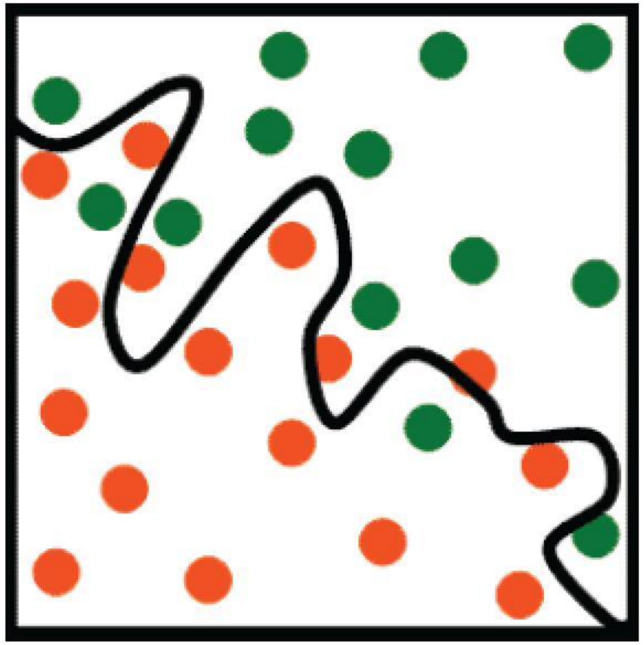
Supervised Learning (II)



This Lecture: Supervised Learning Methods

I. Linear Model (detail)
II. Decision Tree (detail)
III. Neural Network (brief)
IV. k-Nearest Neighbours (brief)
V. Support Vector Machine (brief)

I. Linear Model (线性模型)

I.1 Univariate Linear Regression (ULR) 单元/一元线性回归

I.2 Multivariate Linear Regression (MLR) 多元线性回归

I.3 Multivariate Linear Classification (MLC) 多元线性分类

Example: House Price

- [Question] How to predict the price of a new house?
- [Answer] Fit a straight line using the training data

  ⇒ linear regression.

- One variable (house size)

  ⇒ univariate.

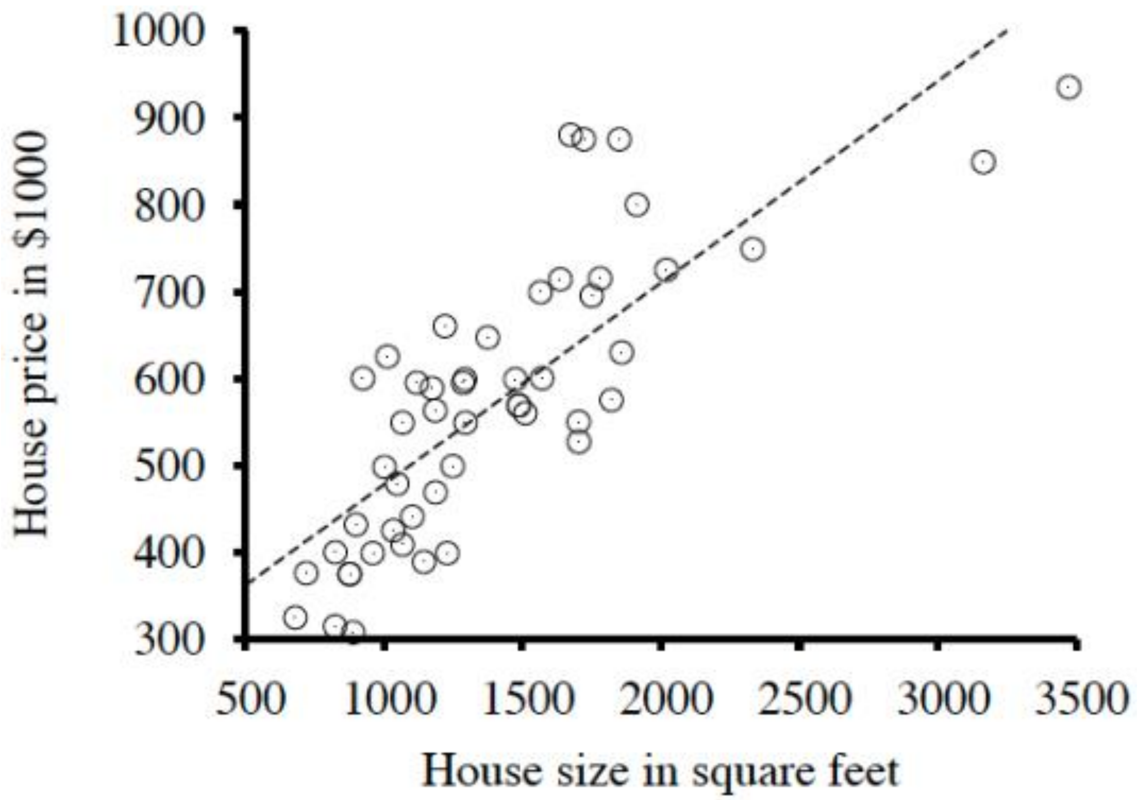


Image source: Figure 18.13.a of the AI book by S. Russell & P. Novig.

Model Formulation

- Linear model: $h_{\boldsymbol{w}}(x) = w_0 + w_1 x$,
  - $\mathbf{w} = [w_0, w_1]^T \in \mathbb{R}^{2\times 1}$: model parameters,
  - $x \in \mathbb{R}^1$: input feature (特征), e.g. house size.
- Training data: $\mathcal{D} = \left\{\left(x^{(n)}, y^{(n)}\right) | \boldsymbol{y}^{(n)} \in \mathbb{R}^1\right\}_{n=1}^{N}$.
- Aim: find the optimal $\boldsymbol{w}$ fitting the observations in $\mathcal{D}$.
- Optimization: minimize empirical square loss regarding $\boldsymbol{w}$ as

$$min_{\boldsymbol{w}}\, \mathcal{L}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left[y^{(n)} - \left(w_1 x^{(n)} + w_0\right)\right]^2.$$

Model Formulation

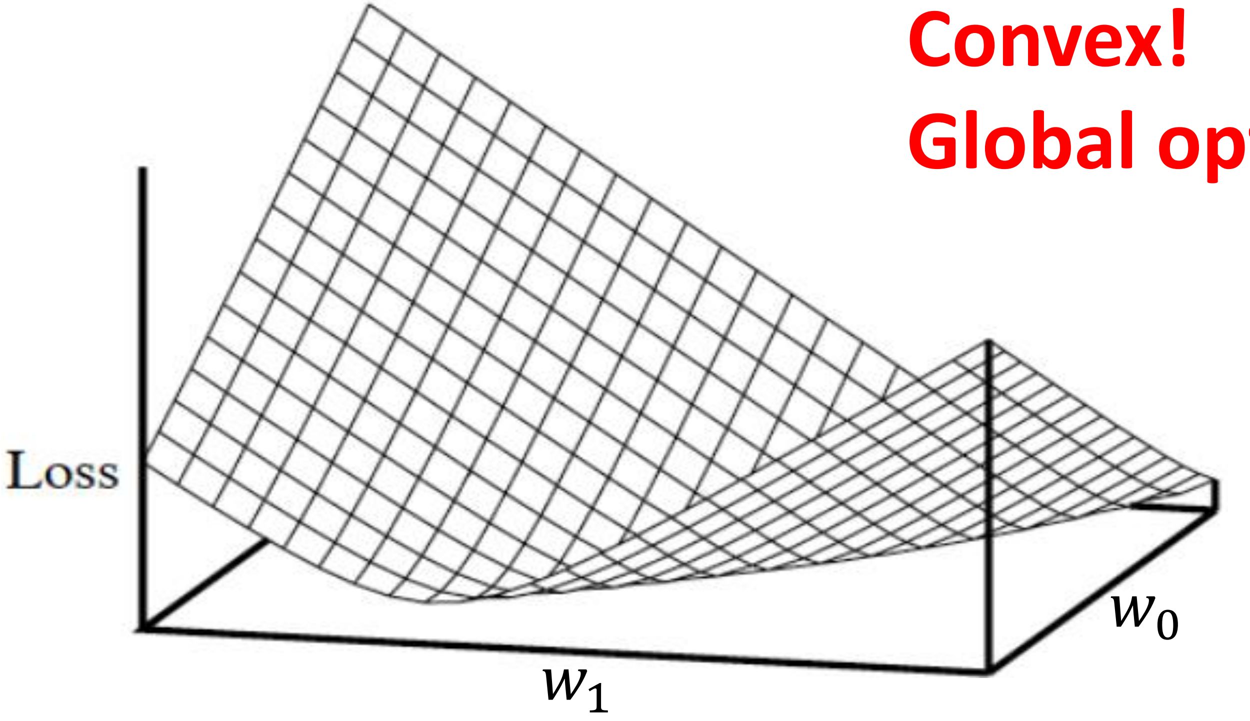
- Aim: find the optimal $\boldsymbol{w}$ fitting the observations in $\mathcal{D}$.
- Optimization: minimize empirical square loss regarding $\boldsymbol{w}$ as

$$min_{\boldsymbol{w}}\ \mathcal{L}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left[y^{(n)} - \left(w_1 x^{(n)} + w_0\right)\right]^2.$$

- **Remarks**:
  - Square loss -> Euclidean distance (欧式距离) -> least square method (最小二乘法)
  - Finding optimal $\boldsymbol{w}$: parameter estimation (参数估计)

Model Parameter Space

- Plot 3D graph for $\mathcal{L}(w_0, w_1) = \frac{1}{2}\sum_{n=1}^{N}\left[y^{(n)} - \left(w_1 x^{(n)} + w_0\right)\right]^2$



Convex!

Global optima assured!

Image source: Figure 18.13.b of the AI book by S. Russell & P. Novig.

1. Closed-form Solution (闭式解)

- Optimization: $\mathcal{L}(w_0, w_1) = \frac{1}{2}\sum_{n=1}^{N}\left[y^{(n)} - \left(w_1 x^{(n)} + w_0\right)\right]^2$.
- First-order equations:

$$\frac{\partial}{\partial w_0}\mathcal{L}(w_0, w_1) = 0,\ \frac{\partial}{\partial w_1}\mathcal{L}(w_0, w_1) = 0.$$

- Solution:

$$w_1 = \frac{N\left(\sum_n x^{(n)} y^{(n)}\right) - \left(\sum_n x^{(n)}\right)\left(\sum_n y^{(n)}\right)}{N\left(\sum_n (x^{(n)})^2\right) - \left(\sum_n x^{(n)}\right)^2}$$

$$w_0 = \frac{\sum_n y^{(n)} - w_1 \sum_n x^{(n)}}{N}.$$

2. Iterative Solution

- Sometimes there is no closed-form solution.
- Gradient Descent (GD, 梯度下降法):

  1. $\boldsymbol{w} \leftarrow$ any point in the parameter space

  2. **LOOP** until convergence **DO**

  3. **FOR** each w_i in $\boldsymbol{w}$ **DO**

  4. $w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} \mathcal{L}(\boldsymbol{w})$,

- α: learning rate, positive.

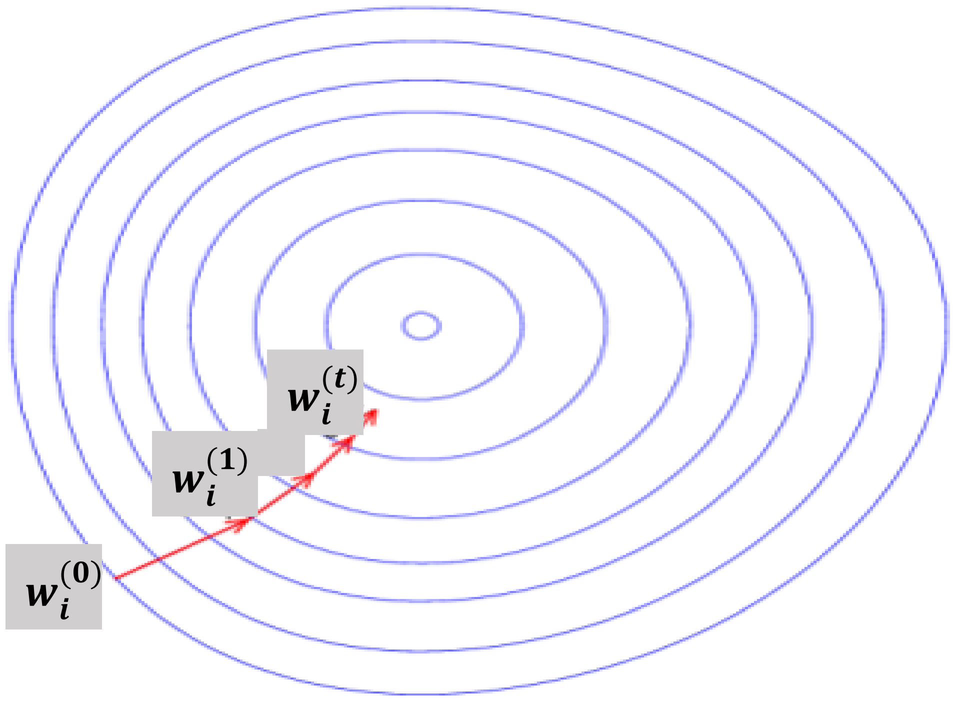
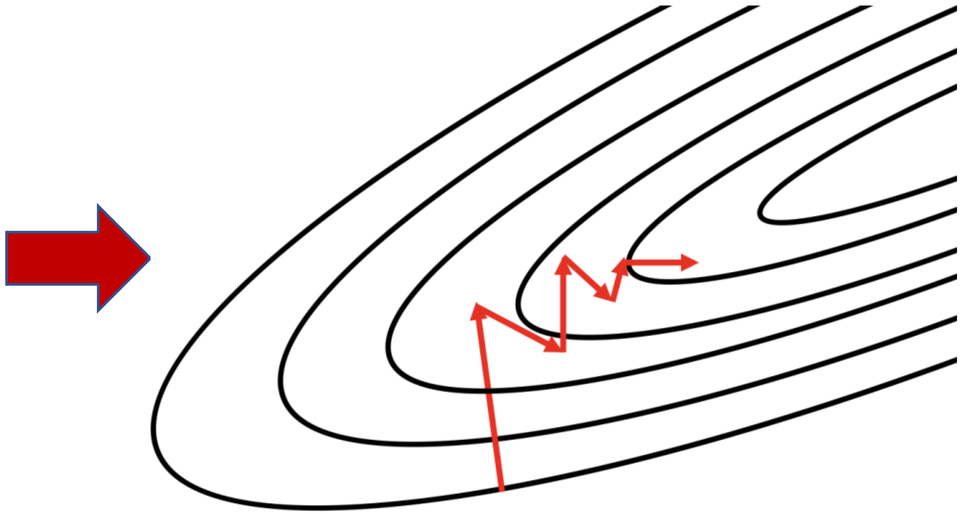


Image source: https://commons.wikimedia.org/wiki/File:Gradient_descent.svg

2. Iterative Solution: Advanced

- Batch GD: update $\boldsymbol{w}$ once with all training samples.
  - Guarantee global optimum but slow.
- Stochastic GD: update $\boldsymbol{w}$ N times with one training data for one update.
  - Fast but do not guarantee global optimum with a fixed α.
  - Online/offline settings
- Mini-batch SGD: update $\boldsymbol{w}$ several times with a subset of $\mathcal{D}$ for one update.



