Wolkite University

College of Computing and Informatics

Department of Computer Science

Chapter Six : Learning Agents

Prepared by Adem (MSc.)

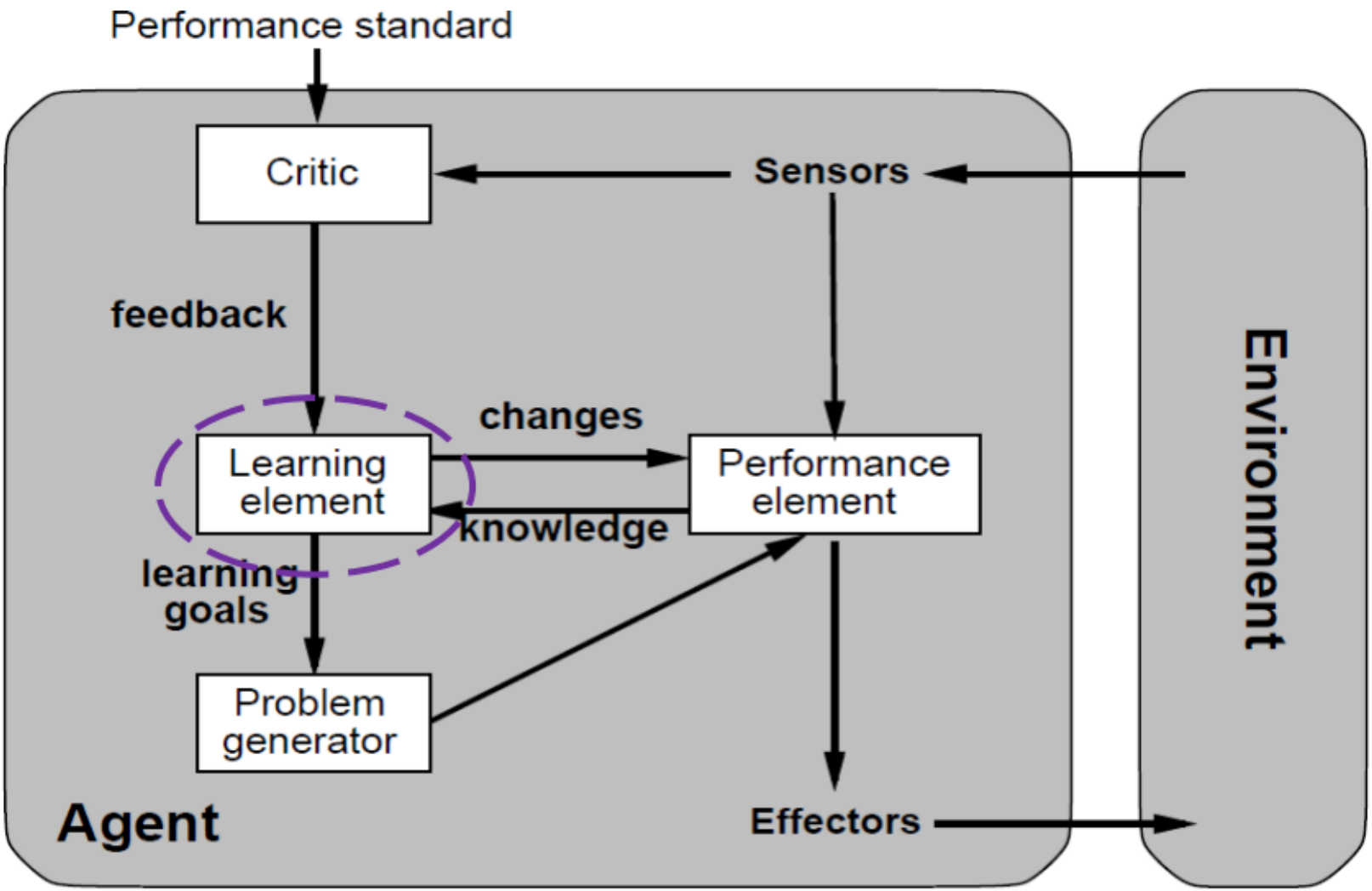
August 15, 2022

Outline

Knowledge in Learning Agents

- A distinct feature of intelligent agents in nature is
  - their ability to learn from experience
- Using his experience and his internal knowledge,
  - a learning agent is able to produce new knowledge
- That is, given his internal knowledge and a percept sequence,
- the agent is able to learn facts that
  - are consistent with both the percepts and the previous knowledge,
  - do not just follow from the percepts and the previous knowledge

Learning Agent: Conceptual Components



Cont...

- Machine learning research has produced a large variety of learning elements
- Major issues in the design of learning elements:
  - Which components of the performance element are to be improved
  - What representation is used for those components
  - What kind of feedback is available:
    - Supervised learning
    - Reinforcement learning
    - Unsupervised learning
  - What prior knowledge is available

Cont...

- Any component of a performance element can be described mathematically as a function:
  - condition-action rules
  - predicates in the knowledge base
  - next-state operators
  - goal-state recognizers
  - search heuristic functions
  - belief networks
  - utility functions
- All learning can be seen as learning the representation of a function

Cont...

- Up until now: hand-craft algorithms to make rational/optimal or at least good decisions. Examples: Search strategies, heuristics.
- Learning:
  - Improve performance after making observations about the world.
  - That is, learn what works and what doesn't.
- Machine learning: how to build a model from data/experience
  - Supervised Learning: Learn a function to map input to output. Examples:
  - Use a naïve Bayesian classifier to distinguish between spam/no spam
  - Learn a playout policy to simulate games (current board -> good move)
  - Reinforcement Learning: Learn from rewards/punishment (e.g., winning a game).

Cont...

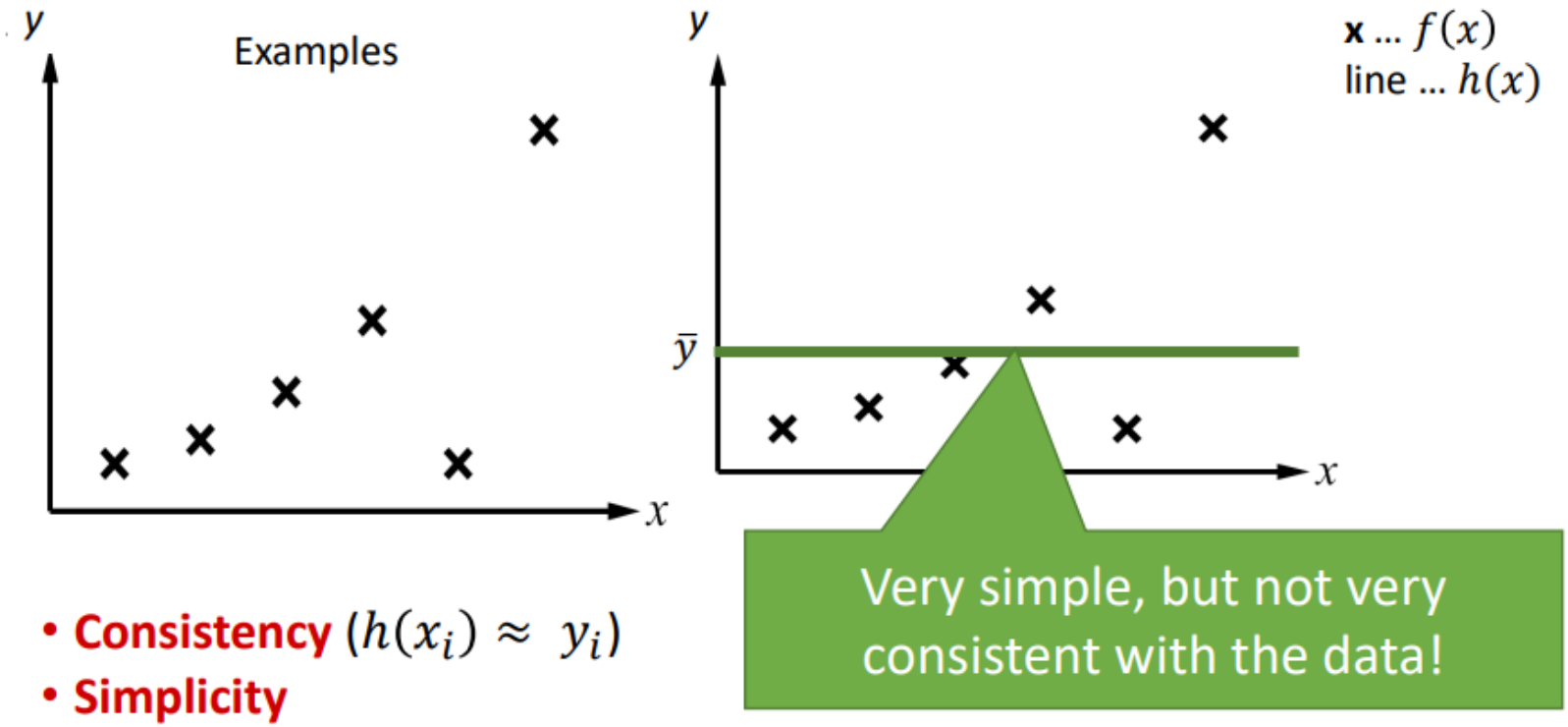
- Learning vs. hard coding the agent function
- Designer cannot anticipate all possible future situations.
- Designer may have examples but does not know how to program a solution.

