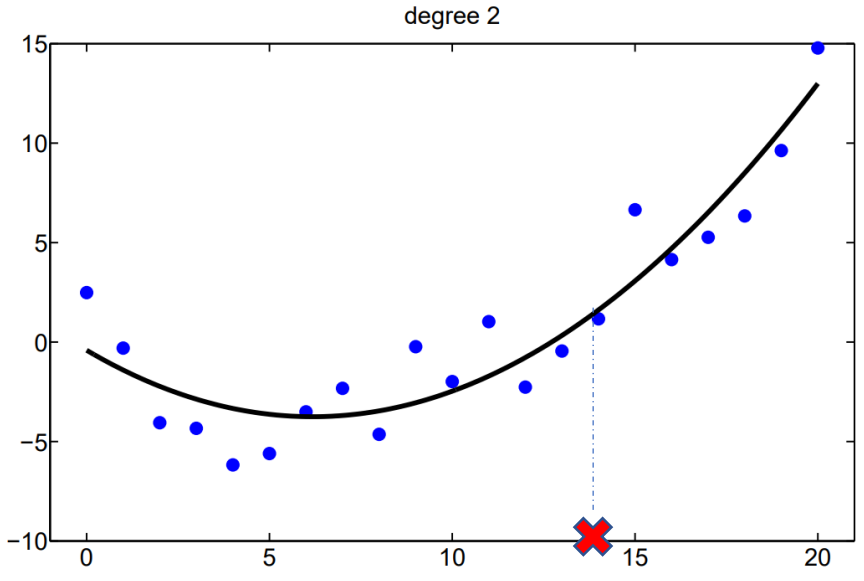
$$\hat{\boldsymbol{y}} = \hat{\boldsymbol{f}}(\mathbf{x})$$

- It is similar to the classification problem, but $y_i \in \mathbb{R}$
- Probabilistic view

$$\hat{\boldsymbol{y}} = \hat{\boldsymbol{f}}(\mathbf{x}) = \underset{y}{\operatorname{argmax}}\, \boldsymbol{p}(\boldsymbol{y}|\mathbf{x}, \boldsymbol{\mathcal{D}})$$

- Linear regression





Unsupervised Learning

- We are just given output data, without any inputs
- We formalize our task as a **density estimation problem**

$$\boldsymbol{p(\mathbf{x}_i|\boldsymbol{\theta})}$$

- It is an **unconditional density estimation** problem
- $\mathbf{x}_i$ is a vector of features, so we need to create **multivariate probability models**

Discovering Clusters

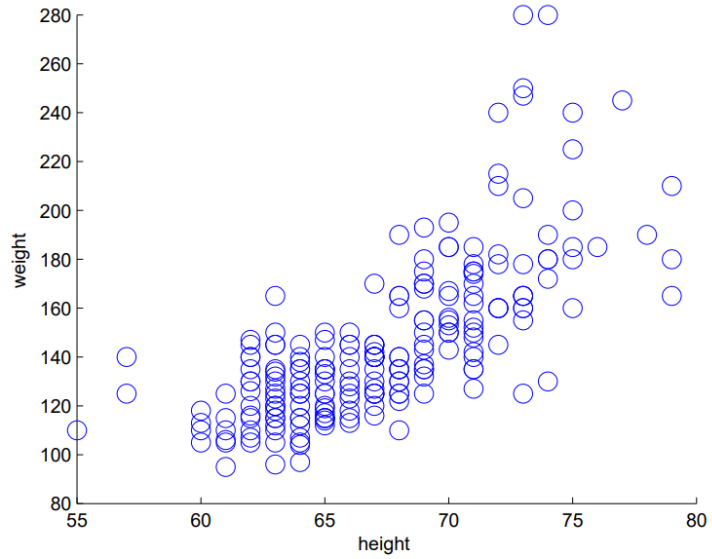
- **Clustering**

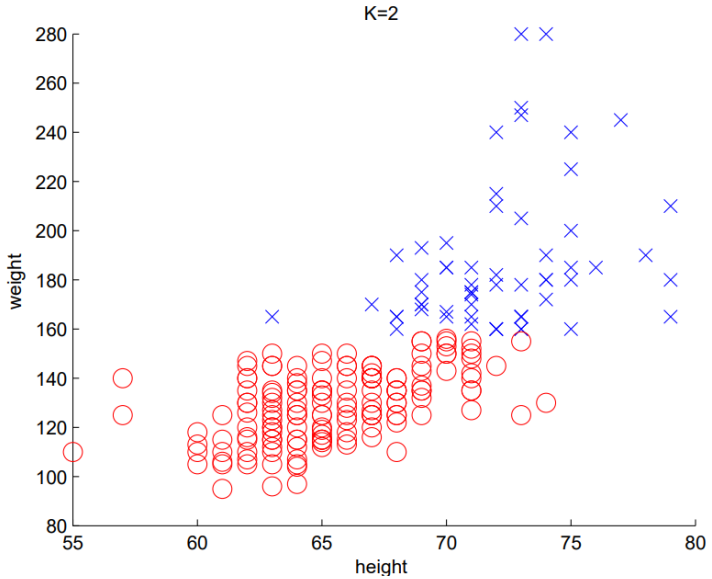
$$K^* = \underset{K}{\operatorname{argmax}}\, p(K|\mathcal{D})$$

Picking the "right" complexity – or number of clusters here – is called "model selection"

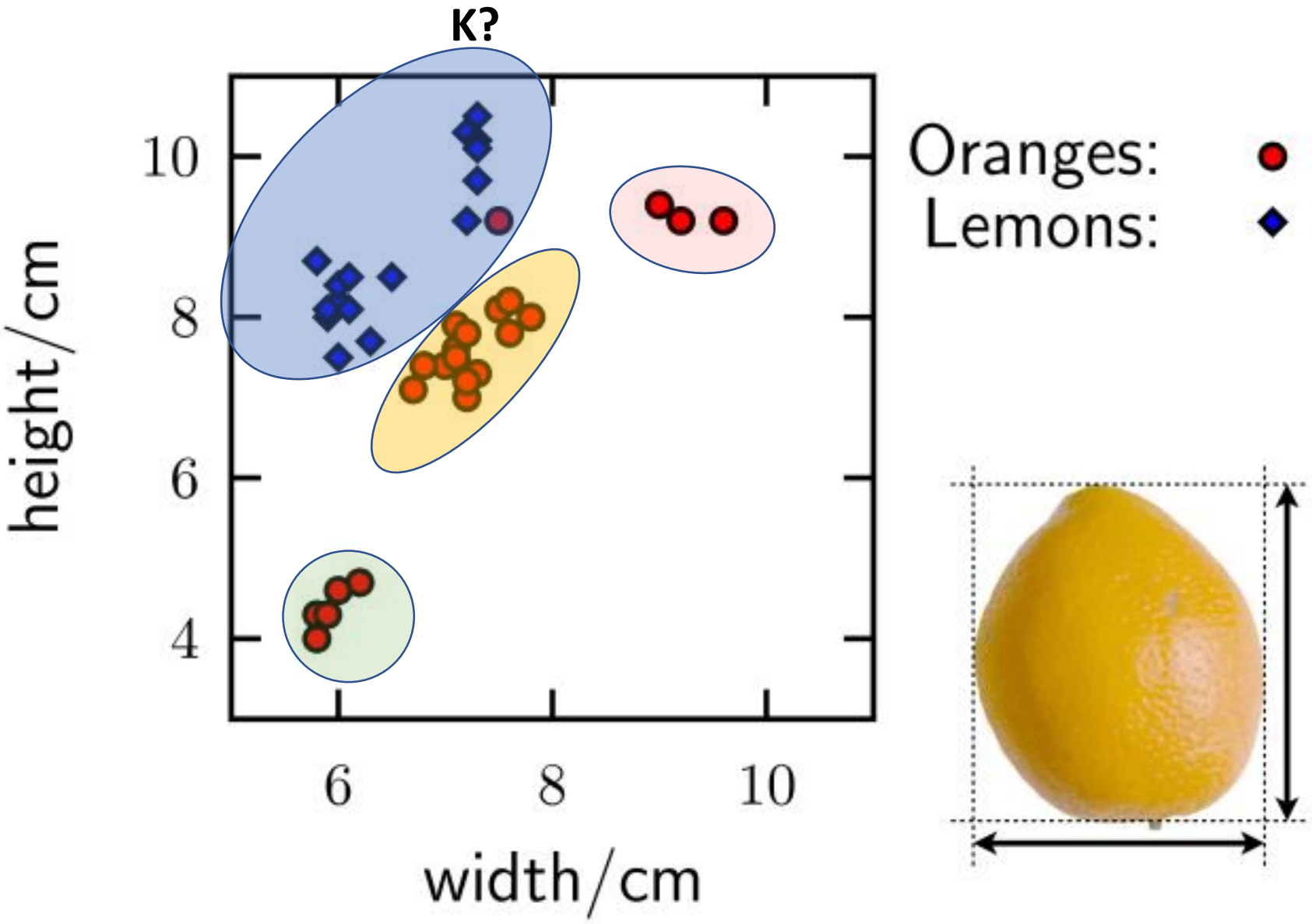
- **Cluster assignment** $z_i = \{1, \cdots, K\}$

$$z_i^* = \underset{K}{\operatorname{argmax}}\, p(z_i = k|\mathbf{x}_i, \mathcal{D})$$