Zigzag problem of SGD. Image source: Figure 4.10 of "Fundamentals of Deep Learning" by Nikhil Buduma.

I. Linear Model (线性模型)

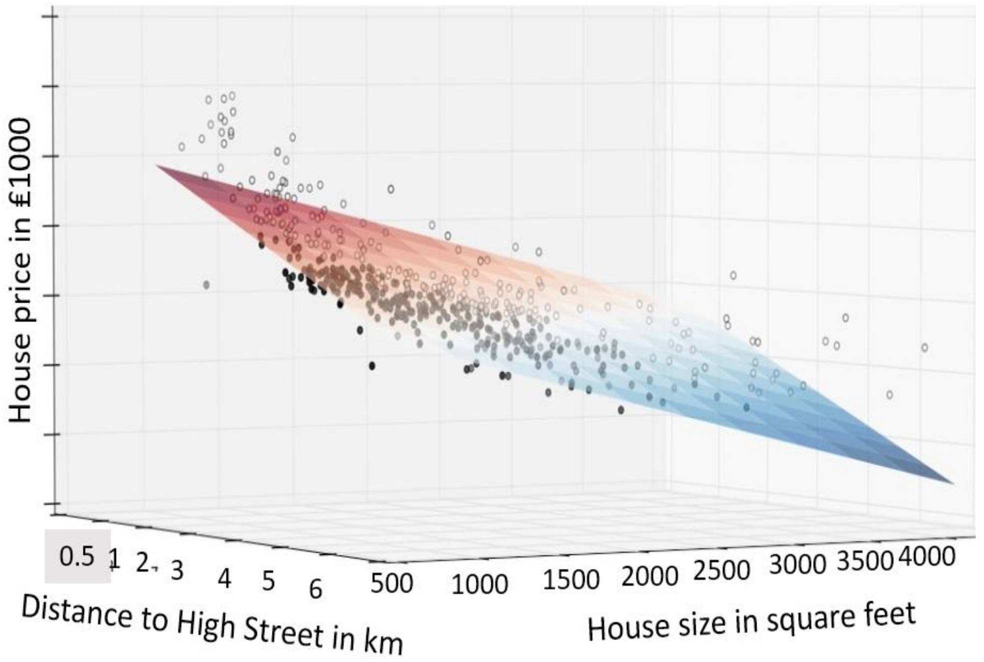
I.1 Univariate Linear Regression (ULR) 单元/一元线性回归
I.2 Multivariate Linear Regression (MLR) 多元线性回归
I.3 Multivariate Linear Classification (MLC) 多元线性分类

Example: House Price Revisit

- One more feature: distance to High Street.

- [Question] How to predict the price of a new house with the two features?
- [Answer] Fit a plane using $\mathcal{D}$
  $\Rightarrow$ linear regression.

- Multiple variables (distance + size)
  $\Rightarrow$ multivariate.



Model Formulation

- Linear model: $h_{\boldsymbol{w}}(\boldsymbol{x}) = w_0 + w_1 x + \cdots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x} \in \mathbb{R}^1$,
  - $\mathbf{w} = [w_0, w_1, \cdots, w_m]^T \in \mathbb{R}^{(m+1)}$: model parameters.
  - $\boldsymbol{x} = [1, x_1, \cdots, x_m]^T \in \mathbb{R}^{(m+1)}$: input features; e.g. size, distance to High Street, etc.
- Training data: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right) \mid y^{(n)} \in \mathbb{R}^1\right\}_{n=1}^{N}$.
- Aim: find the optimal $\boldsymbol{w}$ fitting the observations in $\mathcal{D}$.
- Optimization: minimize empirical loss regarding $\boldsymbol{w}$ as

$$min_{\boldsymbol{w}}\, \mathcal{L}(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \left[y^{(n)} - \boldsymbol{w}^T \boldsymbol{x}^{(n)}\right]^2.$$

1. Closed-form Solution

- Optimization: $min_{\boldsymbol{w}}\, \mathcal{L}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left[y^{(n)} - \boldsymbol{w}^T\boldsymbol{x}^{(n)}\right]^2.$
- First-order equations: $\frac{\partial}{\partial w_i}\mathcal{L}(\boldsymbol{w}) = 0, \forall i = 0,1,\cdots,m$
- Solution: solve the system of linear equations to have

$$\boldsymbol{w} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y},$$

  - $\boldsymbol{X} = \left[x'^{(1)}, \cdots, x'^{(N)}\right]^T \in \mathbb{R}^{N\times(m+1)},$
  - $\boldsymbol{y} = \left[y^{(1)}, \cdots, y^{(N)}\right]^T \in \mathbb{R}^{N\times 1}.$

2. Iterative Solution

- Optimization: $min_{\boldsymbol{w}}\, \mathcal{L}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left[y^{(n)} - \boldsymbol{w}^T\boldsymbol{x}^{(n)}\right]^2$.
- Gradient descent (GD):

$$w_i \leftarrow w_i + \alpha \sum_n \left(y_i^{(n)} - \boldsymbol{w}^T\boldsymbol{x}^{(n)}\right) \cdot x_i^{(n)}$$

  - α: learning rate, positive.

Overfitting for MLR

- MLR in a high-dimensional space may encounter overfitting.
- MLR: common to use regularization.

- ULR does not have this problem – only 1 feature.

Regularized Objective for MLR

- Regularized Objective:

$$min_{\boldsymbol{w}}\, \mathcal{L}_{tr}(\boldsymbol{w}) + \lambda \cdot \Omega(\boldsymbol{w}).$$

- $\mathcal{L}_{tr}(\boldsymbol{w})$: training loss; measures how well the model fits the training data.
  - Square loss: $l(y^{(n)}, \widehat{y^{(n)}}) = (y^{(n)} - \widehat{y^{(n)}})^2 = (y^{(n)} - \boldsymbol{w}^T \boldsymbol{x}^{(n)})^2.$
  - Logistic loss: $l\left(y^{(n)}, \widehat{y^{(n)}}\right) = y^{(n)} \ln\left(1 + e^{-\widehat{y^{(n)}}}\right) + \left(1 - y^{(n)}\right) \ln(1 + e^{\widehat{y^{(n)}}})$
- λ: trade-off & manually tuning parameter.

Regularization

- $\Omega(\boldsymbol{w})$: regularization; how complex the model is?
- $\Omega(\boldsymbol{w}) \triangleq \ell_p(\boldsymbol{w}) = \sum_i |w_i|^p$, in particular:
  - ℓ_0 regularization: $p = 0$, penalize #(non-zero parameters);
  - ℓ_1 regularization: $p = 1$, penalize the sum of the absolute parameters;
  - ℓ_2 regularization: $p = 2$, penalize the sum of square parameters.
- [Question] Which ℓ_p (p范数) should we use?
- [Answer] Depend on the specific problem.

Illustration: ℓ_1 vs ℓ_2

- Let $w = [w_1, w_2]^T$, we have:
  - $\ell_1 = |w_1| + |w_2|$,
  - $\ell_2 = w_1^2 + w_2^2$.
- Plot contours for $\ell_1 = \ell_2 = c$.
- Goodness of ℓ_1: sparse model.

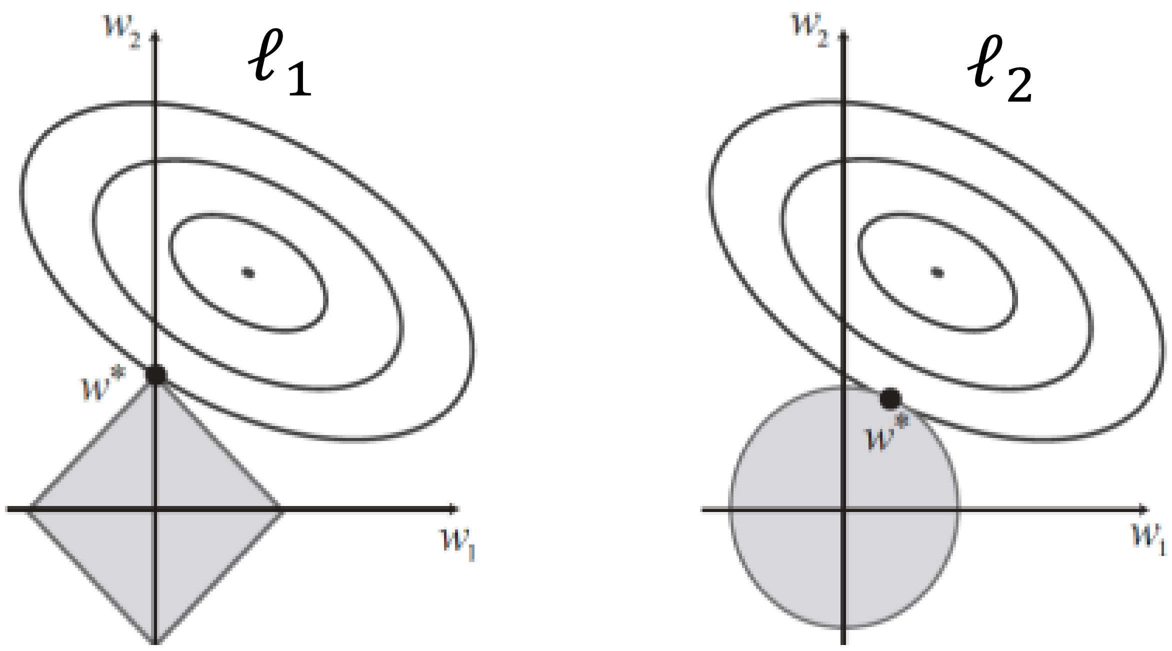


Image source: Figure 18.14 of the AI book by S. Russell & P. Novig.

Exercise: Closed-form Solution of MLR

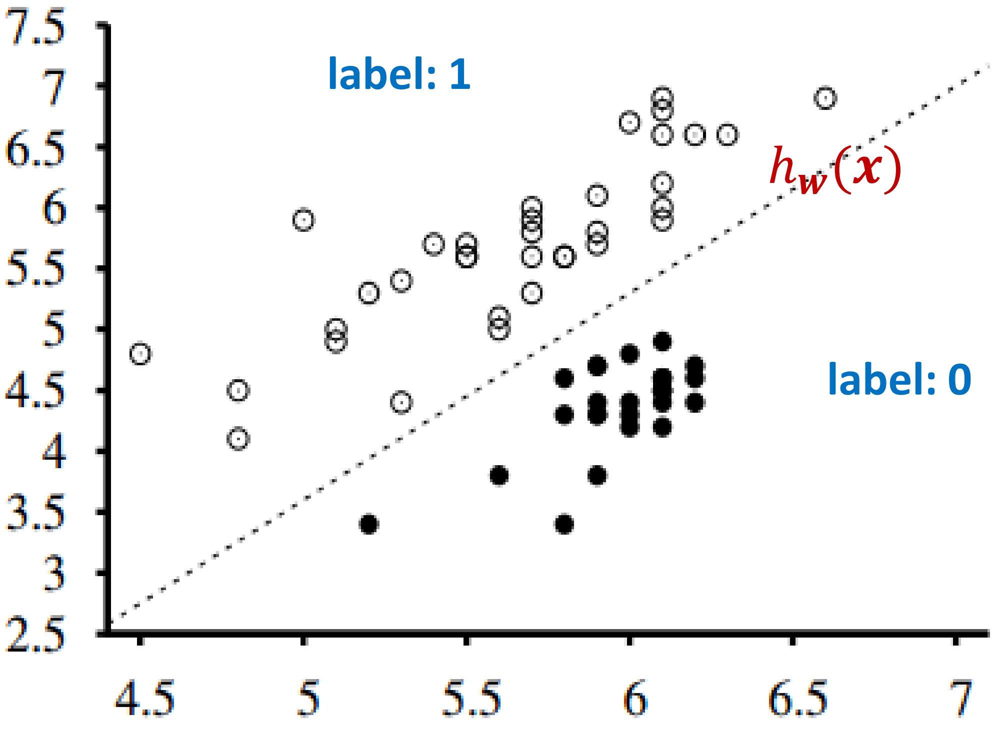
- [Question] Derive the closed-form solution of MLR?
- [Hint] At variable-level:
  1) Compute and write the i^{th} equation $\frac{\partial}{\partial w_i}\mathcal{L}(\boldsymbol{w}) = 0$.
  2) Re-write the i^{th} equation into ‘vector-vector’ form.
  3) Align the m equations about $x_i^{(n)}$ into a matrix-vector form. Note to check the match of dimensionality.
  4) Obtain the representation of $\boldsymbol{w}$.
- [Hint] The solution is much easier to characterize in matrix notation.

I. Linear Model (线性模型)

1. Univariate Linear Regression (ULR) 单元/一元线性回归
2. Multivariate Linear Regression (MLR) 多元线性回归
3. **Multivariate Linear Classification (MLC) 多元线性分类**

Example: Earthquakes or Explosions

- [Question] How to distinguish earthquakes (○) from explosions (•) using two features?
- [Answer] Learn a linear decision boundary that can separate the two-class points.
- [Denote] the classifier as $h_{\boldsymbol{w}}(\boldsymbol{x})$.
  - Classifier: linear decision boundary.



Linear separable data points. Image source: Figure 18.15.a of the AI book by S. Russell & P. Novig.