Supervised Learning

- Examples
  - Input-output pairs: E = X1 , Y1 ,... , xi , yi,...., Xn , Yn .
  - We assume that the examples are produced (with noise and errors) from a target function Y = f(x).
- Learning problem
  - Given a hypothes h is space H
  - Find a hypothesis h ∈ H such that y = h(xi) approximate to yi
  - That is, we want to approximate f by h using E.
- Includes
  - Classification (outputs = class labels). E.g. x is an email and f(x) is spam / ham
  - Regression (outputs = real numbers). E.g. x is a house and f(x) is its selling price

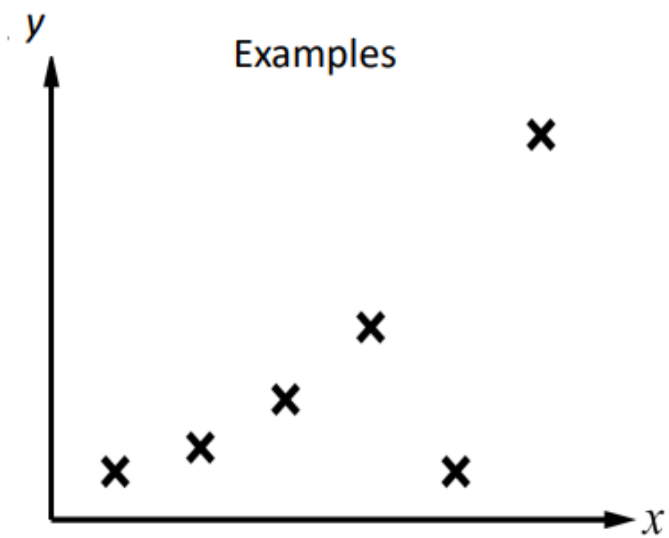
Consistency vs. Simplicity

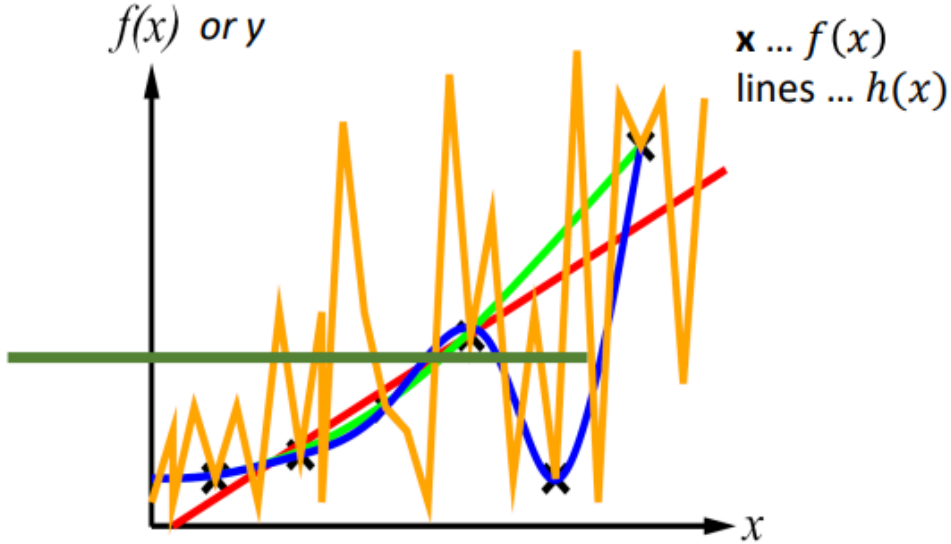
- Example: Curve fitting (regression, function approximation)



Cont...

- Example: Curve fitting (regression, function approximation)





- **Consistency** $(h(x_i) \approx y_i)$
- **Simplicity**

Consistency and Loss

- Goal of learning: Find a hypothesis that makes good predictions that are consistent with the examples E =

$$(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_N, y_N).$$

That is, $\hat{y} = h(x) \approx y.$

- Measure mistakes: Loss function L(Y, Y')

Absolute-value loss	$L_1(y, \hat{y}) = \lvert y - \hat{y} \rvert$	For Regression
Squared-error loss	$L_2(y, \hat{y}) = (y - \hat{y})^2$	
0/1 loss	$L_{0/1}(y, \hat{y}) = 0$ if $y = \hat{y}$, else 1	For Classification

- Empirical loss: average loss over the N examples in the dataset

$$EmpLoss_{L,E}(h) = \frac{1}{|E|} \sum_{(x,y) \in E} L(y, h(x))$$

Example: Bayes Classifier

- For 0/1 loss, the empirical loss is minimized by the model that predicts for each x the most likely class y.

$$h(x)^* = \underset{y}{\operatorname{argmax}}\, P(Y = y \mid X = x)$$

- This is equivalent (by Bayes' rule and dropping of P(x)) to:

$$h(x)^* = \underset{y}{\operatorname{argmax}}\, P(X = x \mid Y = y)P(y) = \underset{y}{\operatorname{argmax}}\, P(X, y)$$

- The Bayes Classifier is optimal and guarantees the lowest possible error rate called the Bayes error rate.
- Issue: Needs the complete joint probability which requires in the general case a probability table with one entry for each possible value for x.
- That is why we often use the naïve Bayes classifier, which is not optimal.