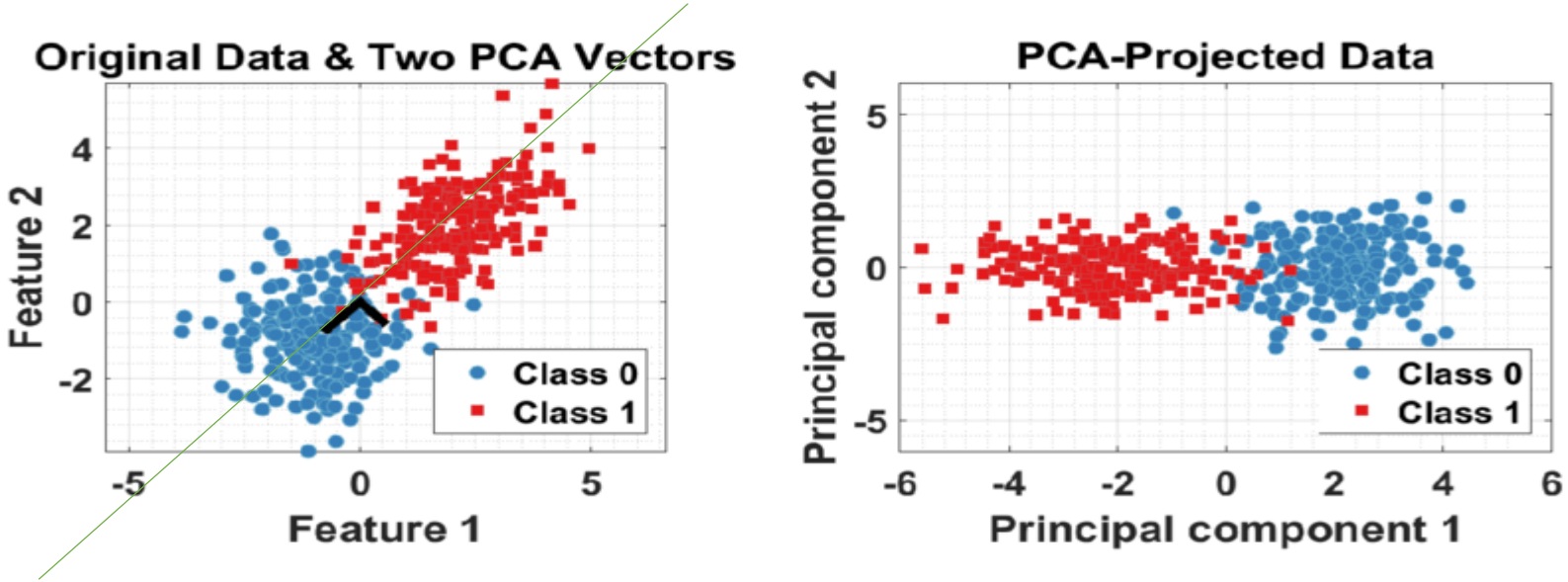


Cluster Analysis



Discovering Latent Factors

- In high dimensional data analysis, it is often useful to reduce the dimensionality by projecting the data to a lower dimensional subspace. **Dimensionality reduction**
- **Latent factors**, a small number of degree of variability that explain high dimensional data
- **Principal Components Analysis (PCA)** is the most common dimensionality reduction technique

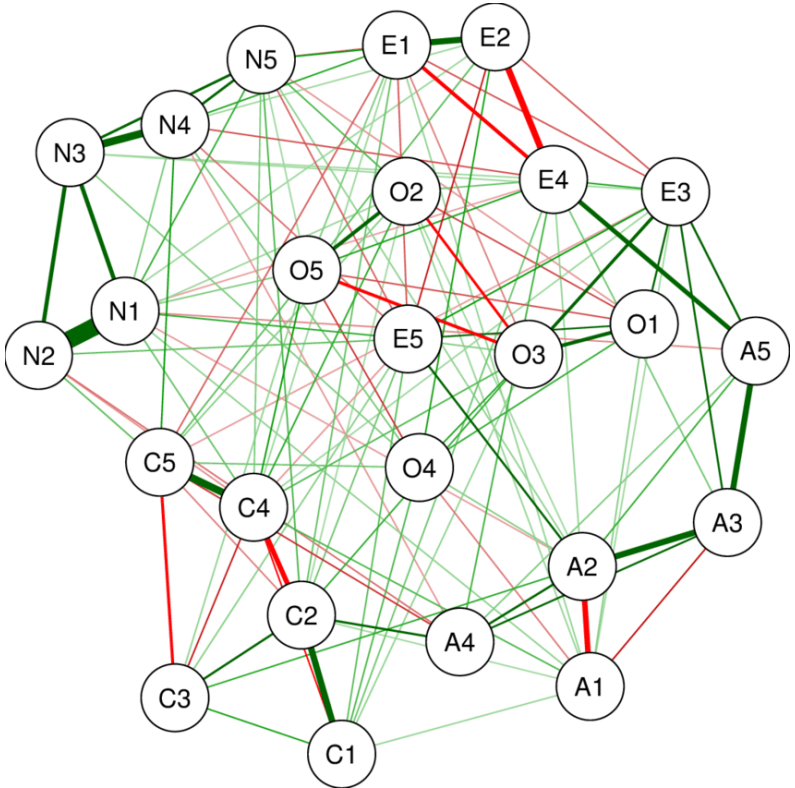


Discovering Graph Structure

- Graph structures in high dimensional data

$$\hat{G} = \operatorname{argmax} p(G|\mathcal{D})$$

- In a graph G, nodes represent variables and edges represent dependence between variables



Some basic concepts in ML

- **Parametric** vs non-parametric models
- A **parametric model** is a family of probabilistic models – $p(y|\mathbf{x})$ or $p(\mathbf{x})$ - that has a finite number of parameters.
- In non-parametric models, the number of parameters grow with the amount of training data
- Parametric models have the advantage of often being faster to use, but the disadvantage of making stronger assumptions about the nature of the data distributions.
- Nonparametric models are more flexible, but often computationally intractable for large datasets

Non-parametric classifier: K-nearest neighbors

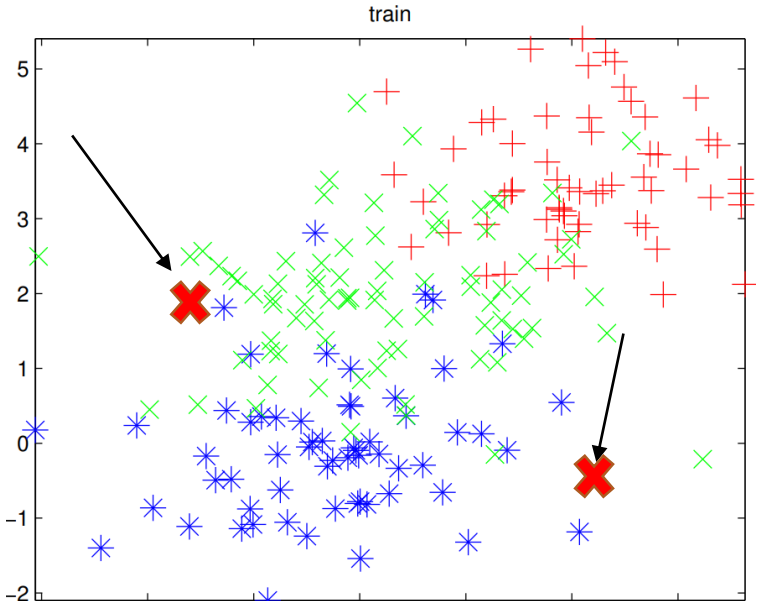
- **KNN** looks at the *K points in the training set that are nearest to the test input x, counts how many members of each class are in this set, and returns that empirical fraction as the estimate*