Classification Problem Formulation

- Training data: $\mathcal{D} = \{(x^{(n)}, y^{(n)}) \mid y^{(n)} \in \{0,1\}\}_{n=1}^{N}$.
- Aim: find the optimal $\boldsymbol{w}$ fitting the observations in $\mathcal{D}$.
- Optimization: minimize square-error regarding $\boldsymbol{w}$ as

$$min_{\boldsymbol{w}}\ \mathcal{L}(\boldsymbol{w}) = \frac{1}{2N}\sum_{n=1}^{N}[y^{(n)} - h_{\boldsymbol{w}}(\boldsymbol{x}^{(n)})]^2.$$

- Remaining Question: How to formulate $h_{\boldsymbol{w}}(\boldsymbol{x})$?

Problem of MLR Model in Classification

- MLR model: $h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} \in \mathbb{R}^1$.
- Problem: $h_{\boldsymbol{w}}(\boldsymbol{x}) \in \mathbb{R}^1$ cannot constrain $0/1$ output.

=> Hard-threshold Linear Classifier (带硬阈值的线性分类器)

Hard-threshold Classifier

- Hard-threshold function:

$$\sigma(z) = \mathbb{1}_{z\geq 0} = \begin{cases} 1, z \geq 0 \\ 0, z < 0 \end{cases}.$$

- Hard-threshold classification model:

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{x}) \in \{0,1\},$$

  - $h_{\boldsymbol{w}}(\boldsymbol{x})$ has $0/1$ output.

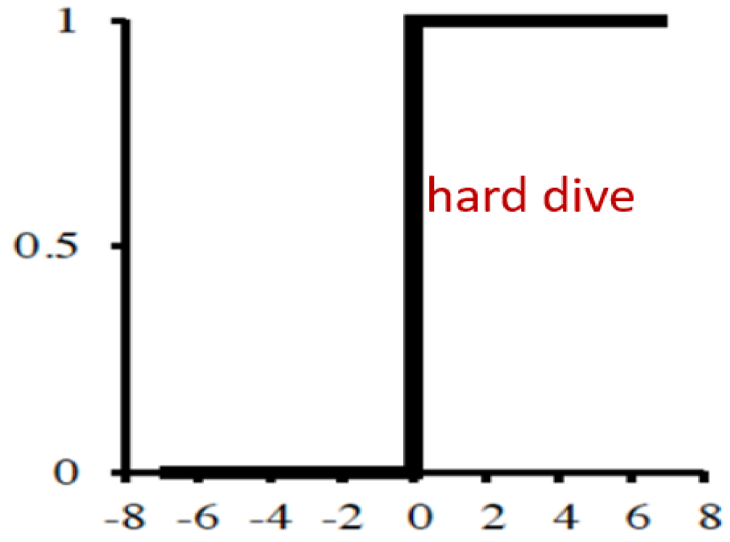


Illustration on $\sigma(z)$

Image source: Figure 18.17.a of the AI book by S. Russell & P. Novig.

Problem in Learning Method

- Problem: The below derivative does NOT exist (either 0 or undefined)

$$\frac{\partial}{\partial w_i} h_{\boldsymbol{w}}(\boldsymbol{x}) = \frac{\partial}{\partial z} \boxed{\sigma\,(z)} \frac{\partial(\boldsymbol{w}^T\boldsymbol{x})}{\partial w_i}.$$

- Closed-form solution: Cannot proceed.
- Iterative solution: $w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} \mathcal{L}(\boldsymbol{w})$ needs the above derivative.

Classification optimization:

$$min_{\boldsymbol{w}}\ \mathcal{L}(\boldsymbol{w}) = \frac{1}{2N} \sum_{n=1}^{N} [y^{(n)} - h_{\boldsymbol{w}}(\boldsymbol{x}^{(n)})]^2.$$

Proposed Learning Method

- Perceptron learning rule: For a training point $(\boldsymbol{x}, y)$, update as

$$w_i \leftarrow w_i + \alpha\big(y - h_{\boldsymbol{w}}(\boldsymbol{x})\big) \cdot x_i, \text{ for } i \in \{1, \cdots, m\}.$$

- Identical to the MLR case in form.

- Implementation:
  - $y \cdot h_{\boldsymbol{w}}(\boldsymbol{x}) = 1$: keep w_i unchanged;
  - $y = 1 \,\&\, h_{\boldsymbol{w}}(\boldsymbol{x}) = 0$: increase w_i if $x_i > 0$ and decrease w_i if $x_i < 0$;
  - $y = 0 \,\&\, h_{\boldsymbol{w}}(\boldsymbol{x}) = 1$: decrease w_i if $x_i > 0$ and increase w_i if $x_i < 0$.

From Hard-threshold to Soft-threshold

Hard threshold: $\sigma(z) = \mathbb{1}_{z\geq 0}$

- $\mathbb{1}_{z\geq 0}$: indicator function.
- Not differentiable.

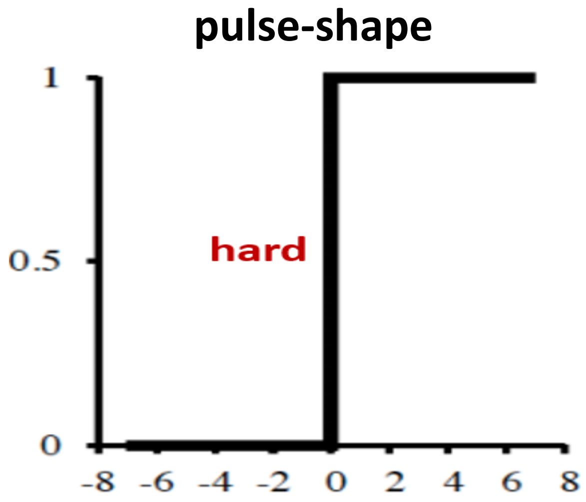


Image source: Figure 18.17.a of the AI book by S. Russell & P. Novig.

Soft threshold function: $\sigma(z) = s(z) = \frac{1}{1+e^{-z}}$

- $s(z)$: sigmoid function.
- Differentiable: $s'(z) = s(z)[1 - s(z)]$.

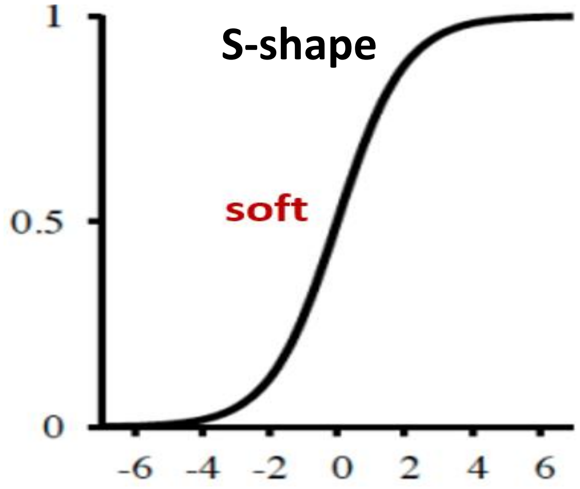


Image source: Figure 18.17.b of the AI book by S. Russell & P. Novig.

Logistic Regression (对数几率回归)

- Logistic regression model: $h_{\boldsymbol{w}}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x})$, where $\sigma(z) = \frac{1}{1+e^{-z}}$.
- Closed-form solution: Does not exist.
- Iterative solution: $w_i \leftarrow w_i - \alpha \frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial w_i}$
  - $\frac{\partial \mathcal{L}(\boldsymbol{w})}{\partial w_i} = -\frac{1}{N}\sum_{n=1}^{N}\left[y^{(n)} - h_{\boldsymbol{w}}\left(\boldsymbol{x}^{(n)}\right)\right] \cdot h_{\boldsymbol{w}}\left(\boldsymbol{x}^{(n)}\right) \cdot \left[1 - h_{\boldsymbol{w}}\left(\boldsymbol{x}^{(n)}\right)\right] \cdot x_i^{(n)}$
  - α: learning rate, positive.

Classification optimization:

$min_{\boldsymbol{w}}\, \mathcal{L}(\boldsymbol{w}) = \frac{1}{2N}\sum_{n=1}^{N}[y^{(n)} - h_{\boldsymbol{w}}(\boldsymbol{x}^{(n)})]^2.$

II. Decision Tree (决策树)

1. **Tree Representation**
2. Decision Tree Construction with Heuristics
   - ❖ Information Gain: Good Feature Heuristics
   - ❖ Information Gain: Continuous Feature
   - ❖ Overall: Decision Tree Construction
3. Tree Overfitting
4. Decision Tree for Regression

Example: Tree-shape Model

- [Question] How to judge an animal to be a mammal by two features?
- [Answer] Learn a decision tree that can separate the training samples.
- Goodness: Natural for humans, easy to interpret the results.

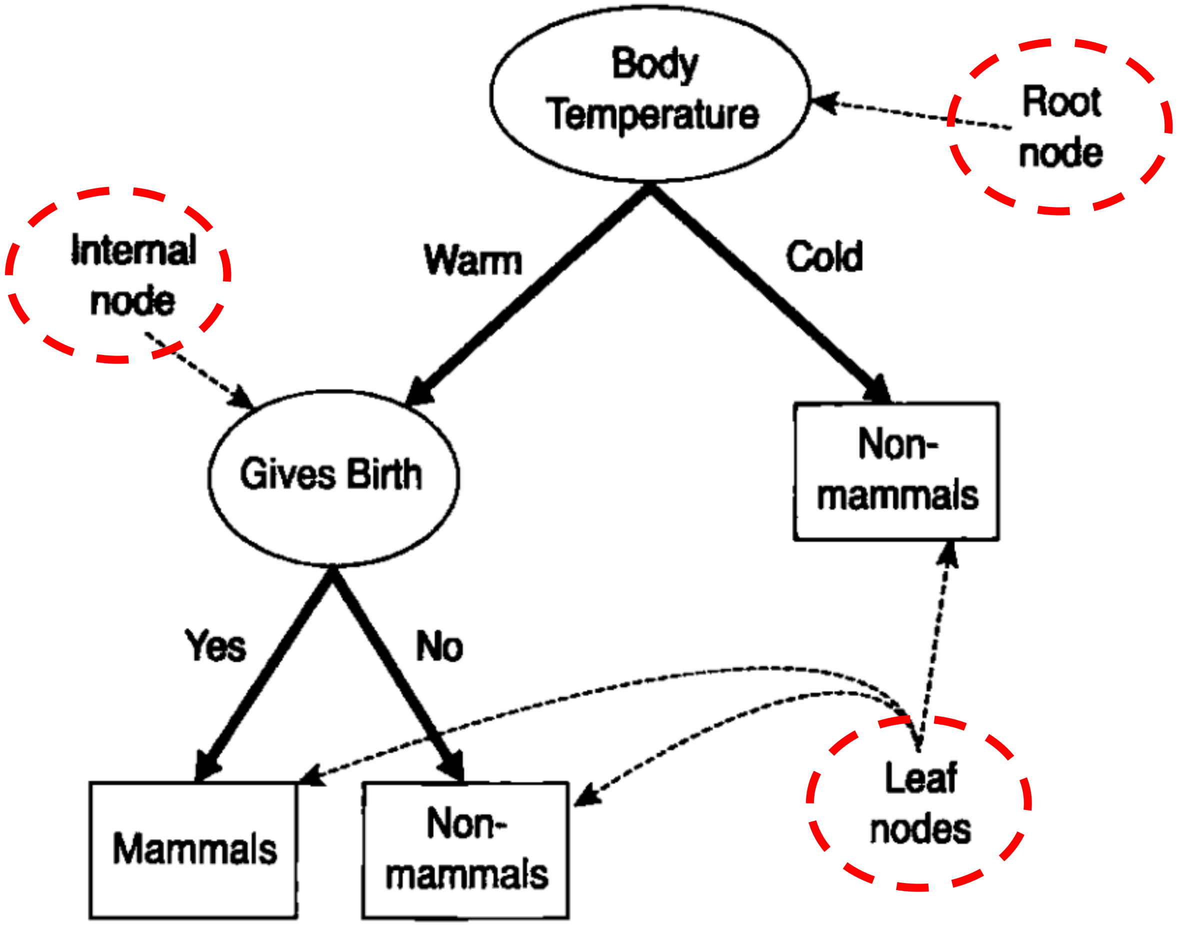


Image source: Figure 5.6 of "Introduction to Data Mining" by P. Tan, M. Steinbach and V. Kumar.

Tree Representation

- Tree model: a function mapping feature vector $\boldsymbol{x}$ to a decision y via a sequence of tests.
  - Discrete y: a decision tree for classification.
  - Continuous y: a decision tree for regression.

- Two types of nodes:
  - Decision nodes: a test on some feature.
  - Leaf nodes: a decision of the tree model.

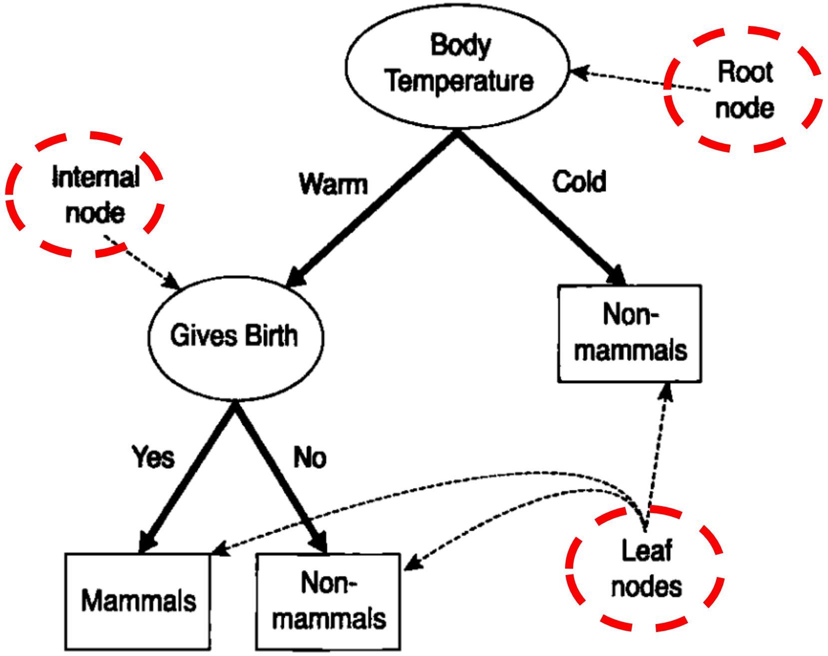


Image source: Figure 5.6 of "Introduction to Data Mining" by P. Tan, M. Steinbach and V. Kumar.