Simplicity

- Ease of use: Simpler hypotheses have fewer parameters and are easier to estimate.
- Generalization: How well does the hypothesis perform on new data?
  - We do not want the model to be too specific to the training examples (an issue called overfitting).
  - Simpler models typically generalize better to new examples.
- How to achieve simplicity?
  - Restrict H to simple models (e.g., linear models)
  - Feature selection (use fewer variables)
  - Regularization (penalize for complexity)
-

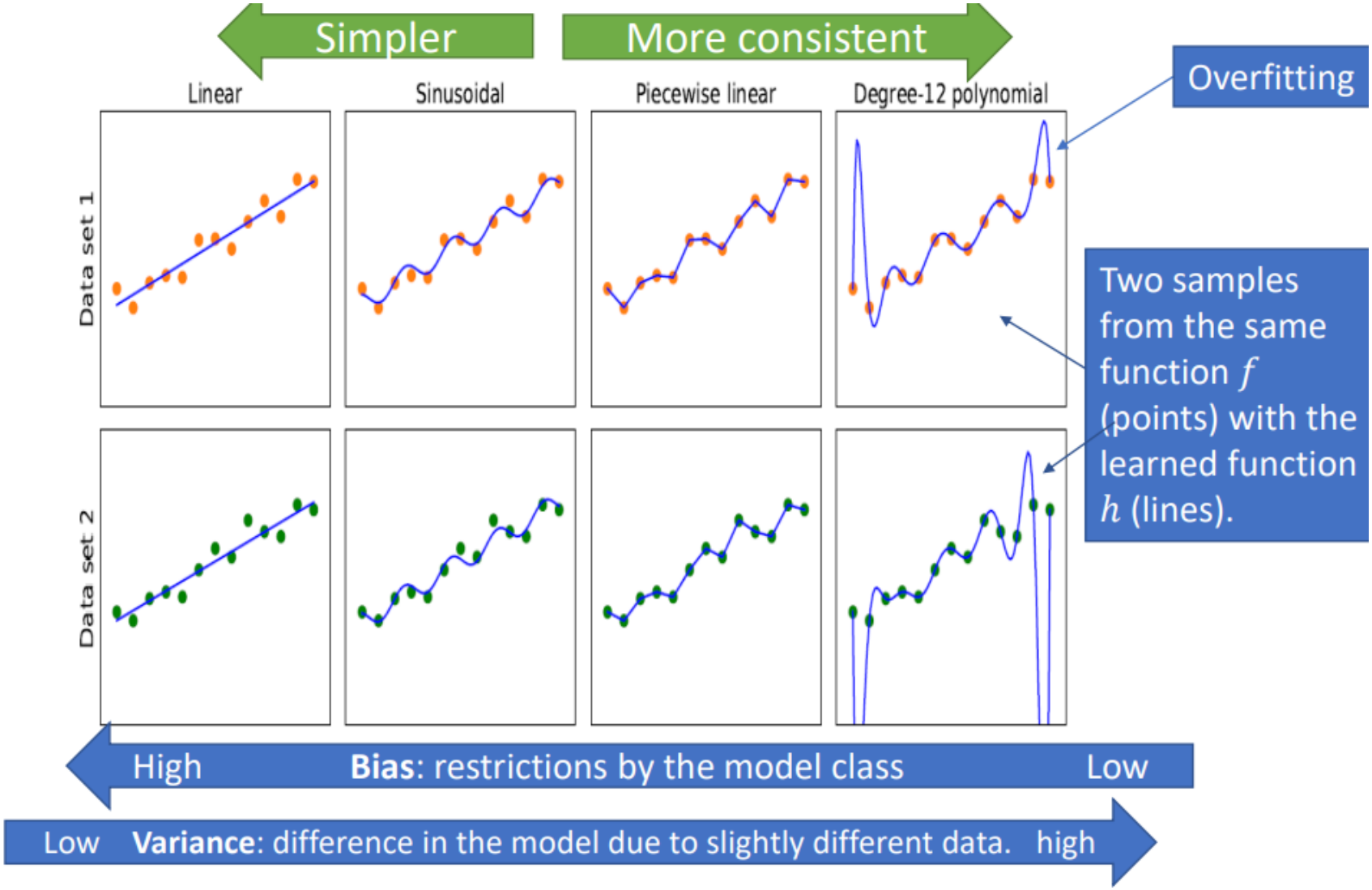
$$h^* = \underset{h \in H}{\operatorname{argmin}} \left[EmpLoss_{L,E}(h) + \lambda \, Complexity(h) \right]$$

Penalty term

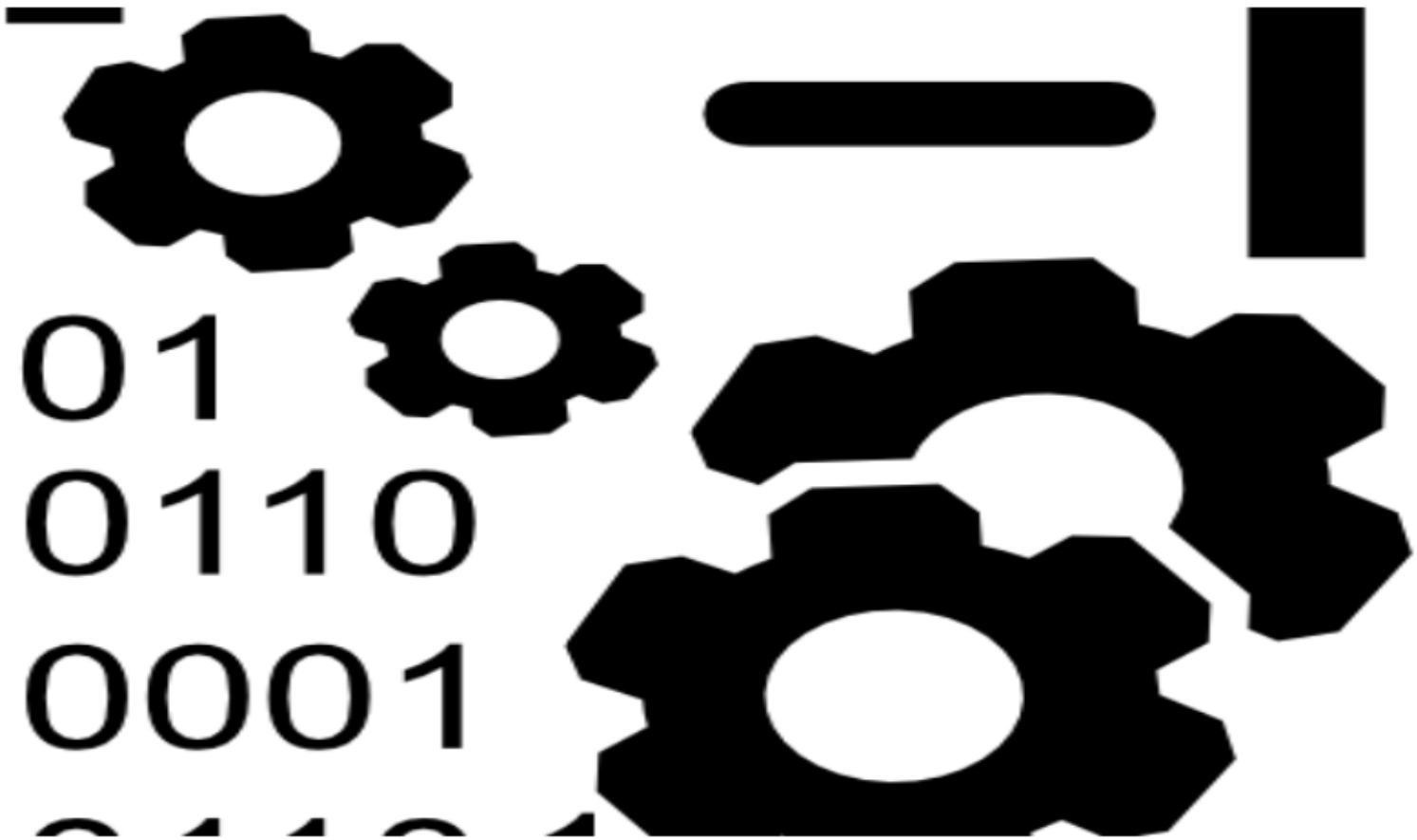
Cont....