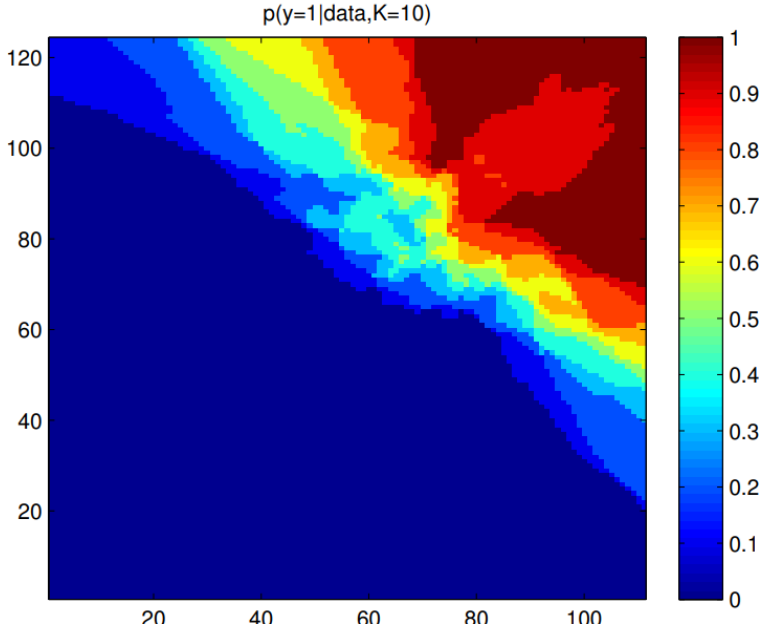
$$p(\hat{y} = c|\mathbf{x}, \mathcal{D}, K) = \frac{1}{K} \sum_{i \in N_K(\mathbf{x}, \mathcal{D})} \mathbb{I}(y_i = c)$$

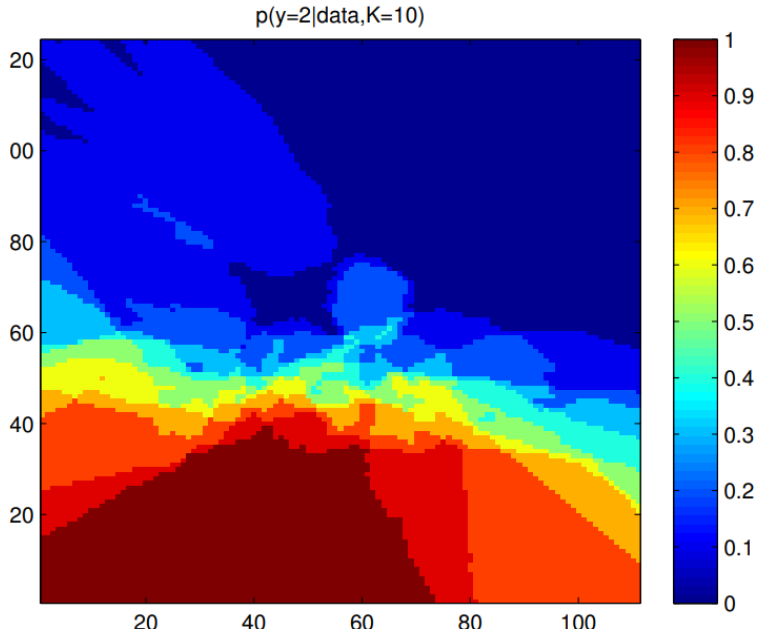
$$\mathbb{I}(y_i = c) = \begin{cases} 1 & \text{if } e \text{ is true} \\ 0 & \text{if } e \text{ is false} \end{cases}$$

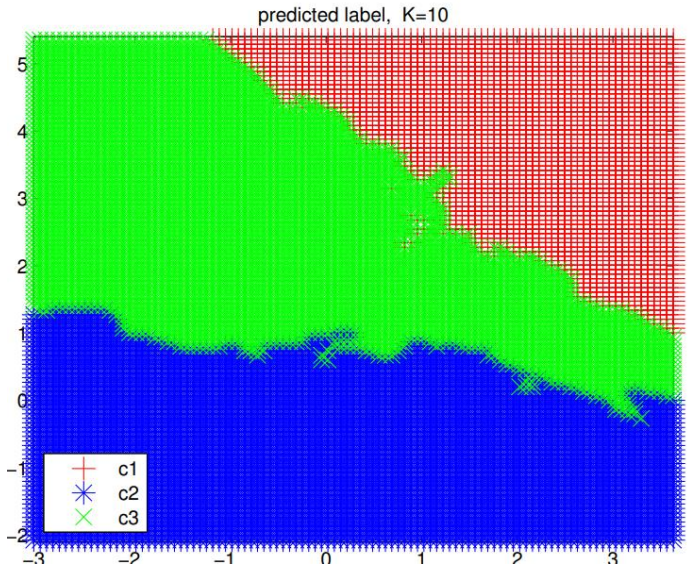
$N_K(\mathbf{x}, \mathcal{D})$ *are the (indices of the)* K *nearest points to x in* $\mathcal{D}$