[Example] Waiting at a Restaurant

- Prediction: Should we wait for a table?
- # Input features: 10
  - Alternate: is there an alternative restaurant nearby?
  - Bar: is there a comfortable bar area to wait in?
  - Fri/Sat: is today Friday or Saturday?
  - Hungry: are we hungry?
  - Patrons: # people in the restaurant (None, Some, Full)
  - Price: price range.
  - Raining: is it raining outside?
  - Reservation: have we made a reservation?
  - Type: kind of restaurant (French, Italian, Thai, Burger)
  - Wait-Estimate: estimated waiting time (0-10, 10-30, 30-60, >60)

True Decision Tree f

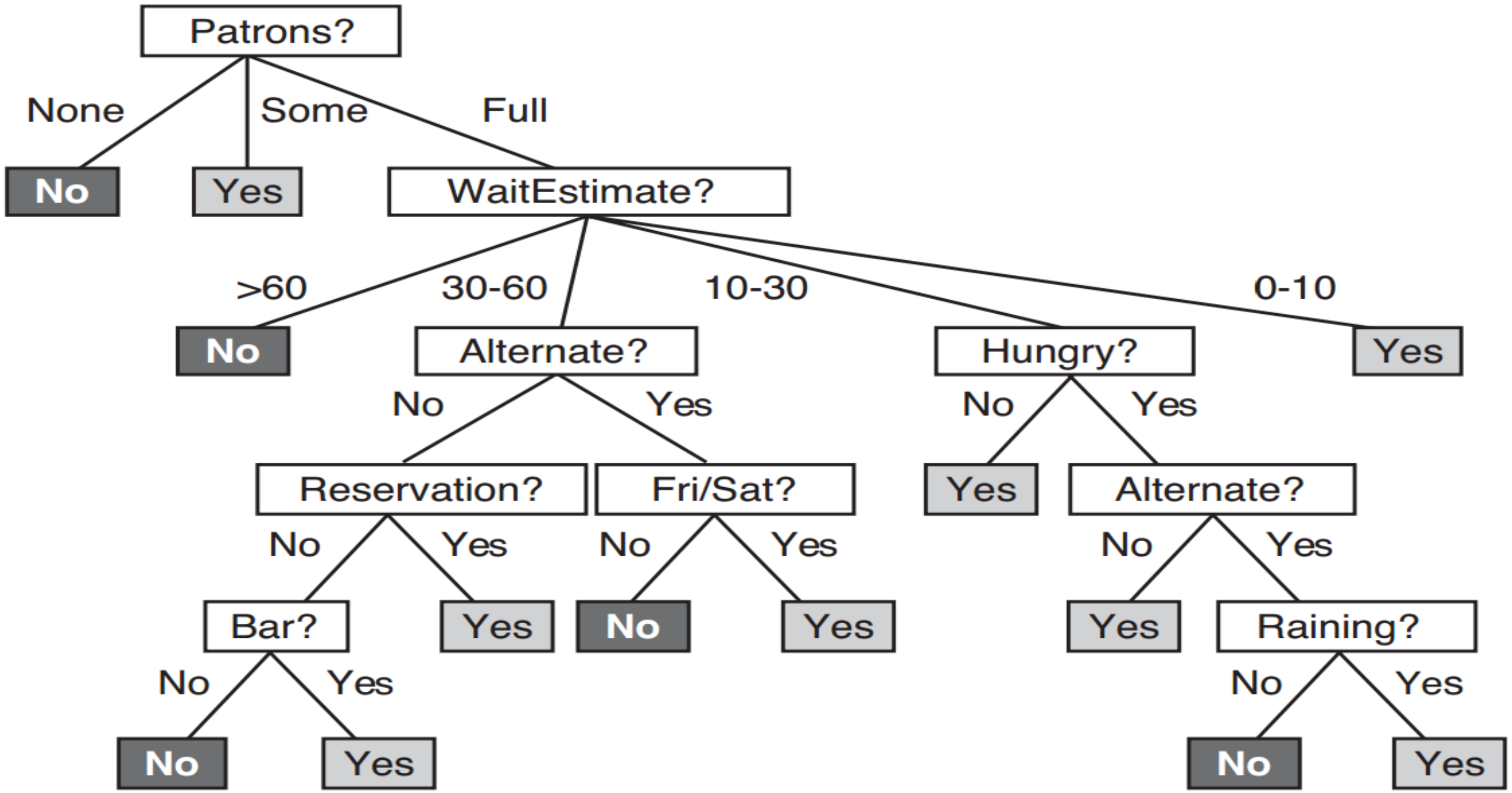


Fig. The decision tree f for deciding whether to wait for a table. Image source: Figure 18.4 of the AI book by S. Russell & P. Novig.

12 Training Examples (6+ 6-)

- Generate 12 training examples from the true decision tree.

Example	Input Attributes										Goal
	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*	$y_{12} = Yes$

Image source: Figure 18.3 of the AI book by S. Russell & P. Novig.

True Decision Tree f

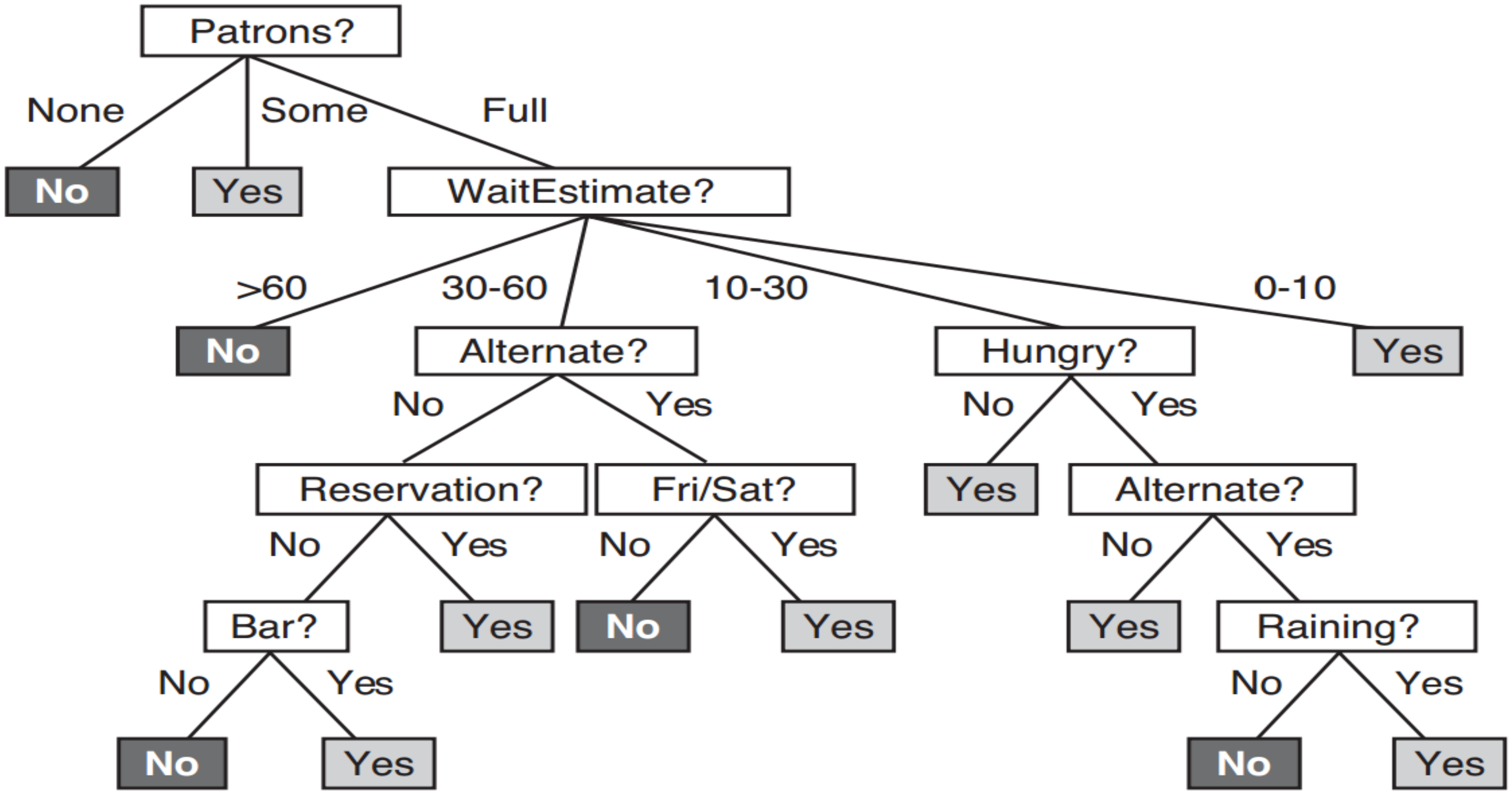


Fig. The decision tree f for deciding whether to wait for a table. Image source: Figure 18.4 of the AI book by S. Russell & P. Novig.

Question

How to induce the decision tree from the training examples?

II. Decision Tree (决策树)

1. Tree Representation
2. **Decision Tree Construction with Heuristics**
   - ❖ **Information Gain: Good Feature Heuristics**
   - ❖ **Information Gain: Continuous Feature**
   - ❖ **Overall: Decision Tree Construction**
3. Tree Overfitting
4. Decision Tree for Regression

Learning Decision Tree is Hard

- Resources: The 12 training examples.
- Aim: Build the smallest tree that classifies the training data correctly.
  - Ockham's razor.
- Challenge: Finding the smallest tree is NP-hard [Hyafil & Rivest'76].

Learning Decision Tree is Hard

- [Question] How many decision trees can be expressed (at least)?

$$
\begin{array}{cc}
\text{Input space} & \text{Output} \\
\left\{
\begin{array}{c}
0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\
0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0 \\
\cdots \\
0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0 \\
\cdots \\
1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1
\end{array}
\right.
&
\left.
\begin{array}{c}
0\ or\ 1 \\
0\ or\ 1 \\
0\ or\ 1 \\
\\
0\ or\ 1 \\
\\
0\ or\ 1
\end{array}
\right\}
\end{array}
$$

- [Answer] $2^{2^{10}}$, super huge search space!

➢Need talented heuristics to guide the search in such a huge space!

Greedy Divide-and-conquer Strategy

- Approach: Greedy divide-and-conquer strategy - heuristic search.
  - (1) Start from empty tree.
  - (2) Decide the best feature based on heuristics.
  - (3) Divide the problem into smaller subproblems;
  - (4) Repeat (2)~(3) until stopping criteria.

➢ Heuristics: Pick the feature that maximizes information gain (信息增益).

- The most informative feature.

Good Feature: Type vs Patrons?

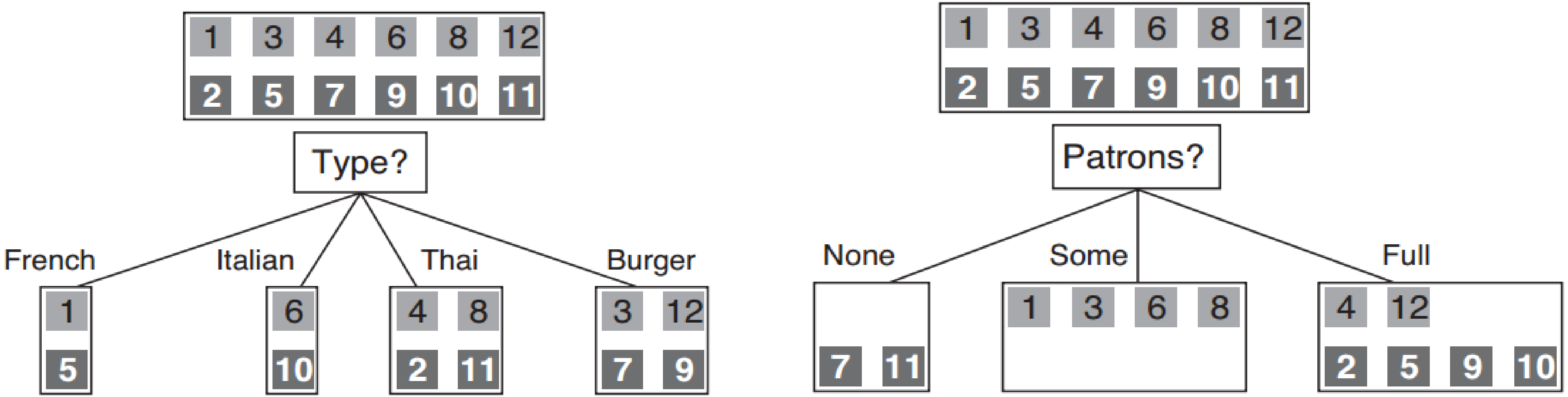


Image source: Figure 18.4 of the AI book by S. Russell & P. Novig.

- [Question] Split the tree based on ‘type’ or ‘patrons’?
- [Answer] ‘patrons’.
- Reason: Divides the 12 data into more distinguishable sub-sets.

Heuristics for Good Feature

- Intuition: More certain about the classification after split regarding this feature.
  - Deterministic (all true or false): perfect
  - Uniform distribution: bad
  - What about in between?

- [Question] How to measure the goodness of a feature formally?
- [Answer] *Information gain*.

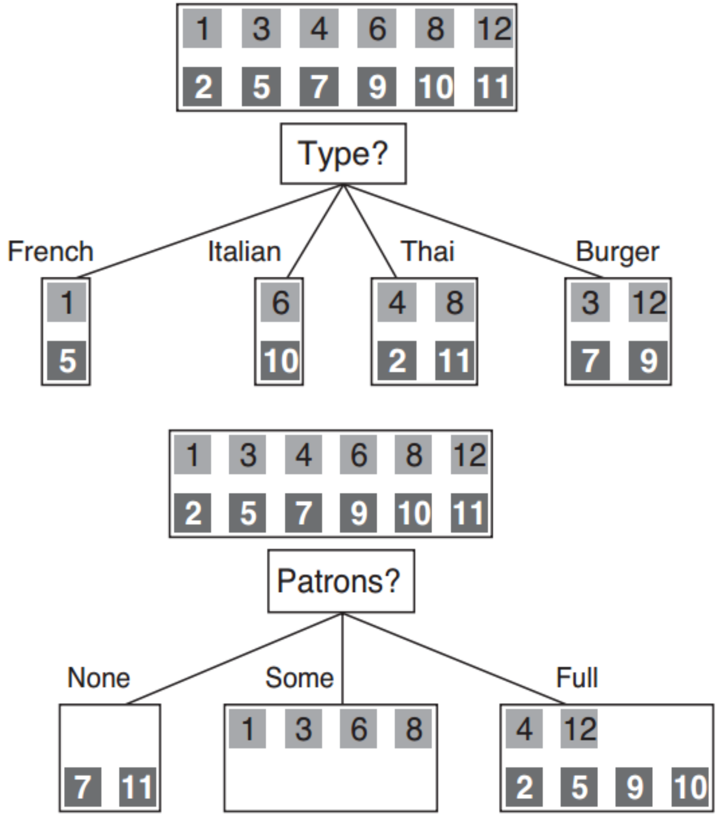


Image source: Figure 18.13 of the AI book by S. Russell & P. Novig.