- Bais error : the diffrence occur betwwen the prediction value and actual value while predicting
  - low bais : fewer assumptions about the form of target functions
  - high bais : higher assumptions about the form of target functions
    - the model become unable to capture important features
    - high bais model can not perform well on new data
- variance error tells how much a random variable is diffrent from its expected values
  - low variance :
    - small variation in the prediction of target functions with change in the training dataset
  - High variance :
    - high variation in the prediction of target functions with change in the training dataset

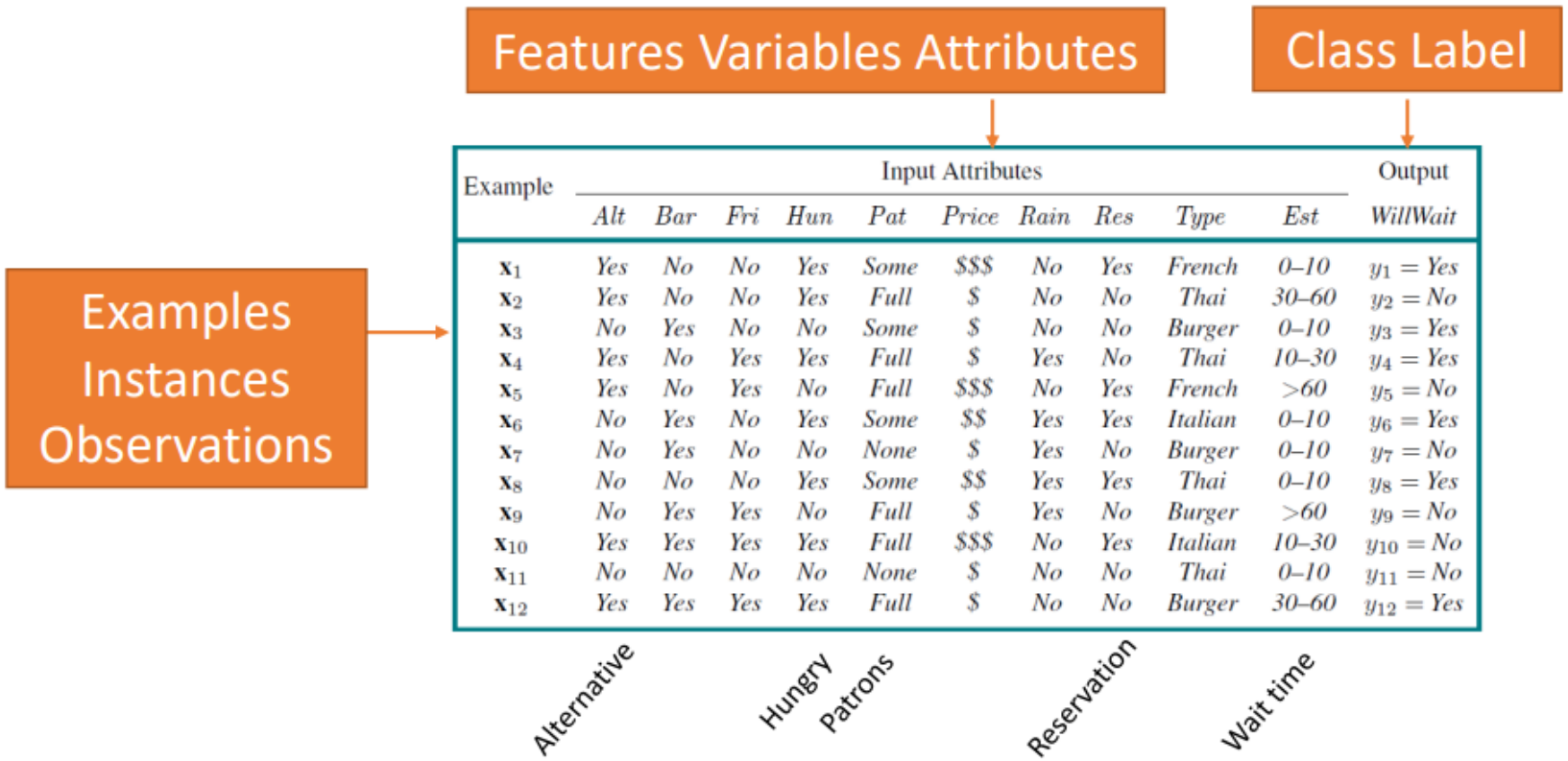
Model Selection: Bias vs. Variance



Data



The Dataset



Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Find a hypothesis (called “model”) to predict the class given the features.

Feature Engineering

- Add information sources as new variables to the model.
- Add derived features that help the classifier
- Example for Spam detection: In addition to words
  - Have you emailed the sender before?
  - Have 1000+ other people just gotten the same email?
  - Is the header information consistent?
  - Is the email in ALL CAPS?
  - Do inline URLs point where they say they point?
  - Does the email address you by (your) name?
- Feature Selection:
  - Which features should be used in the model is a model selection problem (choose between models with different features).

Model Evaluation

- We want to test how well the model will perform on new data (i.e., how well it generalizes).
- Testing loss: Calculate the empirical loss for predictions on a testing data set T that is different from the data used for training.

$$EmpLoss_{L,T}(h) = \frac{1}{|T|} \sum_{(x,y) \in T} L(y, h(x))$$

- For classification we often use the accuracy measure, the proportion of correctly classified test examples.

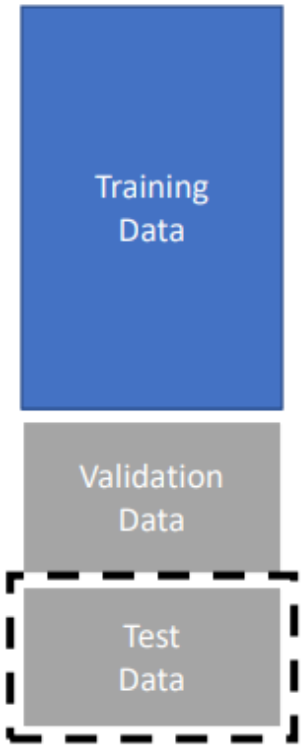
$$accuracy(h,T) = \frac{1}{|T|} \sum_{(x,y) \in T} [h(x) = y] = 1 - EmpLoss_{L_{0/1},T}(h)$$

Training a Model

- Models are “trained” (learned) on the training data (a part of the available data).
- Models have
  - Model parameters (the model): E.g., probabilities, weights, factors.
  - Hyperparameters: Choices for the algorithm used for learning. E.g., learning rate, regularization lamda, maximal decision tree depth, selected features.

Cont...

- The "Learner" (algorithm) tries to optimizes the model parameters given user-specified hyperparameters.
- We can learn models with different hyperparameters, but how do we know which set of hyperparameters is best?



Hyperparameter Tuning/Model Selection

- Learn models using the training set and different hyperparameters.
- Often a grid of possible hyperparameter combinations or some greedy search is used.
- Evaluate the models using the validation data and choose the model with the best accuracy. Selecting the right type of model, hyperparameters and features is called model selection.
- Learn the final model using all training and validation data.
- Notes: The validation set was not used for training, so we get generalization accuracy.
- If no model selection is necessary, then no validation set is used.