KNN classifier



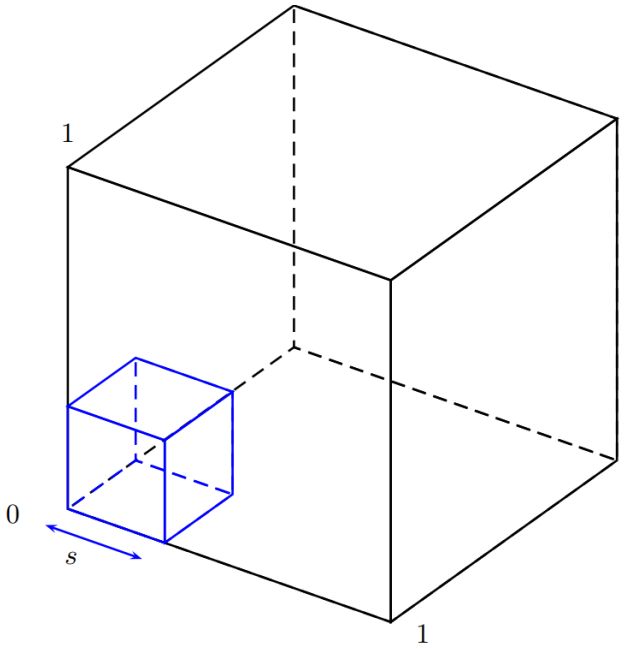


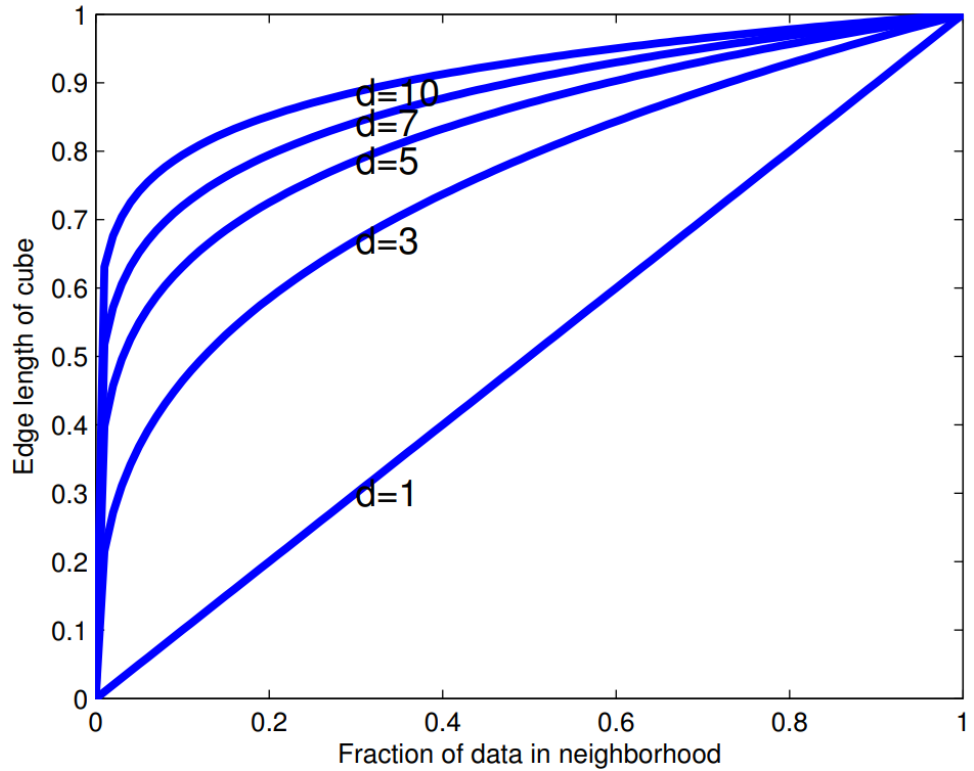




The curse of dimensionality

- The KNN classifier is simple and can work quite well, provided it is given a good distance metric and has enough labeled training data.
- The main problem with KNN classifiers is that they do not work well with high dimensional inputs. The poor performance in high dimensional settings is due to **the curse of dimensionality**





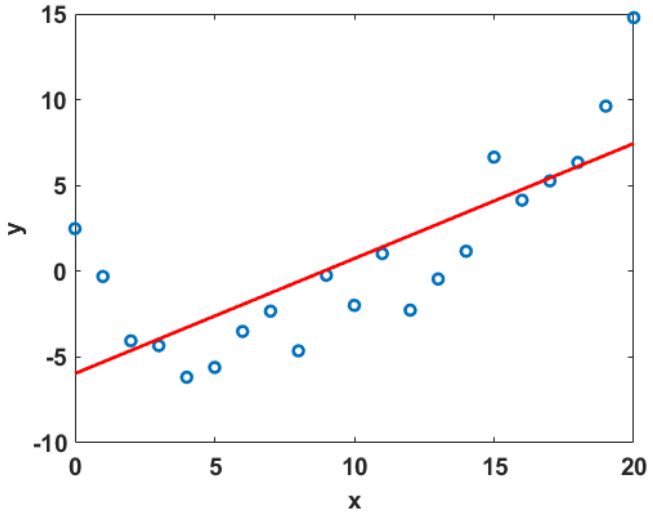
Parametric models for classification and regression

- The main way to combat the curse of dimensionality is to make some assumptions about the nature of the data distribution.
- This assumption is embodied in the form of a **parametric model**, which is a statistical model with a fixed number of parameters.
- Linear regression

pdf

$$y(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x} + \varepsilon = \sum_{j=1}^{D} w_j x_j + \varepsilon$$

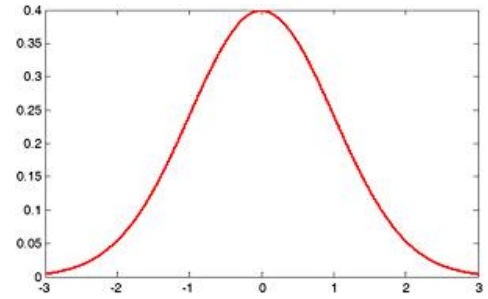
- If we assume that ε has a Gaussian or normal distribution - $\varepsilon \sim \mathcal{N}(0, \sigma^2)$



$$p(y|\mathbf{x}, \theta) = \mathcal{N}(\mu(\mathbf{x}), \sigma^2)$$

$$\mu(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x}$$

$$\theta = \{\sigma^2, \mathbf{w}\}$$

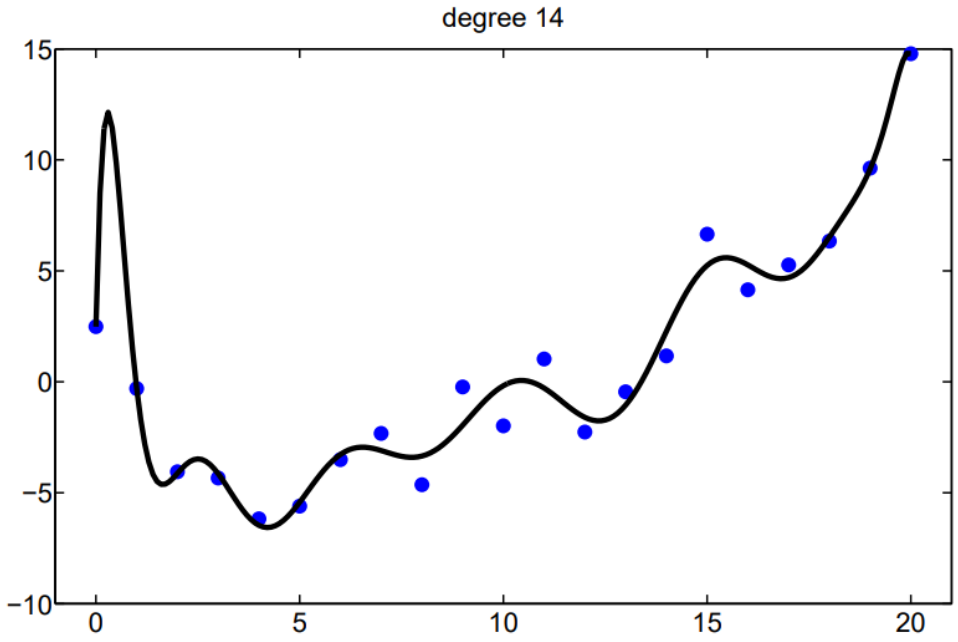


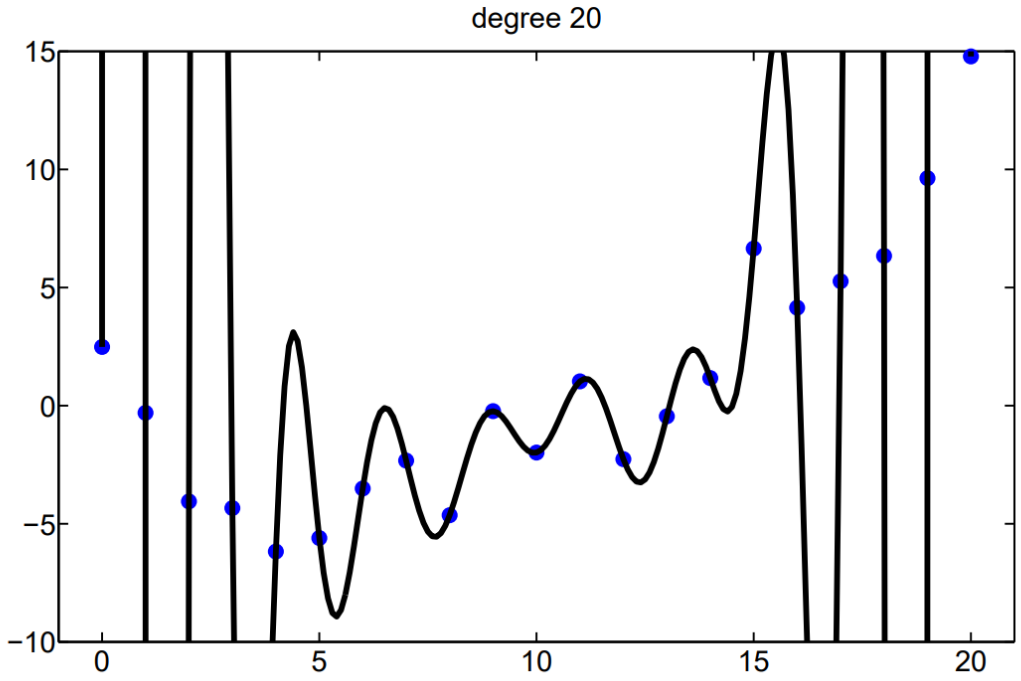
Polynomial regressions

$$p(y|\mathbf{x}, \theta) = \mathcal{N}(\mu(\mathbf{x}), \sigma^2)$$

$$\mu(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\boldsymbol{\phi}(\mathbf{x})$$