Preliminary: Entropy (熵)

- Entropy: $\mathcal{H}(Y) \triangleq -\sum_k p(y_k)\log_2 p(y_k)$.

- Larger entropy, more uncertainty.
  - High entropy: $Y \sim$ uniform or flat distribution → less predictable
  - Low entropy: $Y \sim$ peak/valley distribution → more predictable

- Example 1: $\mathcal{H}(Y='label') = -(0.5\log_2 0.5 + 0.5\log_2 0.5) = 1$.

Goal *WillWait*
$y_1 = Yes$
$y_2 = No$
$y_3 = Yes$
$y_4 = Yes$
$y_5 = No$
$y_6 = Yes$
$y_7 = No$
$y_8 = Yes$
$y_9 = No$
$y_{10} = No$
$y_{11} = No$
$y_{12} = Yes$

Image source: Figure 18.3 of the AI book by S. Russell & P. Novig.

Preliminary: Conditional Entropy

- Conditional entropy:

$$\mathcal{H}(Y|X) \triangleq \sum_j p(X = x_j) \cdot \mathcal{H}(Y|X = x_j).$$

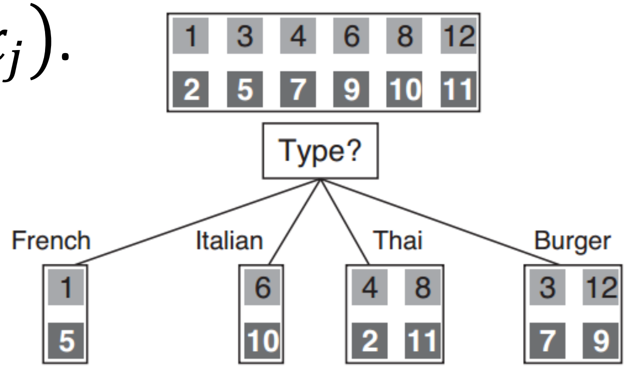


Image source: Figure 18.13 of the AI book by S. Russell & P. Novig.

- Example 2: $Y \sim label \,\&\, X \sim Type$
  - $p(X = French) = p(X = Italian) = \frac{2}{12}$;
  - $p(X = Thai) = p(X = Burger) = \frac{4}{12}$;
  - $\mathcal{H}(Y|X = French\ or\ Italian)$: $-\frac{1}{2}\log\left(\frac{1}{2}\right) - \frac{1}{2}\log\left(\frac{1}{2}\right) = -\log(\frac{1}{2})$;
  - $\mathcal{H}(Y|X = Thai\ or\ Burger)$: $-\frac{2}{4}\log\left(\frac{2}{4}\right) - \frac{2}{4}\log\left(\frac{2}{4}\right) = -\log(\frac{1}{2})$;
  - $\mathcal{H}(Y|X) = -\left[\frac{2}{12} \cdot \log\left(\frac{1}{2}\right) \cdot 2 + \frac{4}{12} \cdot \log\left(\frac{1}{2}\right) \cdot 2\right] = -\log\left(\frac{1}{2}\right) = 1.$

Preliminary: Conditional Entropy

- Conditional entropy:

$$\mathcal{H}(Y|X) \triangleq \sum_j p\left(X = x_j\right) \cdot \mathcal{H}\left(Y|X = x_j\right).$$

- Example 3: $Y \sim label \,\&\, X \sim Patrons$
  - $p(X = None) = \frac{2}{12}; \quad p(X = Some) = \frac{4}{12}; \quad p(X = Full) = \frac{6}{12};$
  - $\mathcal{H}(Y|X = None)$: $\frac{2}{2}\log\left(\frac{2}{2}\right) = 0;$
  - $\mathcal{H}(Y|X = Some)$: $\frac{4}{4}\log\left(\frac{4}{4}\right) = 0;$
  - $\mathcal{H}(Y|X = Full)$: $\frac{2}{6}\log\left(\frac{2}{6}\right) + \frac{4}{6}\log\left(\frac{4}{6}\right) = -0.9183;$
  - $\mathcal{H}(Y|X) = -\left[\frac{2}{12} \cdot 0 + \frac{4}{12} \cdot 0 + \frac{6}{12} \cdot (-0.9183)\right] = 0.4591.$

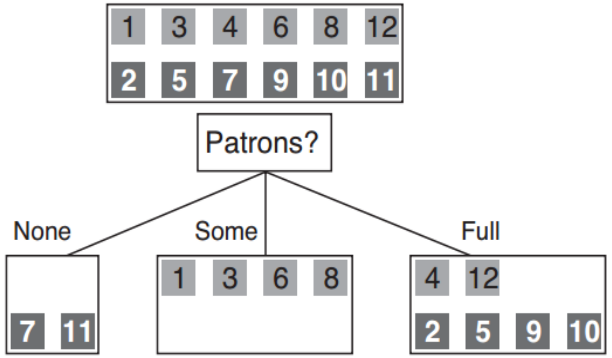


Image source: Figure 18.4(b) of the AI book by S. Russell & P. Novig.

Information Gain (信息增益)

- Information gain: Decrease in entropy after splitting

$$IG(X) = \mathcal{H}(Y) - \mathcal{H}(Y|X)$$

  - X: input feature,
  - Y: classification label.

- Example 4: $Y \sim label \;\&\; X \sim type/patrons$
  - $IG(Type) \quad = \mathcal{H}(Y) - \mathcal{H}(label|Type) = 1 - 1 = 0.$
  - $IG(Patrons) = \mathcal{H}(Y) - \mathcal{H}(label|Patrons) = 1 - 0.4591 = 0.541.$

Information Gain: Type vs Patrons?

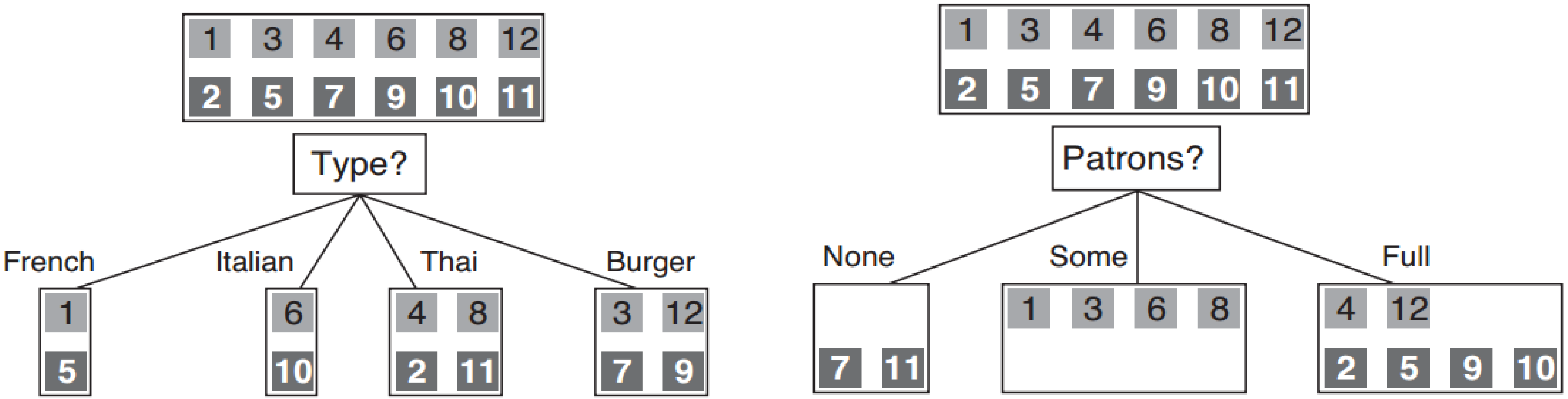


Image source: Figure 18.4 of the AI book by S. Russell & P. Novig.

- [Answer] $IG(Patrons) > IG(Type) \Rightarrow$ Patrons is better.

Continuous Feature Est

- [Question] What should we do for $Est \in \mathbb{R}^1$?
  - Est: estimated waiting time.

- Binary tree: Split on Est at value t,
  - One branch: $\text{Est} < t$,
  - Other branch: $\text{Est} \geq t$.

- Note: Allow repeated features along a path.

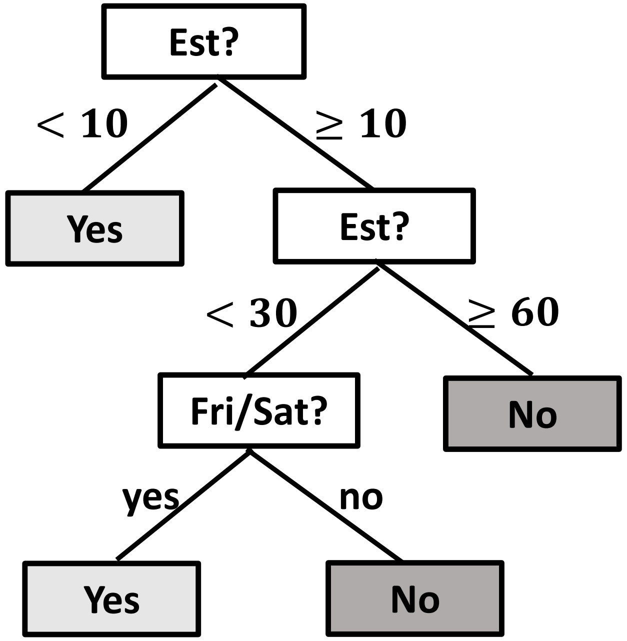


Figure generated by Liyan Song.

Possible $\{t\}$ for Est

- [Question] How to decide the possible $\{t\}$ for Est?
- [Concern] Search through $\mathbb{R}^1 \Rightarrow$ Too hard!
- [Answer] Only a finite number of values are useful:
  - Sort values of Est into $\{x_1, \cdots, x_m\}$ with non-duplicated values;
  - Consider candidates $\left\{t_i = x_i + \frac{x_{i+1} - x_i}{2} \middle| i = 1, \cdots, m-1\right\}$.

Best t^* for Est and its Information Gain

- Take the best t from $\{t\}$: Denote $X \sim Est$,
  - (1) Define $\mathcal{H}(Y|X{:}\,t) = p(X < t) \cdot \mathcal{H}(Y|X < t) + p(X \geq t) \cdot \mathcal{H}(Y|X \geq t)$;
  - (2) Compute $IG(Y|X{:}\,t_i) = \mathcal{H}(Y) - \mathcal{H}(Y|X{:}\,t_i)\ \forall t_i$;
  - (3) Choose $t^* = \arg\max_{t_i} IG(Y|X{:}\,t_i)$

- Use: $IG^*(Est) = IG(Y|X{:}\,t^*) = \max_{t_i} IG(Y|X{:}\,t_i)$.

When to Stop?

- Criterion 1: all records in current subset have the same label.
- Criterion 2: there are no remaining features to help partitioning.
- Criterion 3: The associated dataset is empty.

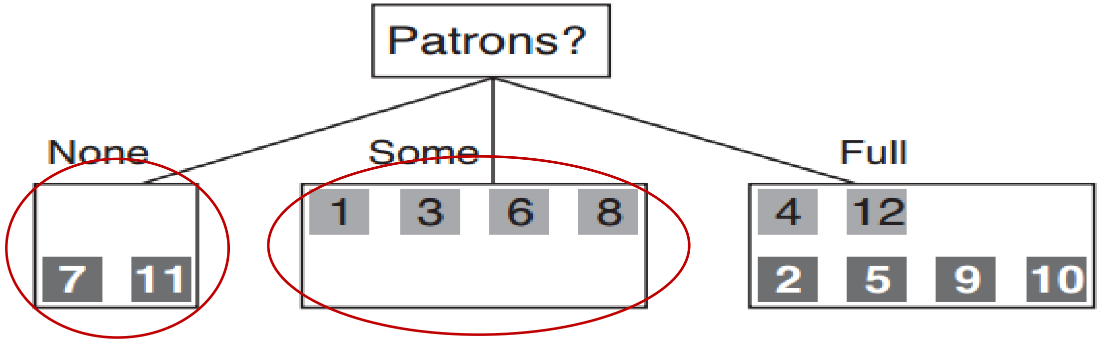


Fig. 1 Criterion 1

Image source: Figure 18.4 of the AI book by S. Russell & P. Novig.