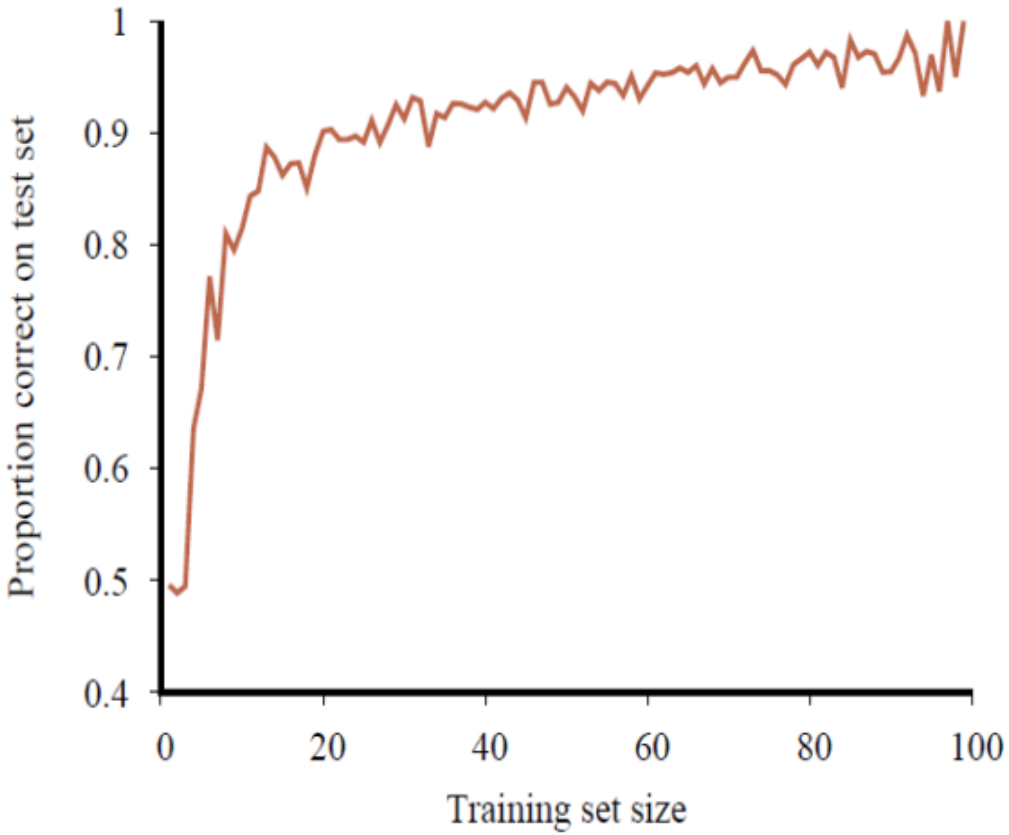
Testing a Model

- After the hyper parameters are chosen, the final model is evaluated against the test set to estimate the model accuracy.
- Very important: never "peek" at the test set during learning!

How to Split the Dataset

- Random splits: Split the data randomly in, e.g., 60 % training, 20 % validation, and 20 % testing.
- k fold cross validation: Use training and validation data better
  - split the training and validation data randomly into k folds.
  - For k rounds hold 1 fold back for testing and use the remaining k-1 folds for training.
  - Use the average error/accuracy as a better estimate.
  - Some algorithms/tools do that internally.
- LOOCV (leave-one-out cross validation): k = n used if very little data is available.

The Effect the Training Data Size



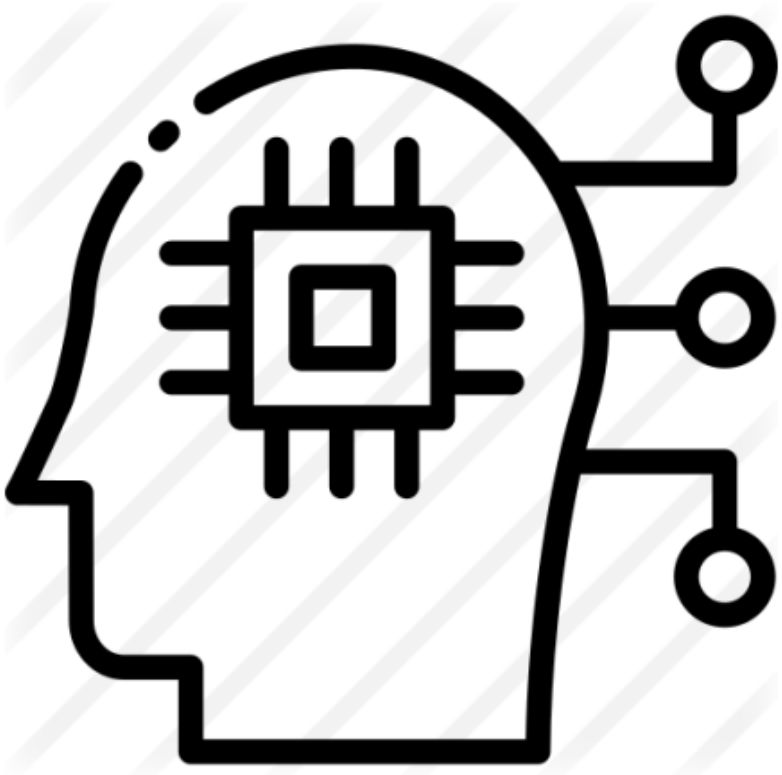
Accuracy of a classifier when the amount of available training data increases.

More data is better!

Cont....

- First step: get a baseline
  - Baselines are very simple “straw man” model.
  - Helps to determine how hard the task is.
  - Helps to find out what a “good” accuracy is.
- Weak baseline:
  - The most frequent label classifier
  - Gives all test instances whatever label was most common in the training set.
  - Example: For spam filtering, give every message the label “ham.”
  - Accuracy might be very high if the problem is skewed (called class imbalance).
  - Example: Calling everything “ham” gets already 66% right, so a classifier that gets 70% isn’t very good...
- Strong baseline: For research, we typically compare to previous work as a baseline.

Types of Models



- **Regression**: Predict a number
- **Classification**: Predict a label

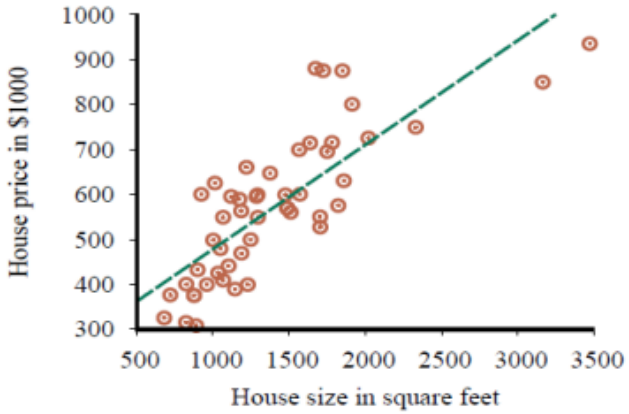
Cont...

Model: $h_{\boldsymbol{w}}(\boldsymbol{x}_j) = w_o + w_1 x_{j,1} + \cdots + w_n x_{j,n} = \sum_i w_i x_{j,i} = \boldsymbol{w}^T \boldsymbol{x}_j$

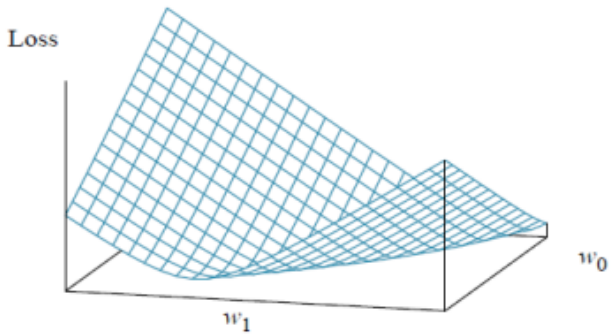
Loss function: $L(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2$ Squared error loss over the whole data matrix X

Gradient: $\nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = 2\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) = 0$ The gradient $\nabla_{\boldsymbol{w}} L(\boldsymbol{w})$ is a vector of partial derivatives