$\boldsymbol{\phi}(x) = \{1, x, x^2\}$ – we assume x is a scalar variable





Logistic Regression

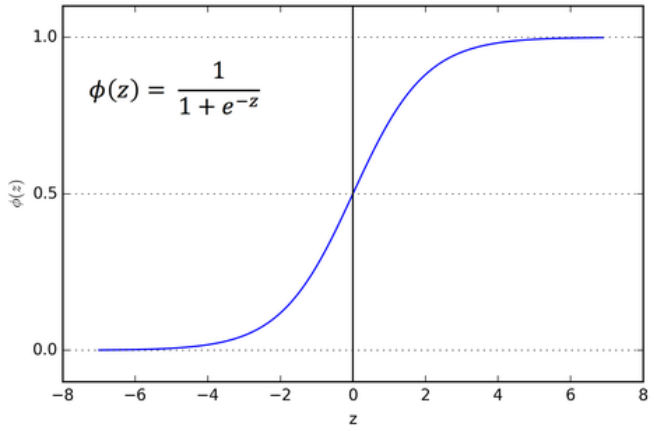
- We can generalize linear regression to binary classification

$$p(y|\mathbf{x}, \mathbf{w}) = Ber(y|\mu(\mathbf{x}))$$

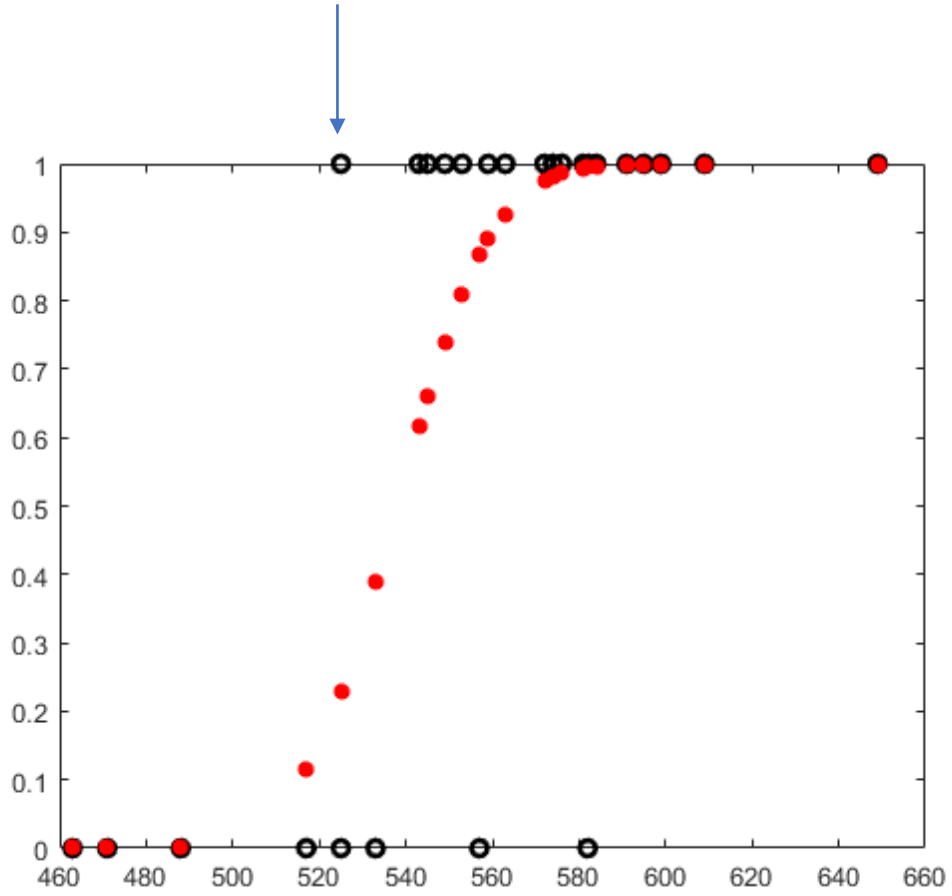
$$\mu(\mathbf{x}) = \mathrm{E}[\mathrm{y}|\mathbf{x}] = \mathbf{w}^{\mathbf{T}}\mathbf{x} = p(y = 1|\mathbf{x})$$

$$\mu(\mathbf{x}) = \mathrm{sigm}\left(\mathbf{w}^{\mathbf{T}}\mathbf{x}\right)$$

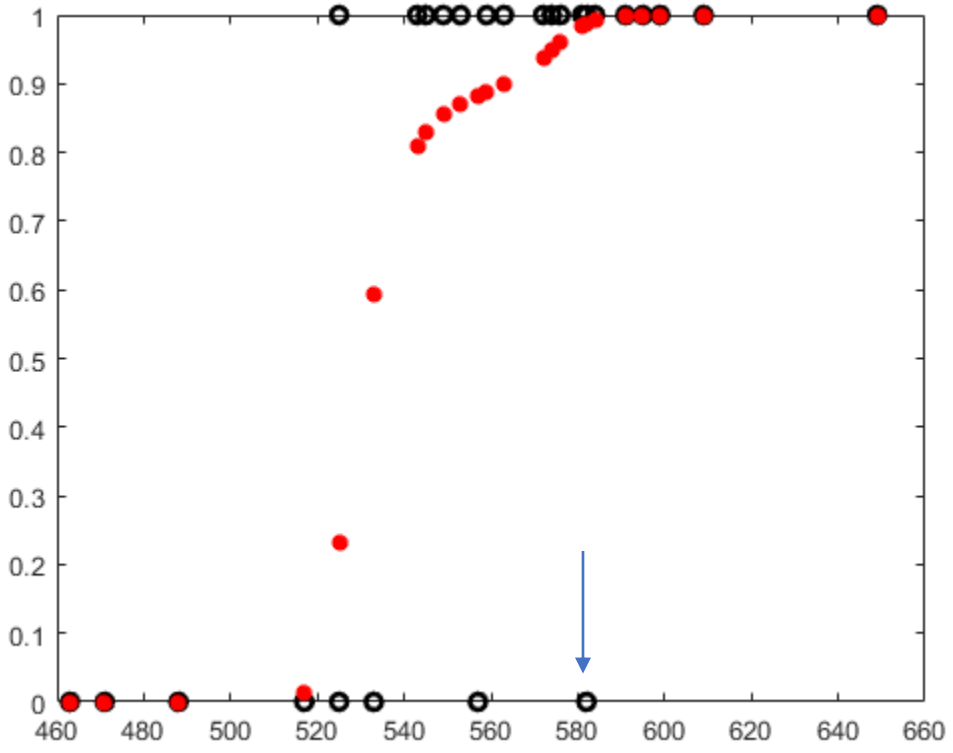
$$p(y|\mathbf{x}, \mathbf{w}) = Ber(y|\mathrm{sigm}\left(\mathbf{w}^{\mathbf{T}}\mathbf{x}\right))$$