Learning Decision Trees

- Start from empty tree.
- Split on next best feature based on *information gain*.
- Repeat

An Example of Induced Decision Tree

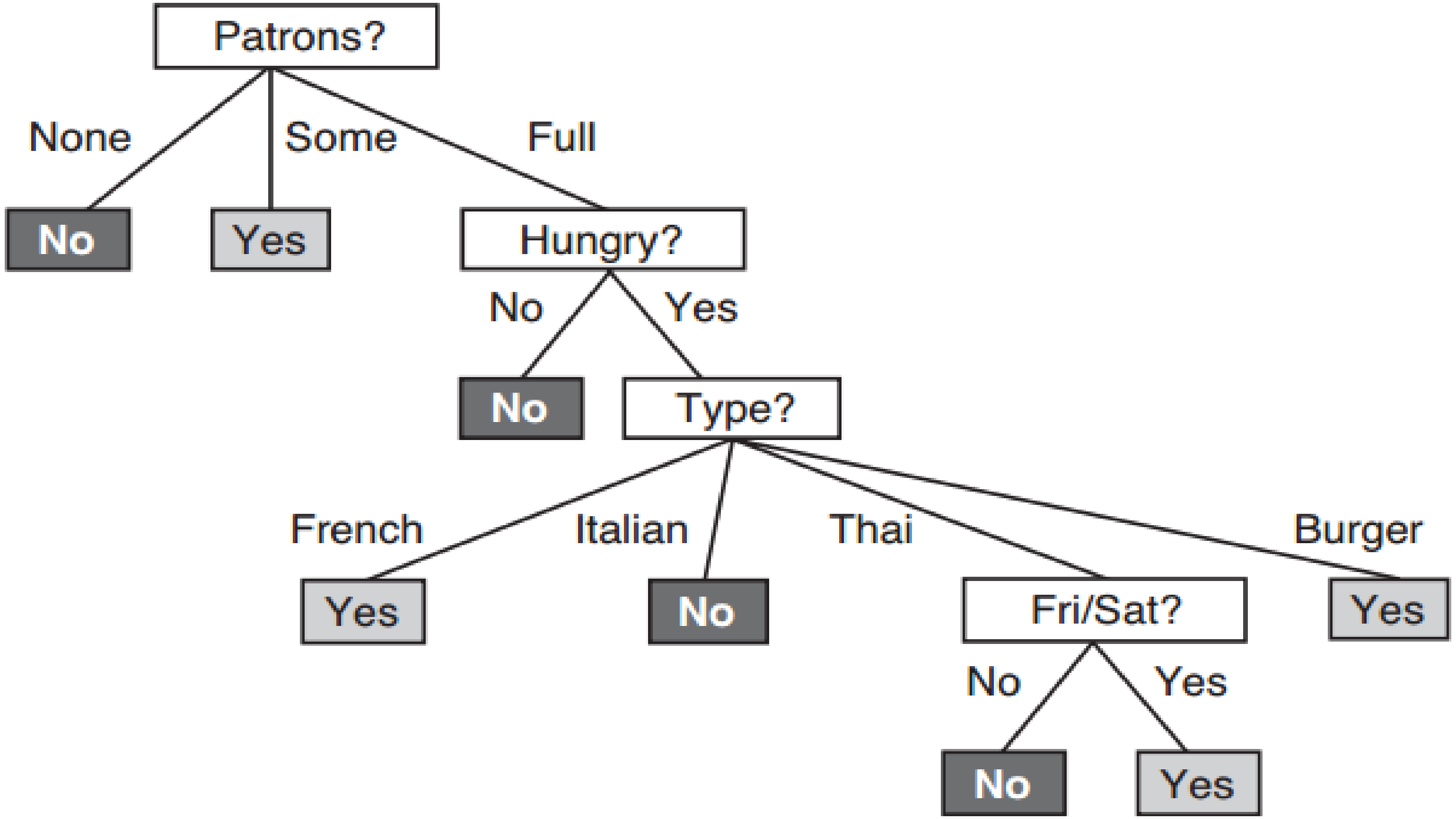


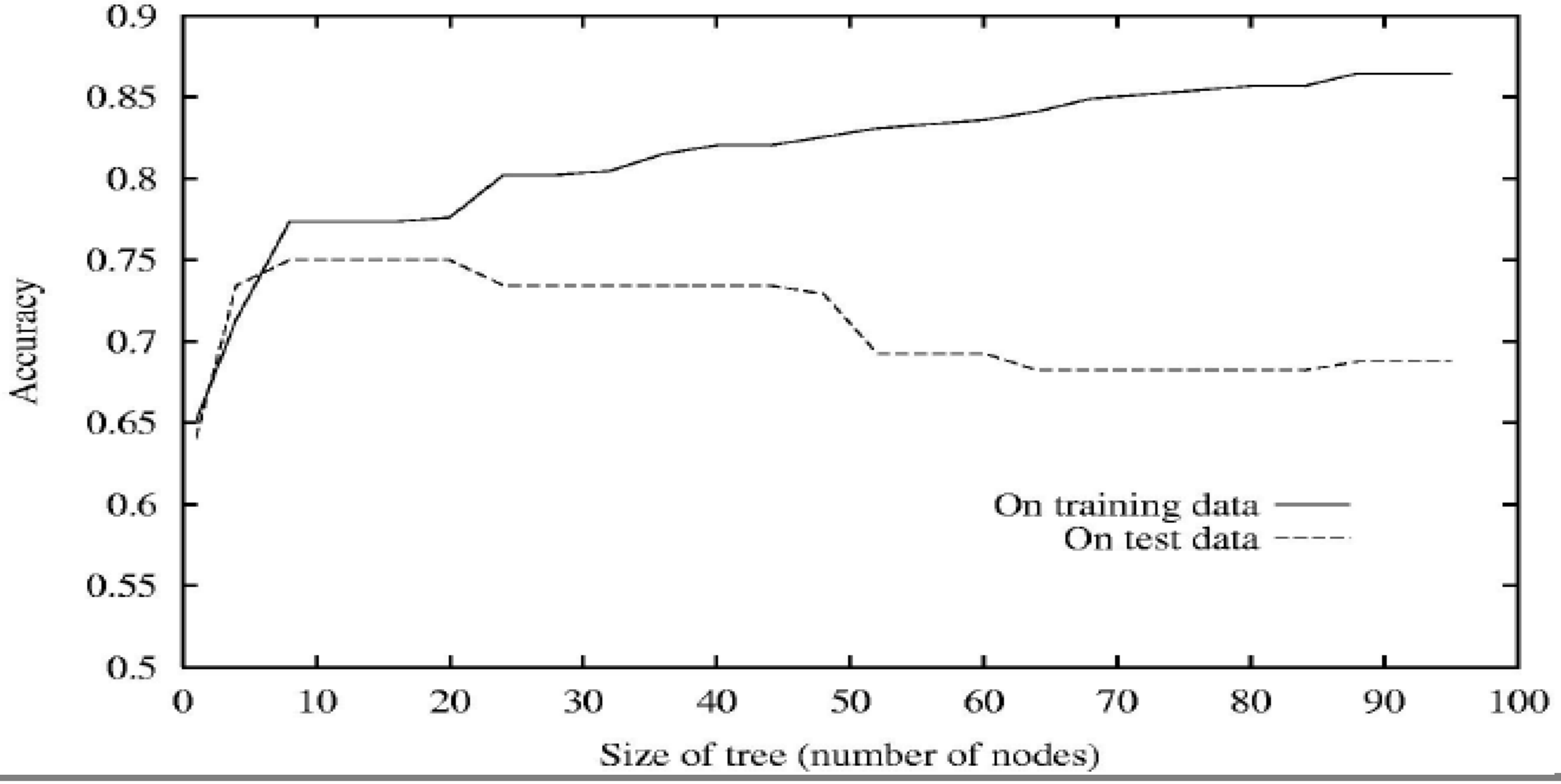
Fig. An estimated decision tree g induced from 12 examples. Image source: Figure 18.6 of the AI book by S. Russell & P. Novig.

II. Decision Tree (决策树)

1. Tree Representation
2. Decision Tree Construction with Heuristics
   - ❖ Information Gain: Good Feature Heuristics
   - ❖ Information Gain: Continuous Feature
   - ❖ Overall: Decision Tree Construction
3. **Tree Overfitting**
4. Decision Tree for Regression

Decision Tree May Overfit

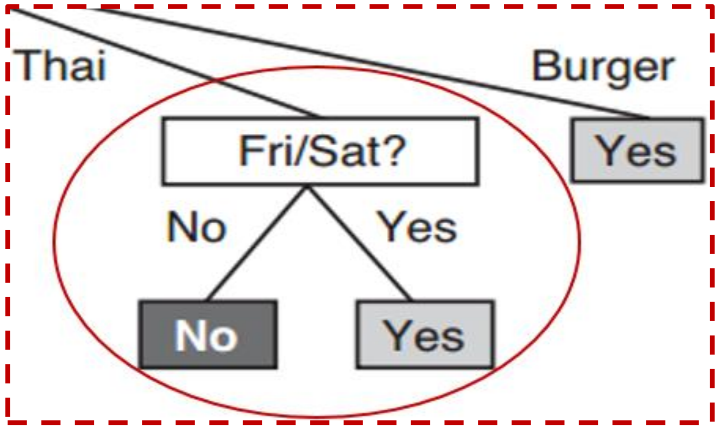
- More #feature, more likely overfitting; more #(train data), less likely overfitting.



Dealing with Overfitting

- Decision tree pruning:
  - (1) Build a fully grown tree.
  - (2) Choose a node that has only leaf nodes as children.
  - (3) Testing the feature ‘relevance’ for this node:
    - (a) relevant→reserve this node.
    - (b) irrelevant: replace it based on its leaf nodes.
  - Repeat (2)~(3) until no such irrelevant nodes.

- Other strategies:
  - Fixed depth
  - Fixed #leaves



A testing node at step (2). Edited from Figure 18.6 of the AI book by S. Russell & P. Novig.

Measure Feature Relevance

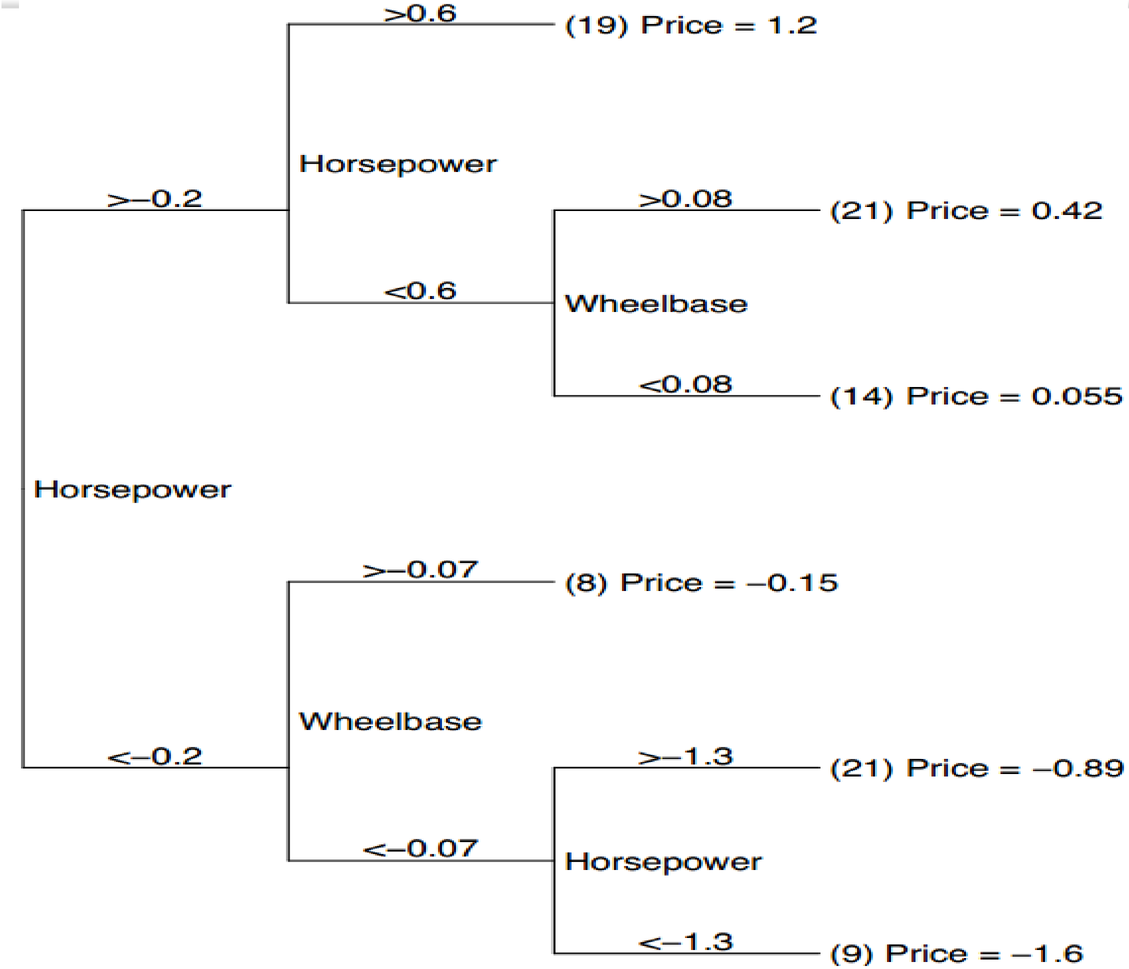
- [Question] How to detect that a node is testing an irrelevant feature?
- [Answer] the node splits the examples evenly & information gain is close to 0 → irrelevant feature.

- [Question] How large a gain should be required to split on the feature?
- [Answer] Using $\mathcal{X}^2$ statistical test, namely $\mathcal{X}^2$ pruning.