Solution: $\boldsymbol{w}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$



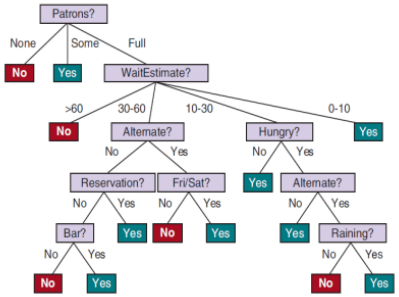
(a)



(b)

Decision Trees

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$



- A sequence of decisions represented as a tree.
- Many implementations that differ by
  - How to select features to split?
  - When to stop splitting?
  - Is the tree pruned?
- Approximates a Bayesian classifier by

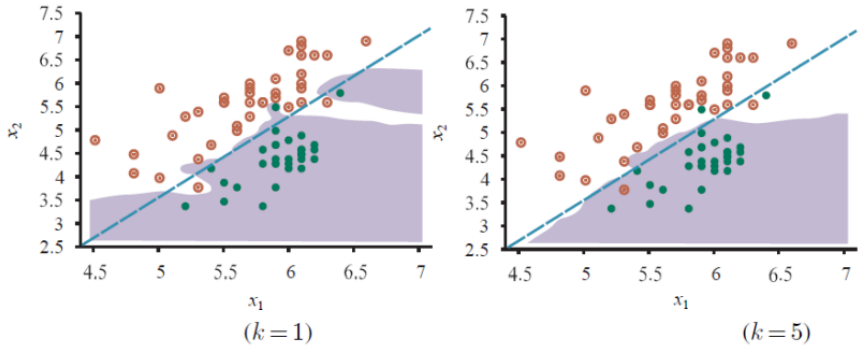
Naive Bayes Classifier

- Approximates a Bayes classifier with the naive assumption that all n features are conditional independent given the class.

$$h(x)^* = \underset{y}{\operatorname{argmax}}\, P(Y = y \mid X = x)$$

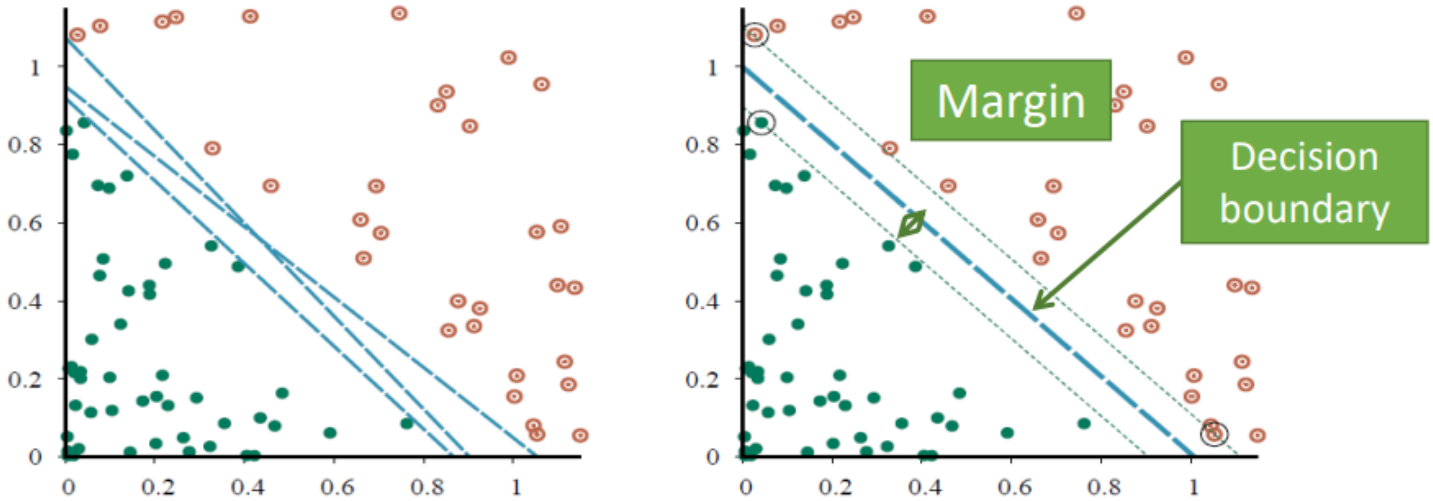
- We have only used discrete features so far, but it can be extended to continuous features.

K-Nearest Neighbors Classifier



- Class is predicted by looking at the majority in the set of the k nearest neighbors.
- Neighbors are found using a distance measure (e.g., Euclidean distance between points).
- k is a hyperparameter. Larger k smooth the decision boundary.

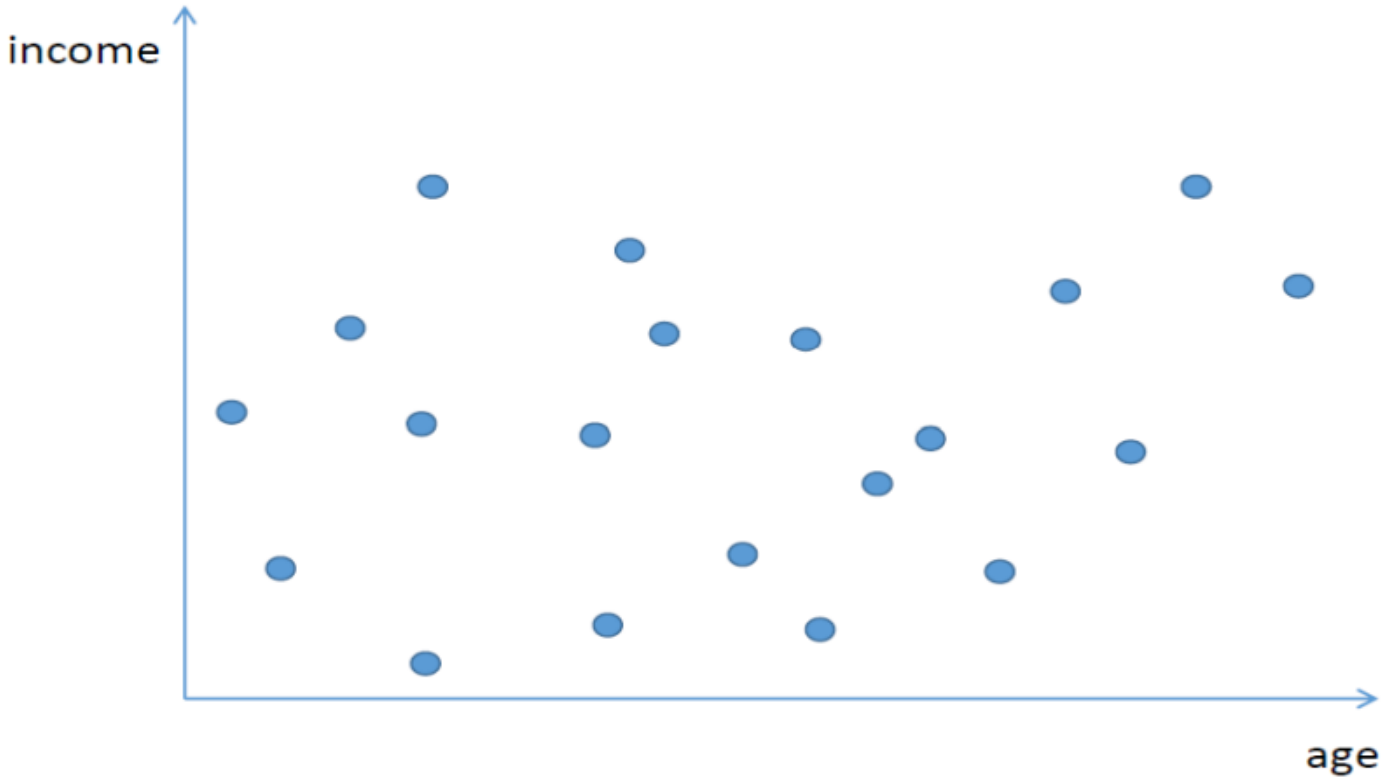
Support Vector Machine (SVM)



- Linear classifier that finds the maximum margin separator using only the "support vectors" and quadratic optimization.
- The kernel trick can be used to learn non-linear decision boundaries.