Logistic Regression



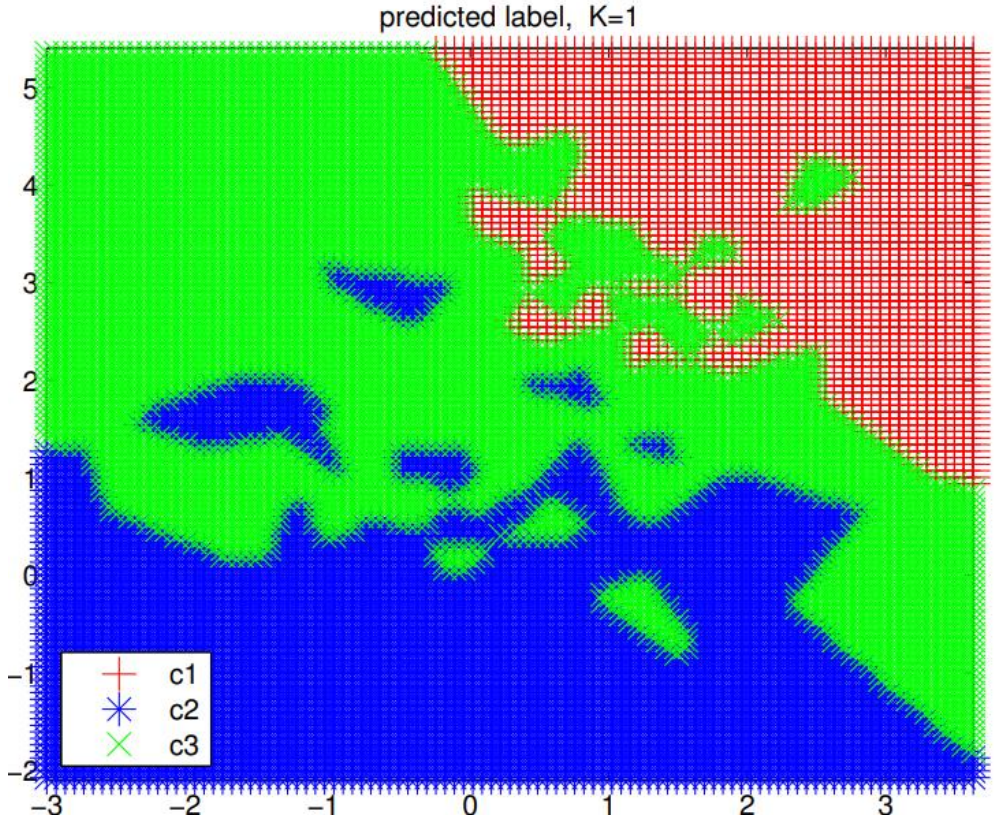
$\mathbf{x} = \{1, x\}$

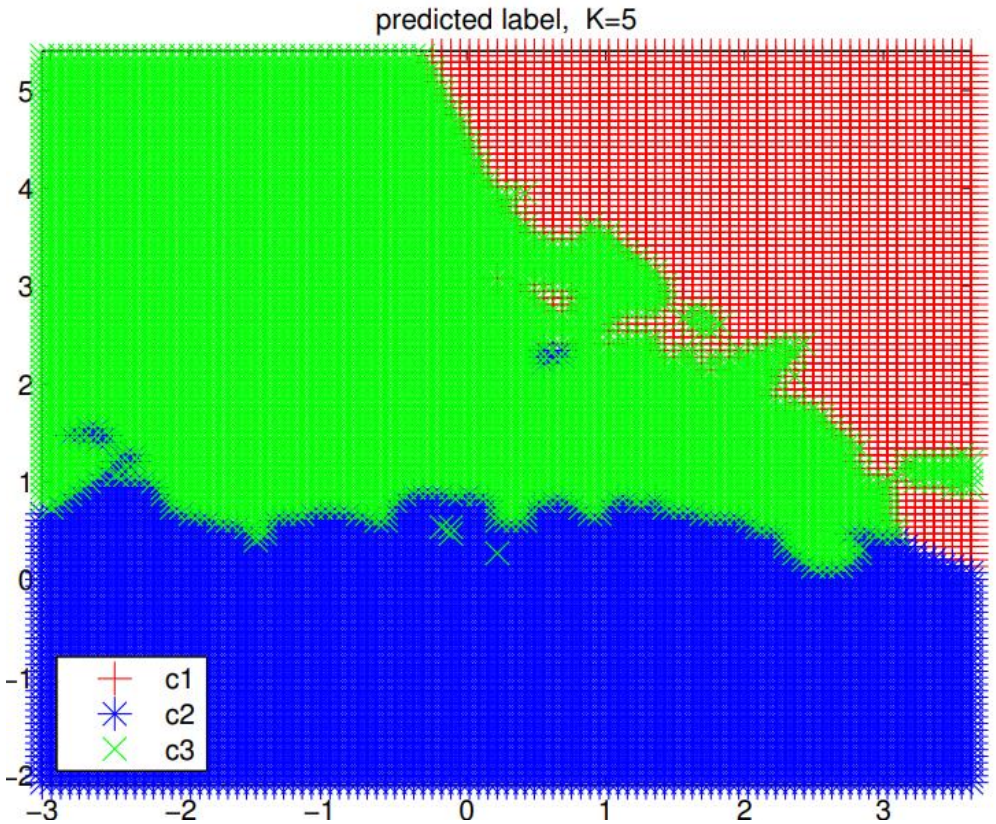


$\mathbf{x} = \{1, x, x^2, x^3\}$

Over fitting issue

- **Overfitting** refers to a model that models the training data too well. **Overfitting** happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.



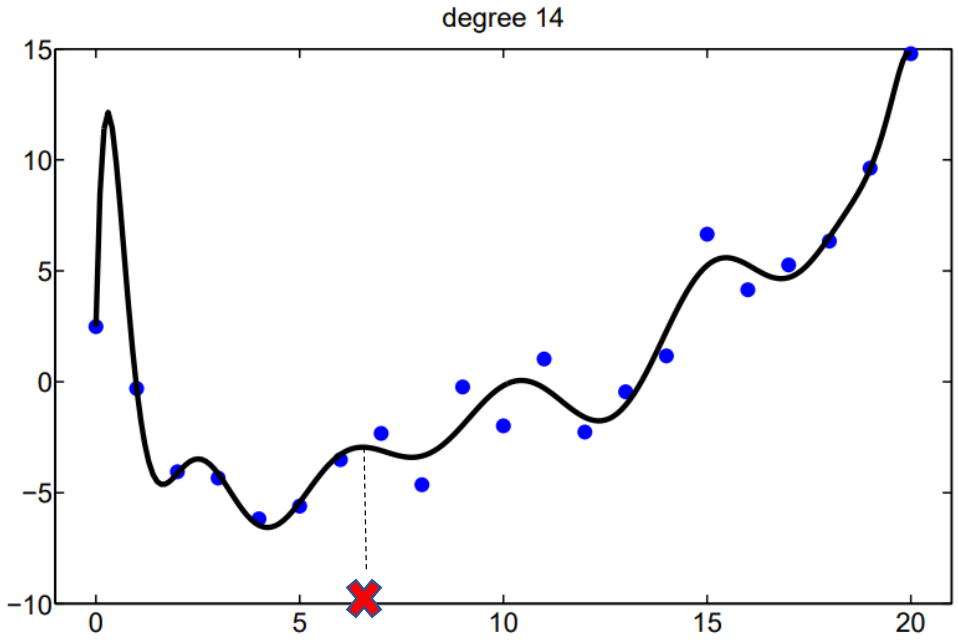


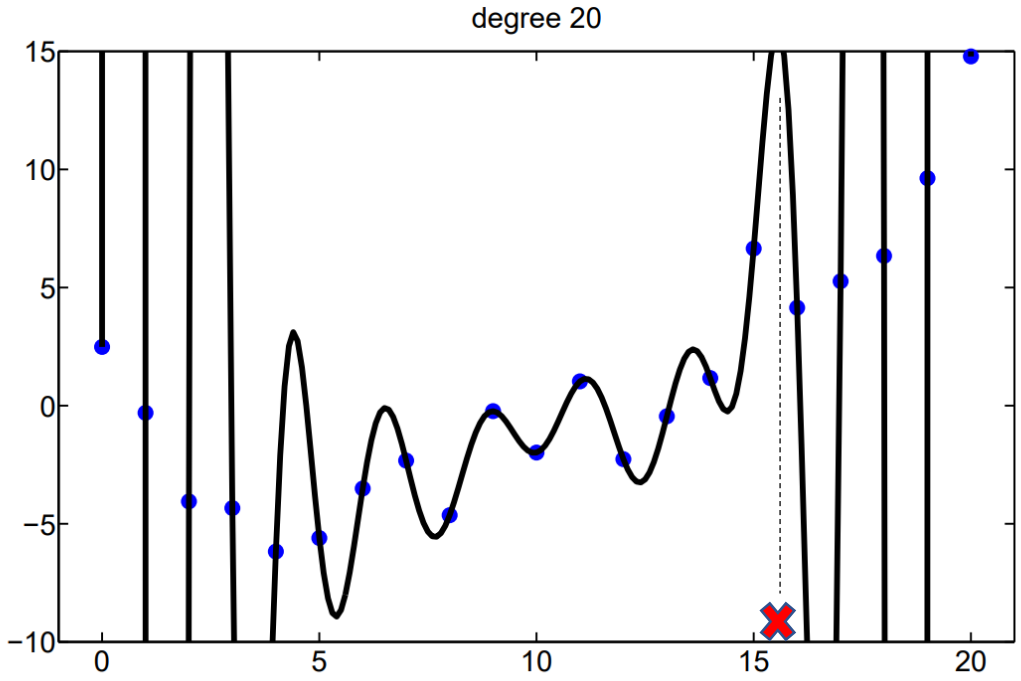
Polynomial regressions

$$p(y|\mathbf{x}, \theta) = \mathcal{N}(\mu(\mathbf{x}), \sigma^2)$$

$$\mu(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\boldsymbol{\phi}(\mathbf{x})$$