II. Decision Tree (决策树)

1. Tree Representation
2. Decision Tree Construction with Heuristics
   - ❖ Information Gain: Good Feature Heuristics
   - ❖ Information Gain: Continuous Feature
   - ❖ Overall: Decision Tree Construction
3. Tree Overfitting
4. **Decision Tree for Regression**

Example: Predict Car Price



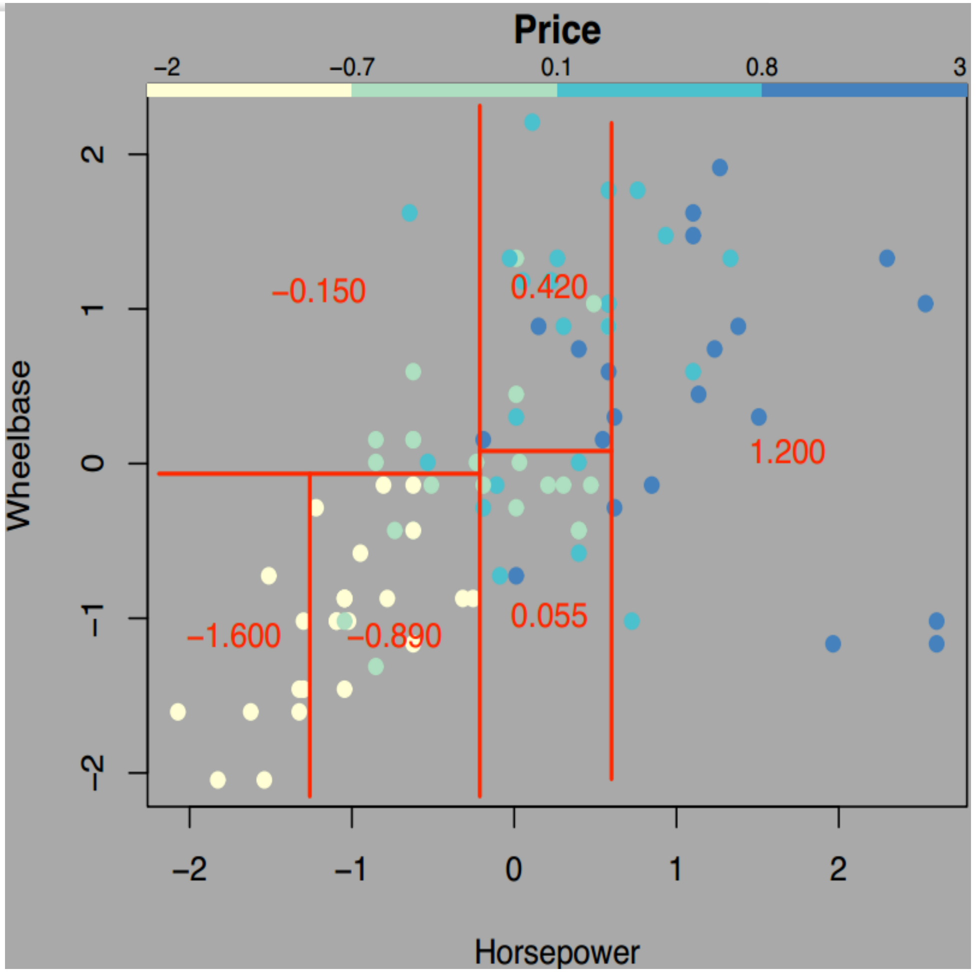


Image source: http://www.stat.cmu.edu/~cshalizi/350-2006/lecture-10.pdf

Regression Tree (RT)

- Construction: Similar to classification trees except:
  - Output: continuous value.
  - Leaf node: (a) mean/median: piecewise constant RT; or (b) a regression model: piecewise linear RT.

Piecewise Constant RT

- Leaf node: Mean of the examples in leaf node C

$$\widehat{y_C} = \frac{1}{||C||} \sum_{i=1}^{||c||} y_i$$

- Algorithms examples: AID, CART.

Piecewise Linear RT

- Leaf node: Linear regression model on the examples in each leaf node.
- Algorithms examples:
  - M5’: (1) Construct a constant regression tree. (2) Fit a linear regression model for each leaf node.
  - GUIDE: (1) Fit a regression model (linear or nonlinear) and compute the residuals. (2) Label the examples with 1 for positive and 2 for negative residuals. (3) Apply the GUIDE for classification tree to split the node.

RT Construction

- Problem: No labels to split features by $IG(X_i)$.
- Feature splitting criteria: Sum of squared error

$$SSE = \sum_{C \in L} \sum_{i \in C} (y_i - \widehat{y_C})^2,$$

  - L: a set of leaf nodes.
  - $\widehat{y_C}$: the estimation on leaf node C from its examples.
  - y_i for $i \in C$: output of the i^{th} example of leaf node C.
- Learning: Search all binary splits that reduce SSE to the full.
- When to stop: $SSE \leq \delta$ or fixed #leaves.

III. Neural Network (神经网络)

Artificial Neural Network: Formulation

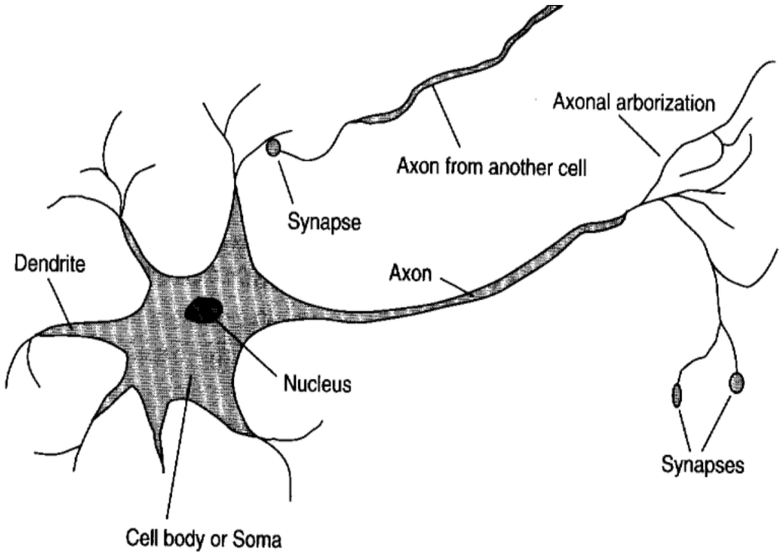


Fig.(1) A brain neuron. Image source: Figure 1.2 of the AI book by S. Russell & P. Novig.

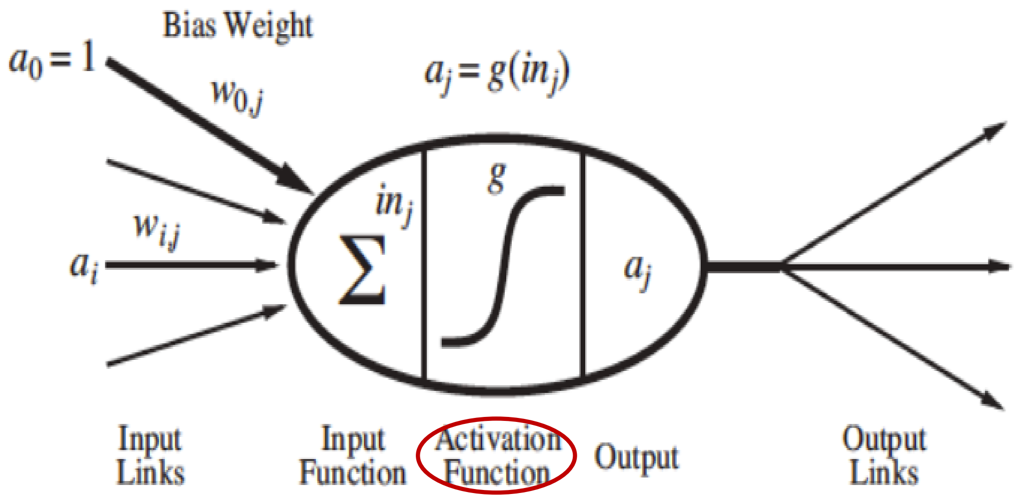
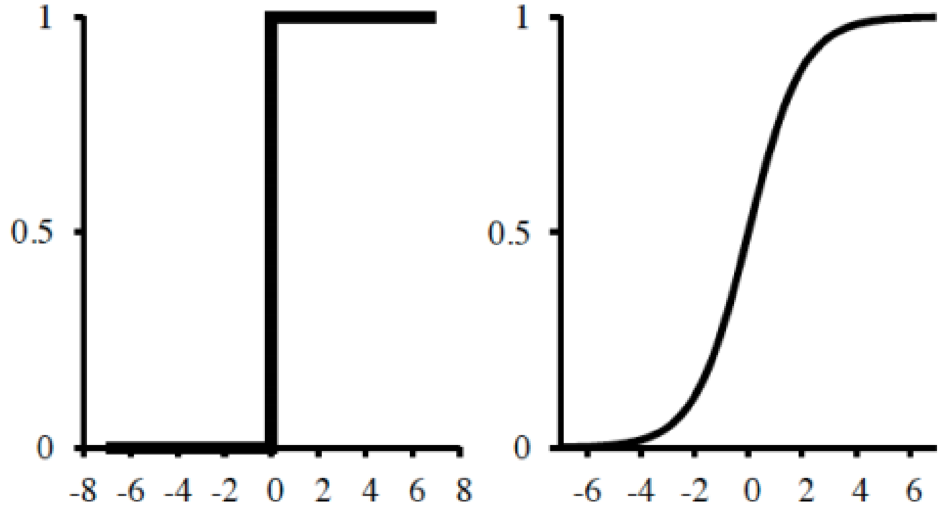


Fig.(2) An artificial neuron. Image source: Figure 18.9 of the AI book by S. Russell & P. Novig.

- $in_j = \sum_{i=0}^{n} w_{i,j} a_i$
- $a_j = g(in_j)$

Activation Function (激活函数)

- Physics: Simulate the activation process of real neuron.
- Math: Nonlinear activation functions encodes the ability to estimate a nonlinear function from inputs to outputs.
- Popular activation functions:
  - hard threshold,
  - logistic function,
  - sigmoid
  - Tanh
  - ReLU, Leaky ReLU
  - ……



hard threshold logistic function

Image source: Figure 18.17 of the AI book by S. Russell & P. Novig.

Single-layer Neural Network

- Single-layer neural network: All input neurons connect directly with the output neurons.

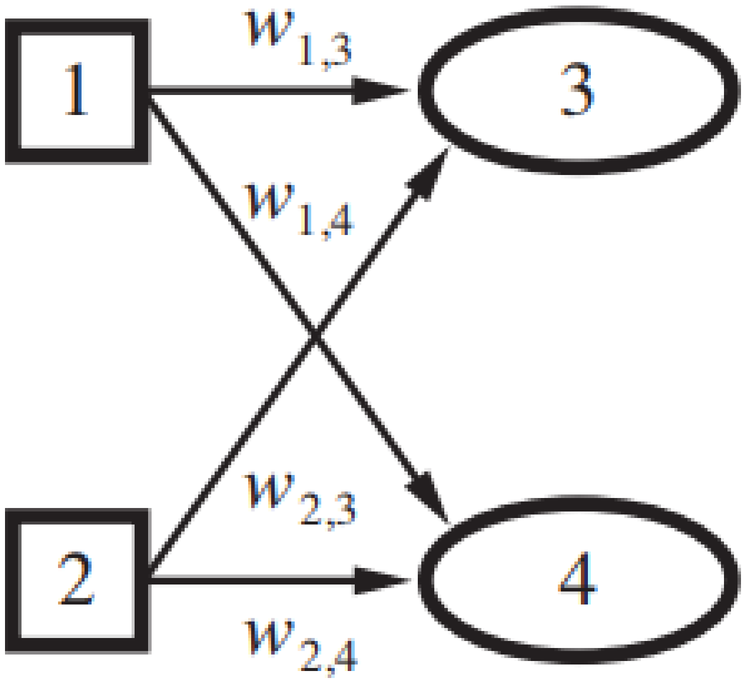
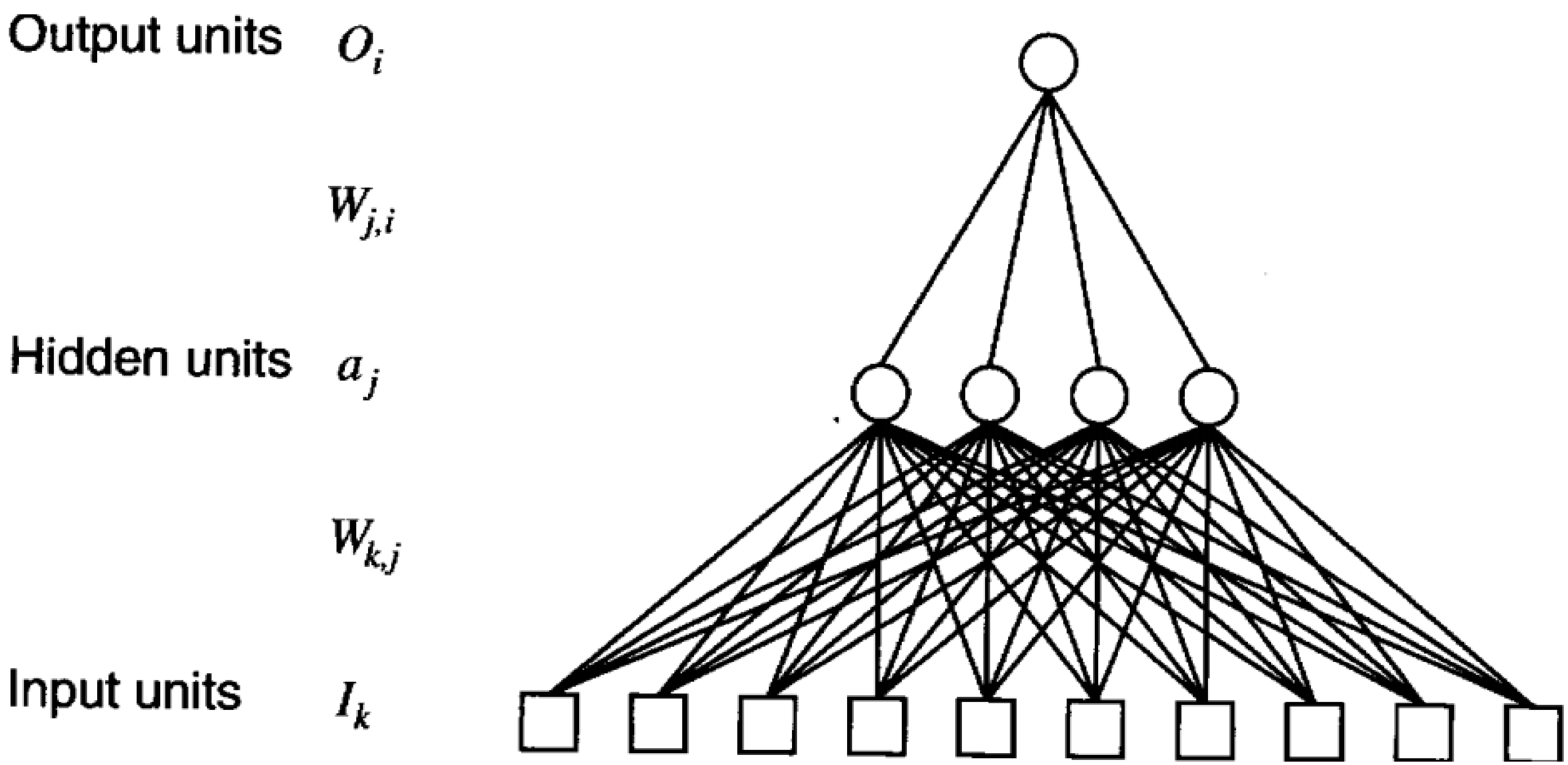


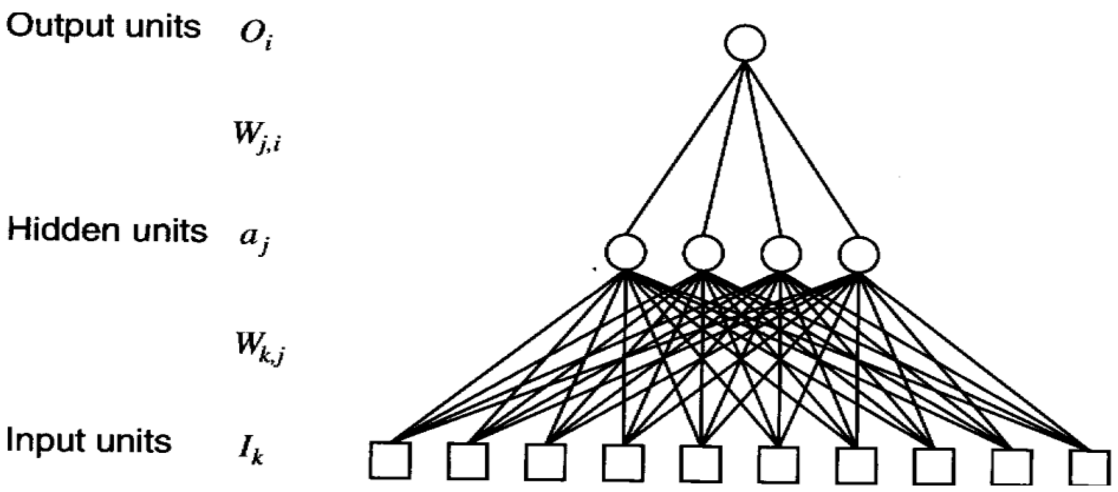
Image source: Figure 18.20.a of the AI book by S. Russell & P. Novig.