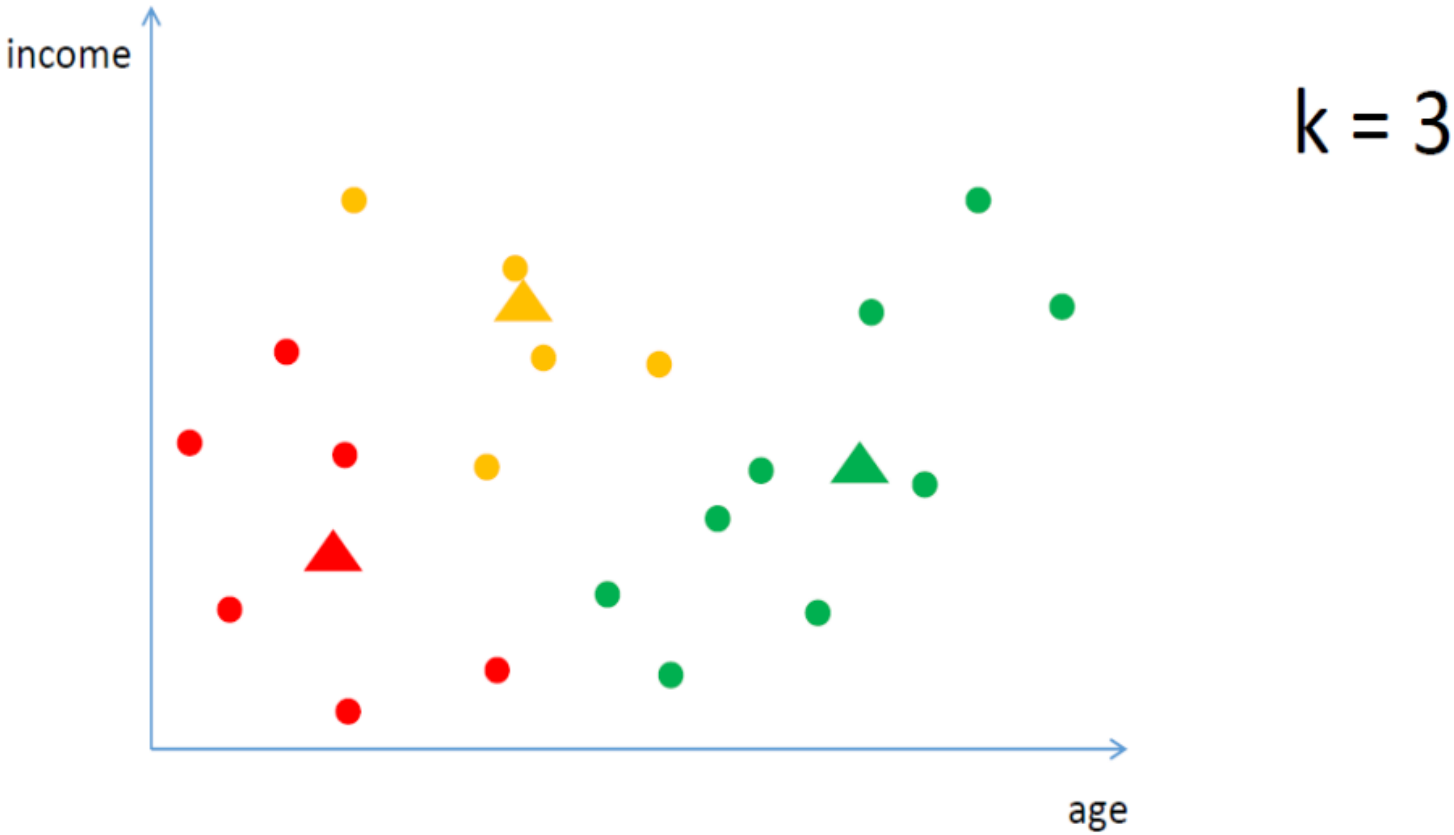
Unsupervised learning

- Unsupervised learning for example clustering (e.g. with K-means)



Unsupervised learning – K-means

- Unsupervised learning for example clustering (e.g. with K-means)

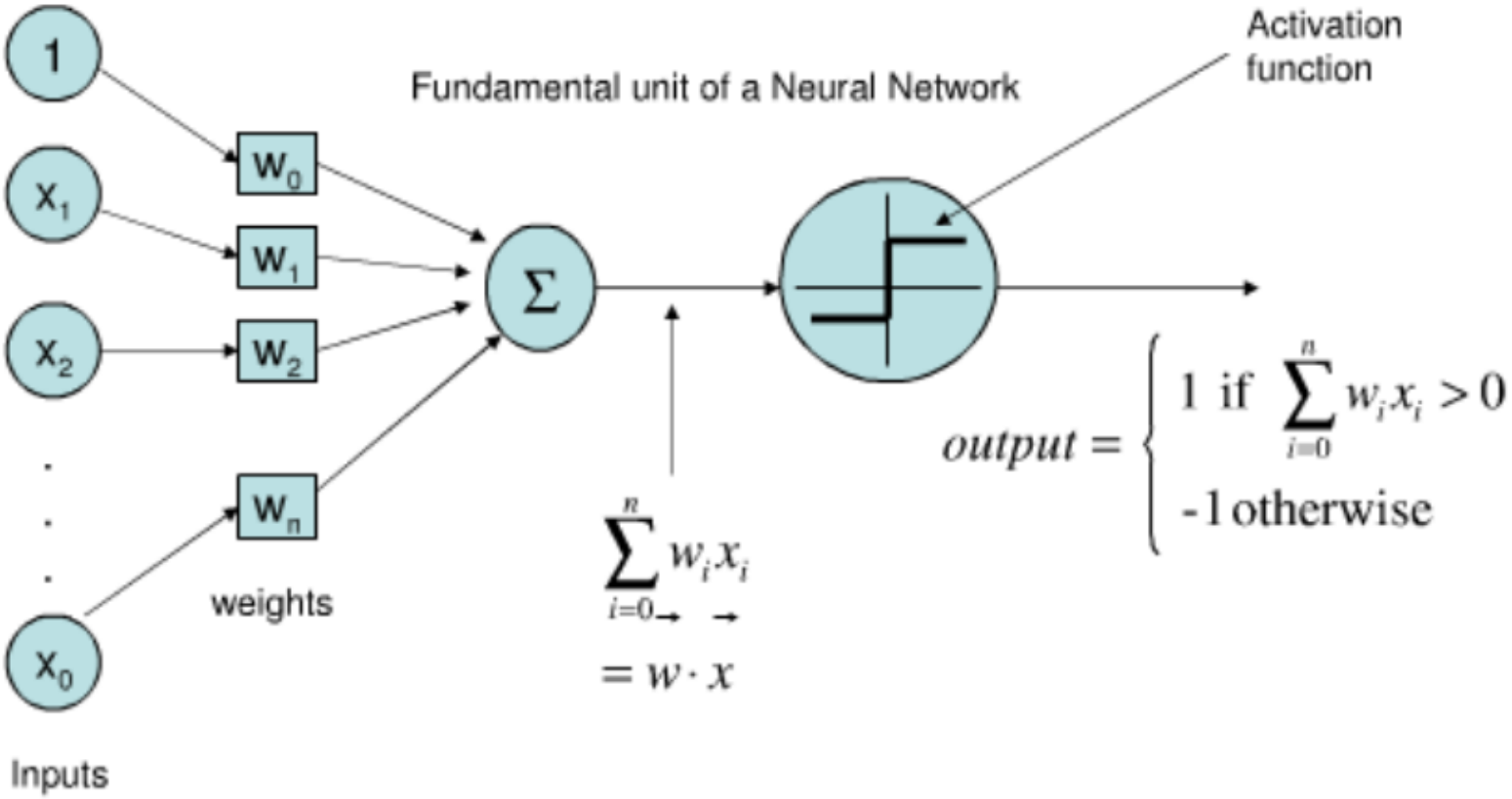


Artificial Neural Networks/Deep Learning

- Machine Learning
  - Uses statistical methods in the data
  - Curse of dimensionality (does not scale for big data)
- Neural networks learn from “experience”
  - It trains on the available data
  - Deep learning for big data