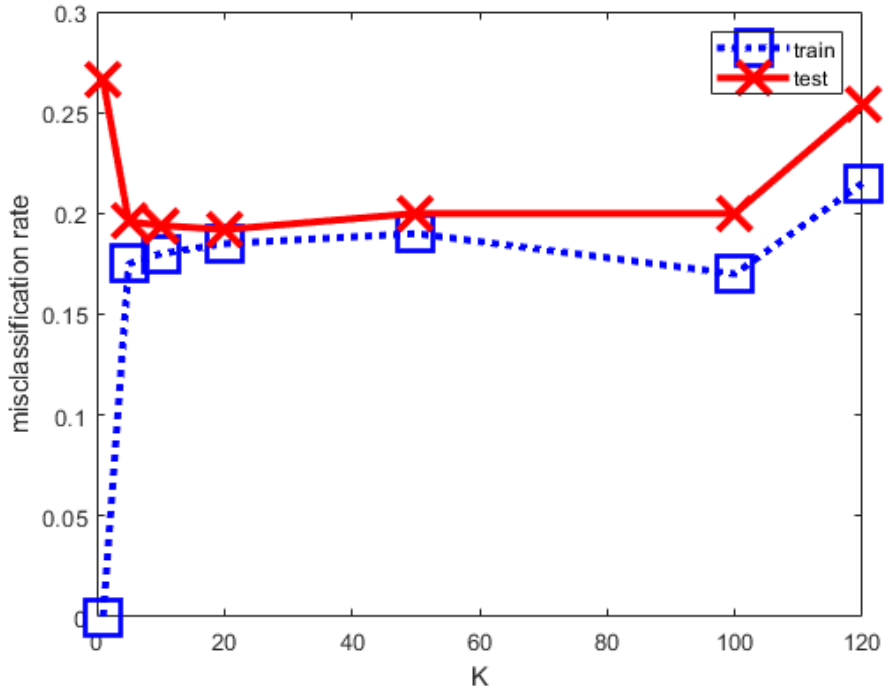
$\boldsymbol{\phi}(x) = \{1, x, x^2\}$ – we assume x is a scalar variable

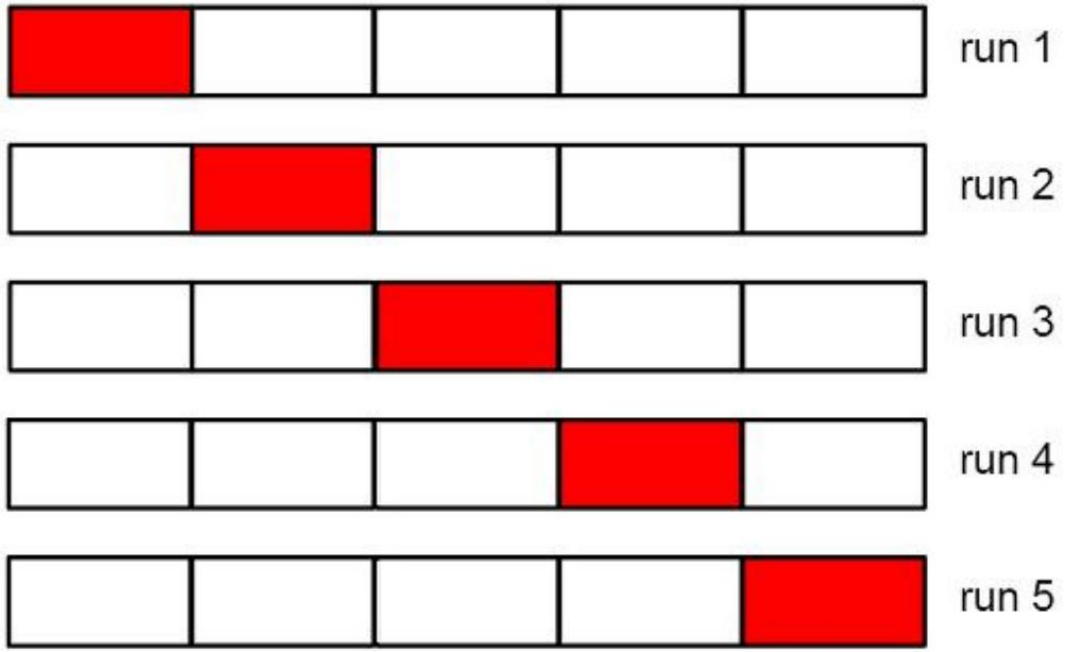




Cross validation

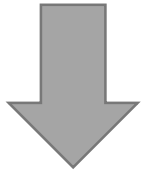
- Wesplit the training data into K folds; then, for each fold $k \in \{1, \ldots, K\}$, we train on all the folds but the k^{th}, and test on the k^{th}, in a round-robin fashion
- It is common to use $K = 5$, which corresponds to 80% of the data for the training set and 20% for the test – or validation – set



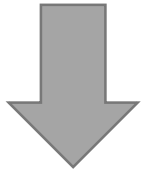


No Free Lunch Theorem

Our model is a simplification of reality



Simplification is based on assumptions (model bias)



Assumptions fail in certain situations

Roughly speaking:

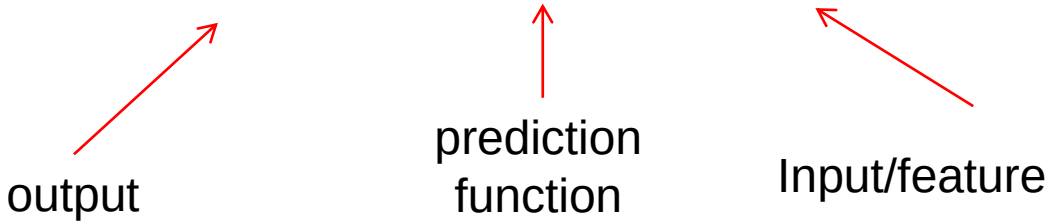
“No one model works best for all possible situations.”

Machine Learning Buzzwords

- Bayesian and frequentist estimation: MAP and MLE
- Model selection, cross-validation, overfitting
- Linear least squares regression, logistic regression
- Robust statistics, sparsity, L1 vs. L2 regularization
- Features and kernel methods: support vector machines (SVMs), Gaussian processes
- Graphical models: hidden Markov models, Markov random fields, efficient inference algorithms
- Expectation-Maximization (EM) algorithm
- Markov chain Monte Carlo (MCMC) methods
- Mixture models, PCA & factor analysis, manifolds
- Deep Neural Network, CNN, LSTM, GAN, Variational autoencoder