Multilayer Neural Network



Multilayer Neural Network

- Notation system:
  - Output index: $1 \cdots i \cdots n$
  - Hidden index: $1 \cdots j \cdots h$
  - Input index: $1 \cdots k \cdots l$
  - Weights between output and hidden: $w_{j,i}$
  - Weight between hidden and input: $w_{k,j}$
  - Activation function of output units: $\sigma()$
  - Activation function of hidden units: $g()$
  - $h_{\boldsymbol{w}}$: the (non-linear) function the NN represents.
- Exercise: express o_i with the above notation system.



Learning Multilayer Networks

- Loss function for an example: $\ell_2(\boldsymbol{w}) = \frac{1}{2}||y - o(\boldsymbol{x})||^2$
  - $(\boldsymbol{x}, y)$: a training example;
  - $o(\boldsymbol{x})$: estimated output for input $\boldsymbol{x}$.
- Partial derivative for any w: 'chain rule'

$$\frac{\partial}{\partial w}\ell_2(\boldsymbol{w}) = \frac{\partial}{\partial w}\frac{1}{2}\sum_i (y_i - o_i)^2 = -\sum_i (y_i - o_i)\frac{\partial o_i}{\partial w} = \cdots$$

- Back propagation (反向传播/逆传播) to train ANN:
  - Gradient descent for $w_{j,i}$ from hidden to output: $w_{j,i} \leftarrow w_{j,i} - \alpha \cdot \frac{\partial}{\partial w_{j,i}}\ell_2(\boldsymbol{w})$
  - Gradient descent for $w_{k,j}$ from input to hidden: $w_{k,j} \leftarrow w_{k,j} - \alpha \cdot \frac{\partial}{\partial w_{k,j}}\ell_2(\boldsymbol{w})$

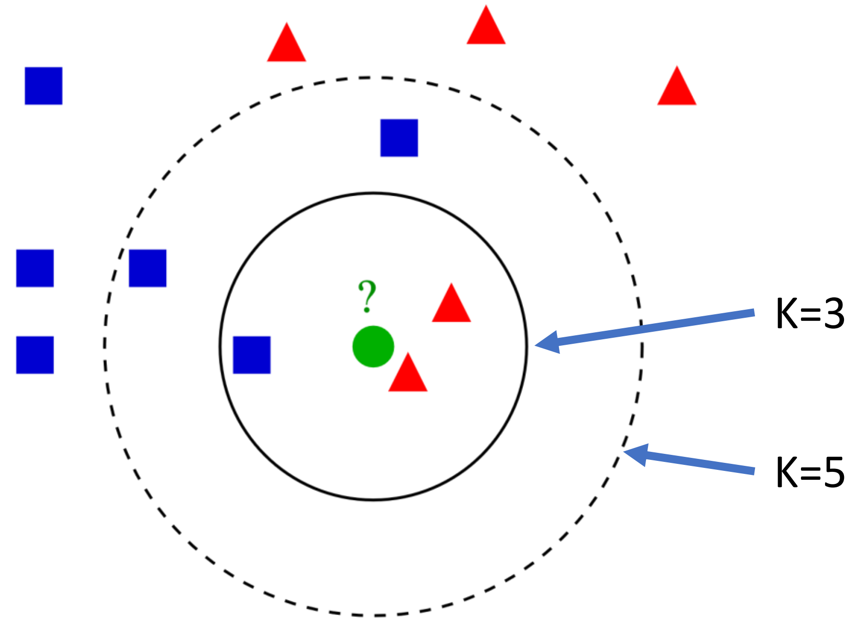
Learning ANN Structures

- Fully connected networks: decide #hidden layers and their sizes using cross-validation.
- Not fully connected networks: optimal brain damage algorithm begins with a fully connected network and removes connections from it.
- Grow a larger network from a smaller one: Subsequent units are added to cater for the examples that the first unit got wrong in the tiling algorithm.

IV. k-Nearest Neighbor (k-近邻)

k-NN

- k-Nearest neighbor method:
  - For classification: find k nearest neighbors of the testing point and take a vote.
  - For regression: take mean or median of the k nearest neighbors, or do a local regression on them.



Example of k-NN classification. Image source: https://en.wikipedia.org/wiki/File:KnnClassification.svg#filelinks.

k-NN Issues

- Distance metric: e.g., $\ell_p(\boldsymbol{x}_j, \boldsymbol{x}_{\boldsymbol{q}}) = \left(\sum_{\boldsymbol{i}} \left|x_{j,i} - x_{q,i}\right|^p\right)^{1/p}$
  - ℓ_1: Manhattan distance,
  - ℓ_2, Euclidean distance.
- Model parameter k: increasing k reduces variance and increases bias.
- Memory-based method: must store all training samples.

k-NN Issues

- Advantage:
  - Training is very fast.
  - Learn complex target functions.
  - Do not lose information.

- Disadvantage:
  - Slow at query time.
  - Easily fooled by irrelevant attributes.

V. Support Vector Machine (支持向量机)

Geometry and SVM Formulation

- Input: $\boldsymbol{x}$
- Output: $y\ (-1\ or\ 1)$
- Model: $\{\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x} + b = 0\}$
- Two Boundaries:

$$\{\boldsymbol{w} \cdot \boldsymbol{x} + b = \pm 1\}$$

- Margin: $d = \frac{2}{||\boldsymbol{w}||^{2}}$

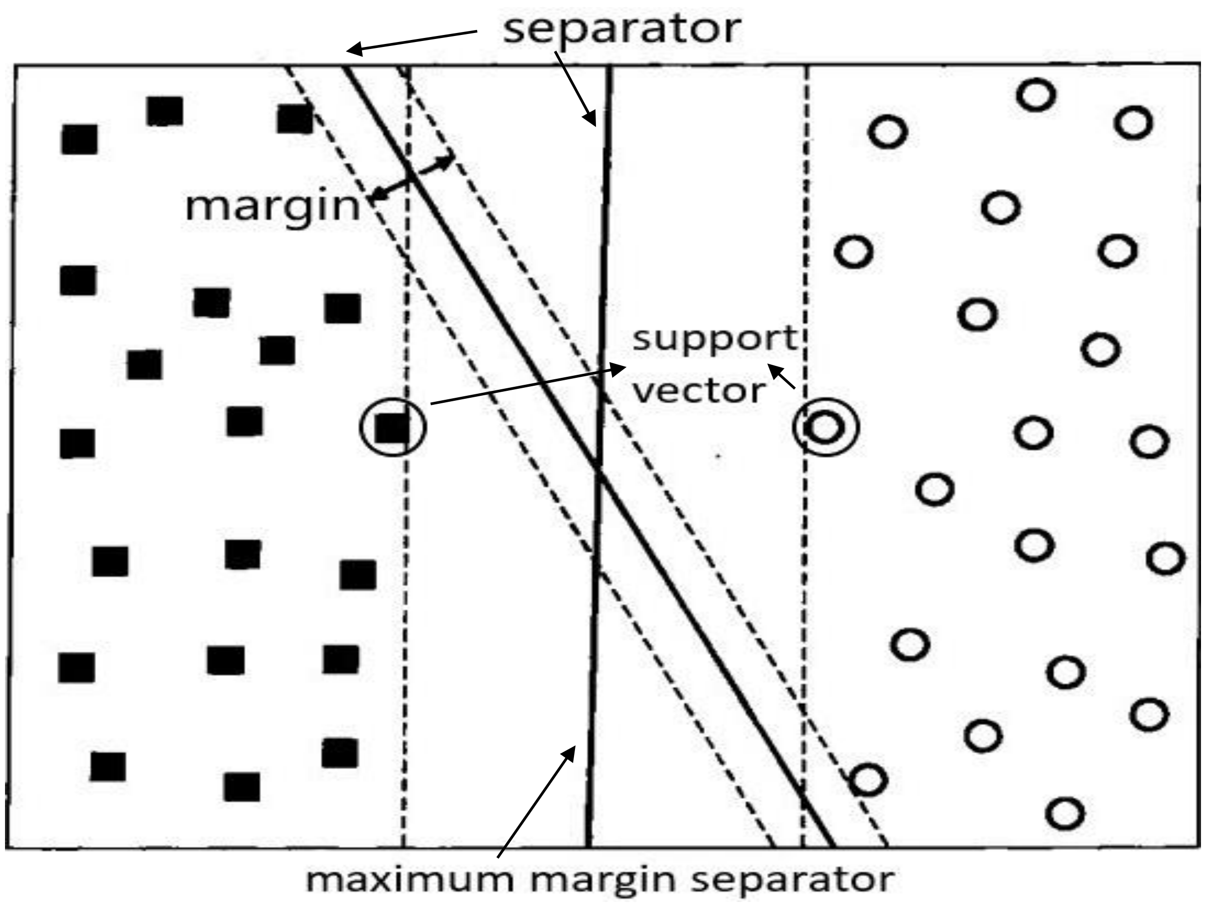


Image source: Figure 5.25 of "Introduction to Data Mining" by P. Tan, M. Steinbach and V. Kumar.

Maximize the Margin

- Training Data: $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$.
- Optimization: maximize the margin with the constraints as

$$\max_{\boldsymbol{w}} \frac{2}{||\boldsymbol{w}||^2},$$
$$s.t.\ [\boldsymbol{w} \cdot \boldsymbol{x}^{(n)} + b] \cdot y_i^{(n)} \geq 1$$

- Learning algorithm [3]:
  - Lagrange multiplier with KKT condition ⇒ Dual representation.
  - Gradient descent.

Kernel Trick

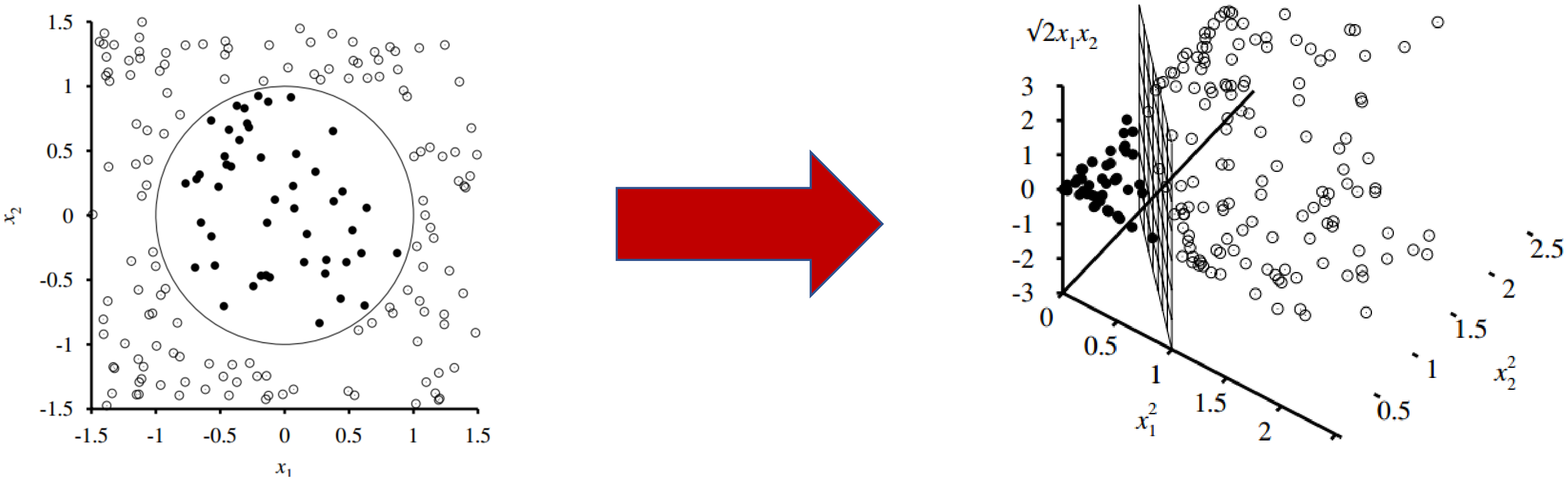


Image source: Figure 18.31 of the AI book by S. Russell & P. Novig.

Kernel trick: nonlinear-separable feature space ⇒ linear-separable one.

Reading Materials for This Lecture

- [1] AI book (P693-748).
- [2] P. Tan, M. Steinbach and V. Kumar. *Introduction to Data Mining* (pages 223-225, 256-276).
- [3] Laurent H, Rivest R L. *Constructing optimal binary decision trees is NP-complete*. Information processing letters, 1976, 5(1): 15-17.
- [4] Gradient Descent: http://ruder.io/optimizing-gradient-descent/
- [5] Classification and Regression Trees: http://www.stat.wisc.edu/~loh/treeprogs/guide/wires11.pdf
- [6] http://scikit-learn.org/stable/modules/ensemble.html