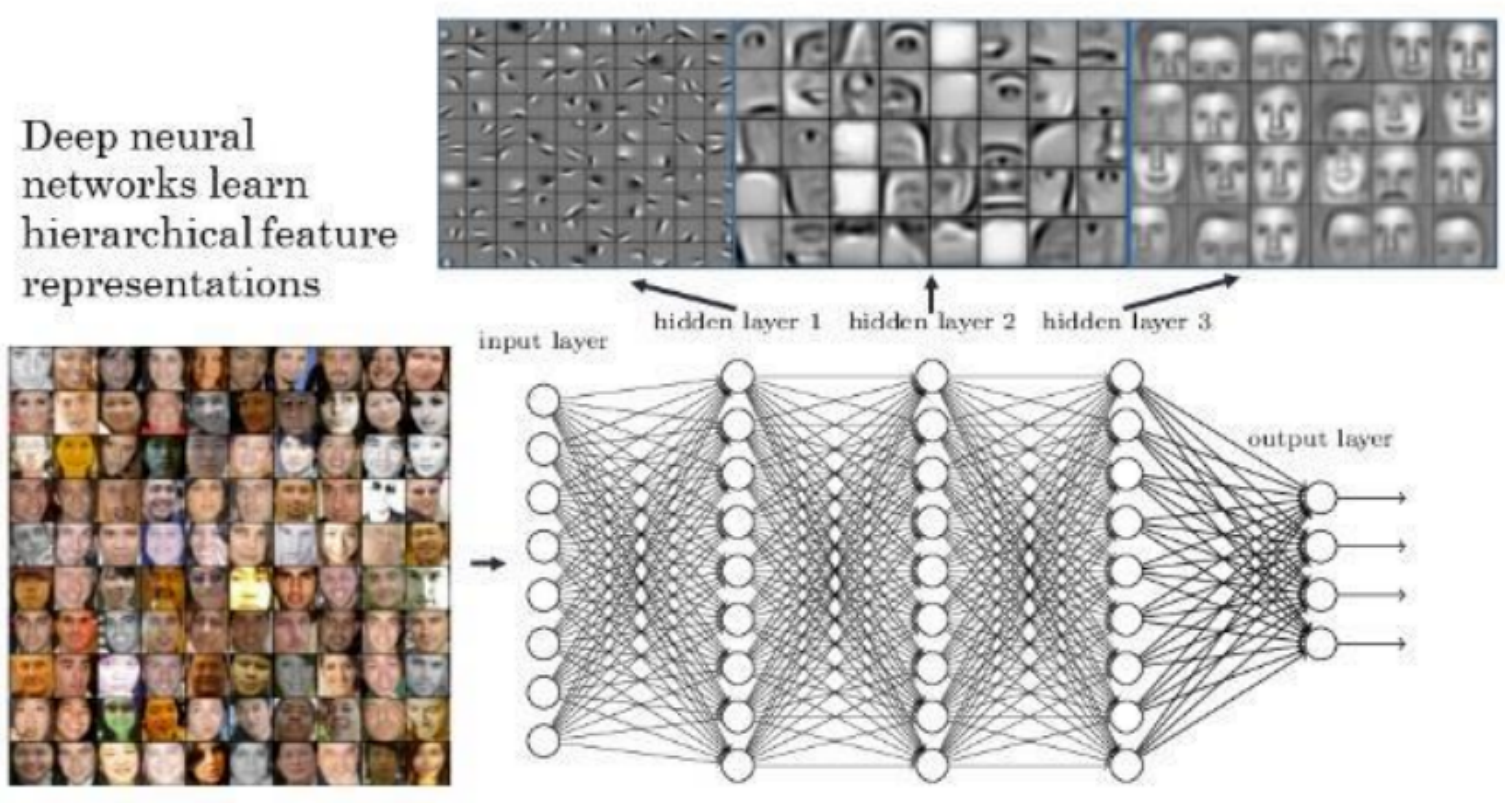
Cont...

- Perceptron (artificial neuron)



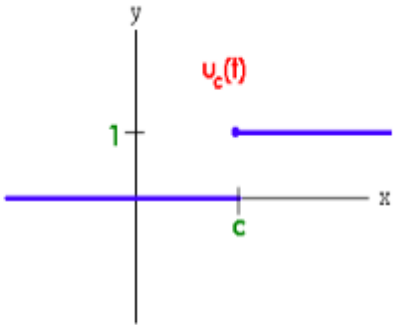
Cont...

- Deep learning

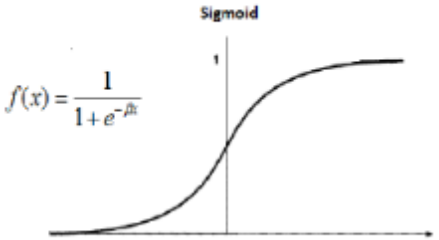


Cont...

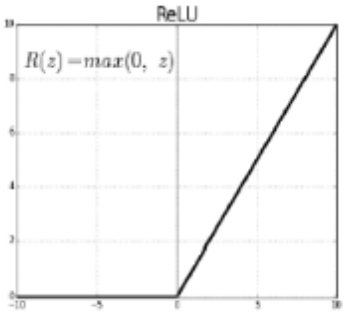
- Activation functions



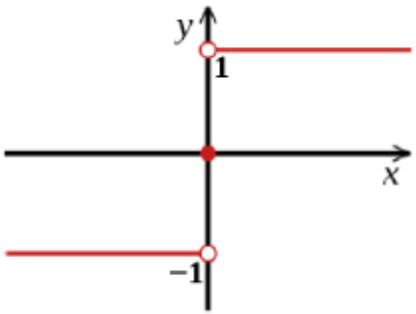
Step function



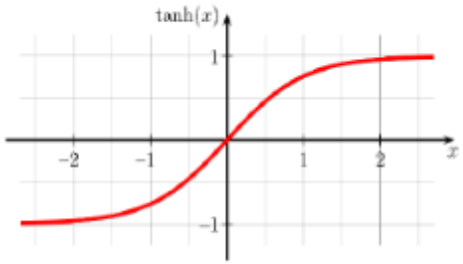
Sigmoid function



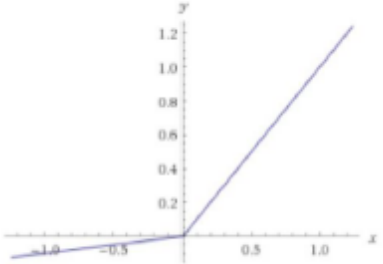
ReLU function



Sign function



Tanh function



Leaky ReLU function

Cont...

- Some interesting types of neural networks
  - Perceptron
  - Multi layer perceptron (MLP)
    - Feed Forward (forward propagation) neural networks (calculating results)
    - Back propagation (learning)
  - Convolutional neural networks (CNN)
    - Image processing
  - Recurrent neural networks (RNN)
    - Sequential data