The ML Framework

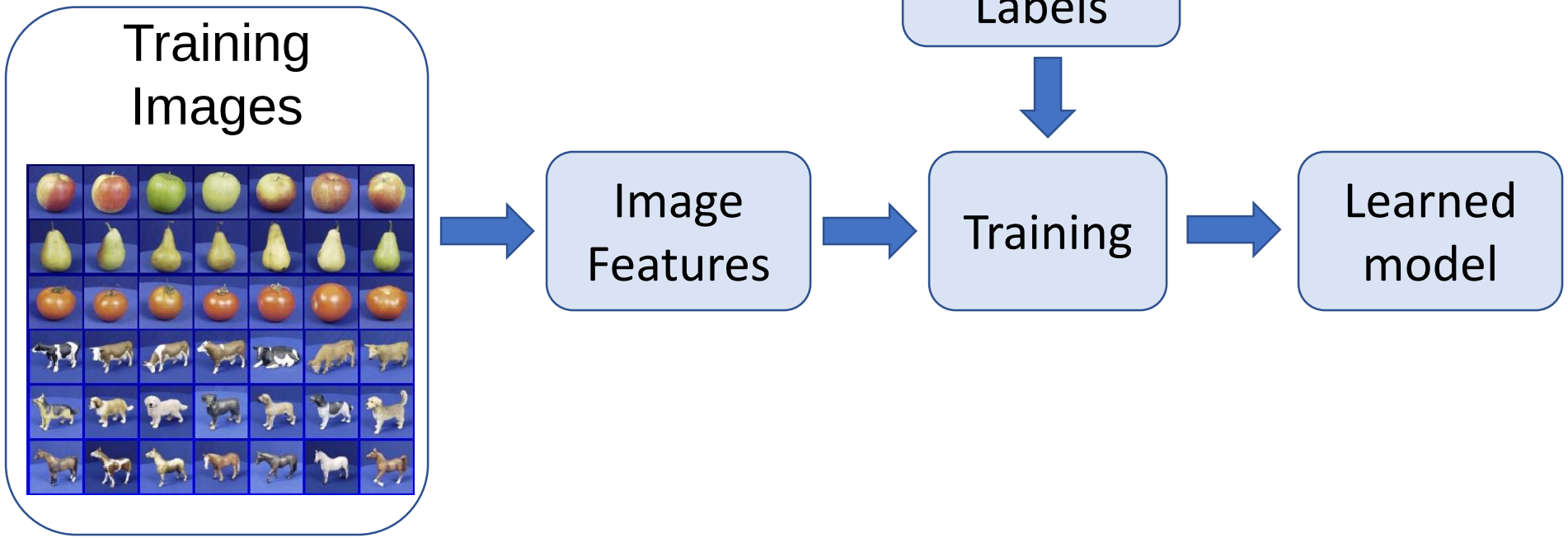
$$y = f(x)$$



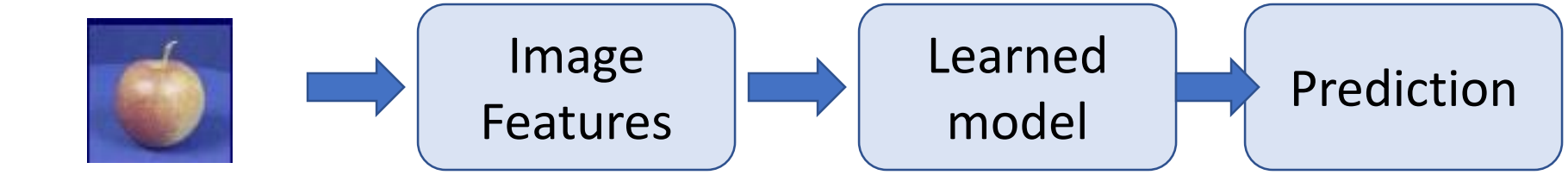
- **Training:** given a *training set* of labeled examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, estimate the prediction function f by minimizing the prediction error on the training set
- **Testing:** apply f to a never before seen *test example* $\mathbf{x}$ and output the predicted value $y = f(\mathbf{x})$

Modeling Steps

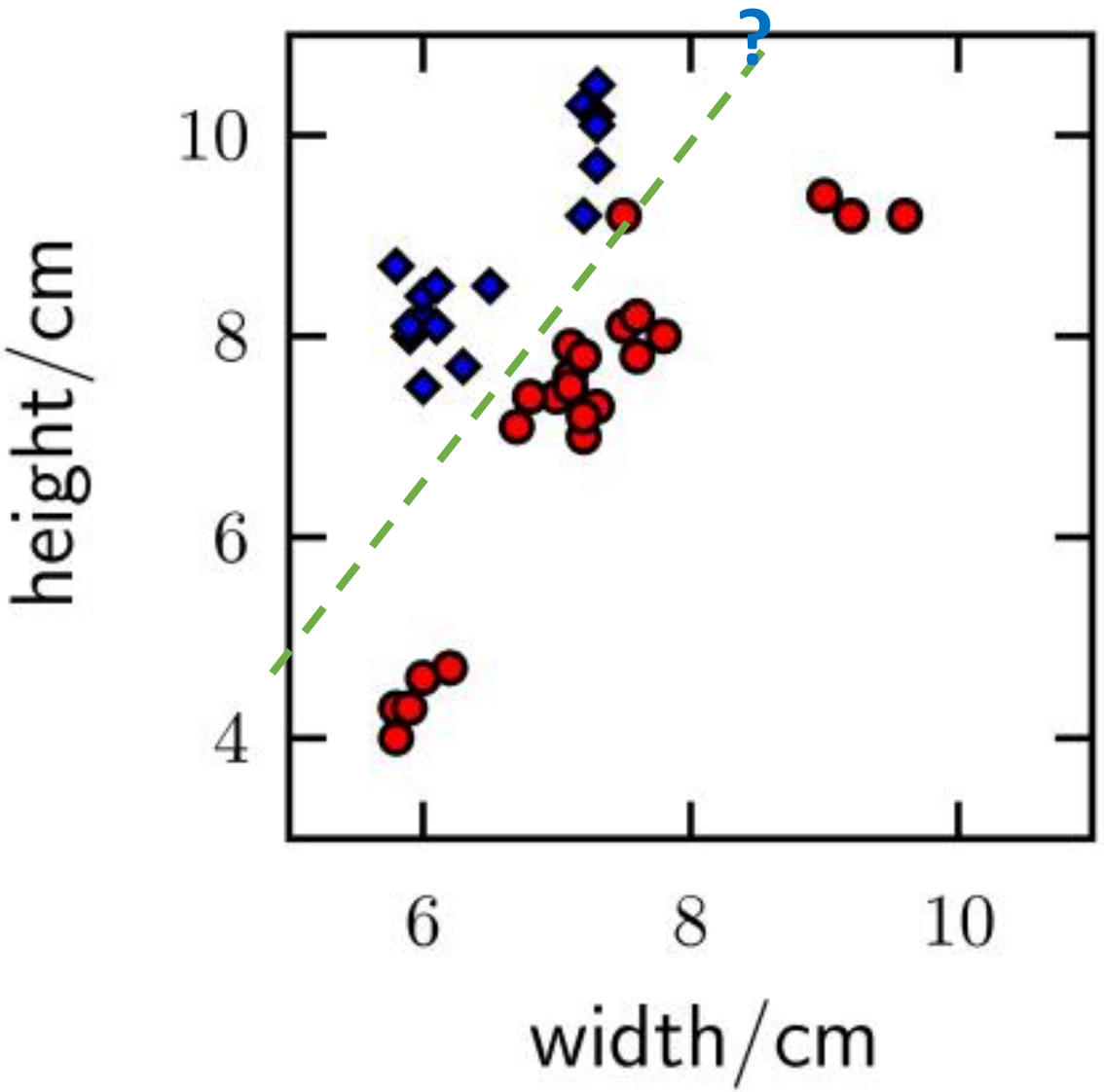
Training



Testing

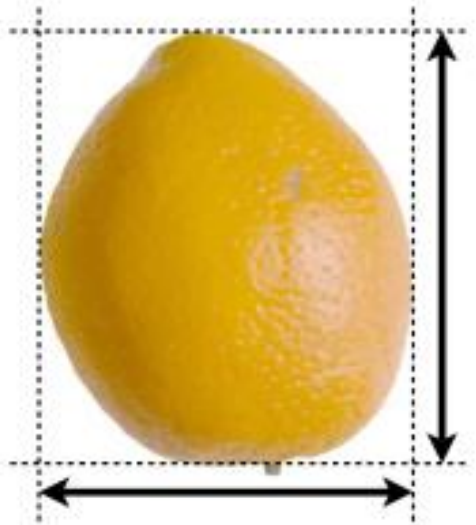


A Linear Classifier



Oranges:

Lemons:



Introduction to probability