COGS 118A, Winter 2020

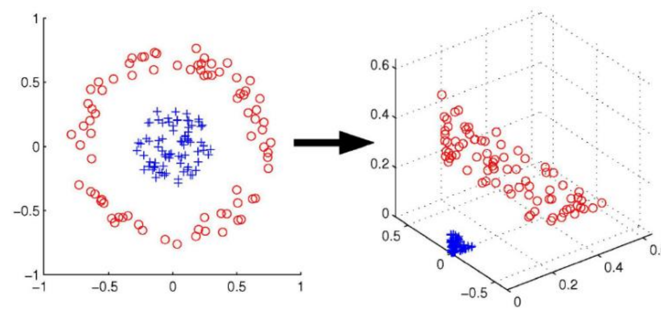
Supervised Machine Learning Algorithms

Lecture 14: Decision Tree Classifier

Zhuowen Tu

Main motivations for applying kernels in machine learning

1. Turn non-separable/difficulty classification problems into separable/easy ones by projecting the original feature space into non-linear (typically higher) dimensions.



original space (non-separable)

kernel space (separable)

2. Turn a parametric (explicit) representation into a non-parametric (implicit) form.

$$sign(\mathbf{w}^T\mathbf{x} + b) \longrightarrow sign(\textstyle\sum_i \alpha_i \times K(\mathbf{x}_i, \mathbf{x}))$$

The kernel trick

non-linear mapping to F

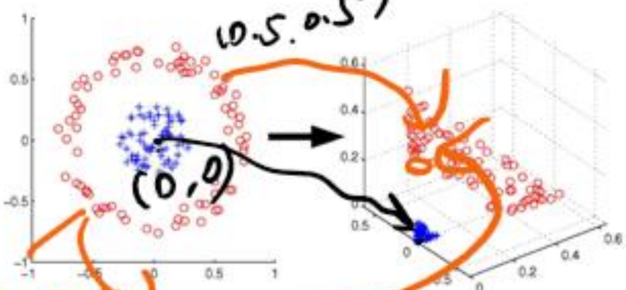
1. high-D space
2. infinite-D countable space :
3. function space (Hilbert space)

needs a careful design

impossible in practice

$$\Phi : \mathbf{x} \rightarrow \phi(\mathbf{x}), \Re^d \rightarrow F$$

(0.5, 0.5)



(0, 0)

(-0.5, -0.5)

example: $(x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1 \times x_2)$

$(0,0) \rightarrow (0,0,0)$ 1.414 $= 0.35$

$(0.5, 0.5) \rightarrow (0.25, 0.25, \sqrt{2} \times 0.25$

$(-0.5, 0.5) \rightarrow (0.25, 0.25, 0.35$

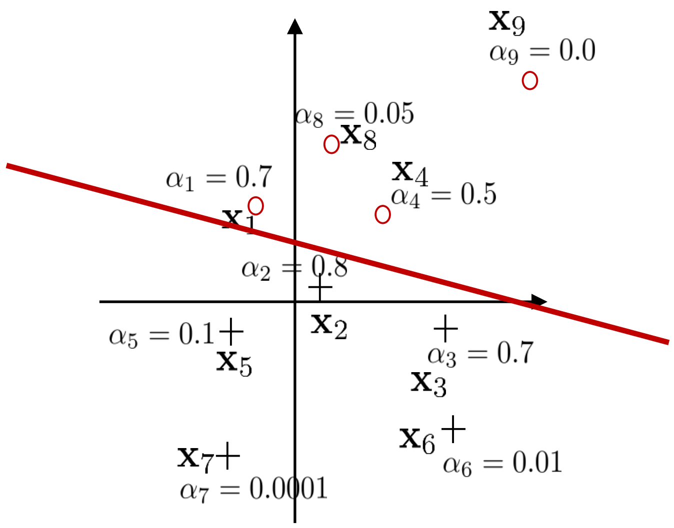
Main motivations for applying kernels in machine learning

Turn a parametric (explicit) representation into a non-parametric (implicit) form.

$$sign(\mathbf{w}^T\mathbf{x} + b) \longrightarrow sign(\textstyle\sum_i \alpha_i \times K(\mathbf{x}_i, \mathbf{x}))$$

$K(\mathbf{x}_i, \mathbf{x})$ measures the “similarity” between $\mathbf{x}_i$ and $\mathbf{x}$ in the kernel space.

$\alpha_i \in \mathbb{R}$ refers to the learned “weight” for each input sample $\mathbf{x}_i$. Samples with large magnitude of α_i are referred to as the Support Vectors in the SVM classifier.



Main motivations for applying kernels in machine learning

Turn a parametric (explicit) representation into a non-parametric (implicit) form.

$$sign(\mathbf{w}^T\mathbf{x} + b) \longrightarrow sign(\textstyle\sum_i \alpha_i \times K(\mathbf{x}_i, \mathbf{x}))$$

$K(\mathbf{x}_i, \mathbf{x})$ measures the "similarity" between $\mathbf{x}_i$ and $\mathbf{x}$ in the kernel space.

There are two strategies to compute $K(\mathbf{x}_i, \mathbf{x})$.

1. If we know the projection function: $\phi(\mathbf{x})$.

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}) &= \phi(\mathbf{x}_i)^T\phi(\mathbf{x}) \\ &\equiv \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}) \\ &\equiv <\phi(\mathbf{x}_i), \phi(\mathbf{x})> \end{aligned}$$

2. If we do not know the projection function, then e.g.

$$K(\mathbf{x}_i, \mathbf{x}) = e^{-||\mathbf{x}_i - \mathbf{x}||^2}$$

It's an implicit function to compute the similarity between $\mathbf{x}_i$ and $\mathbf{x}$, without knowing the projection function $\phi(\mathbf{x})$ explicitly.

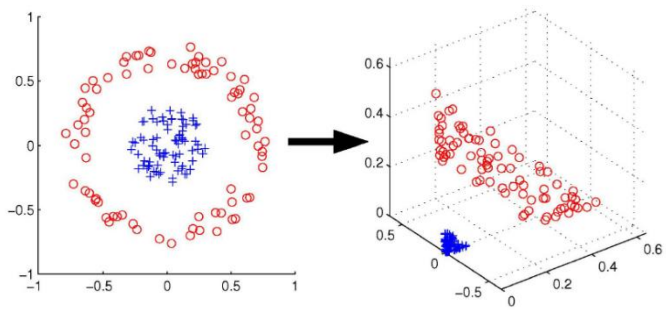
Defining kernels

$$sign(\sum_i \alpha_i \times K(\mathbf{x}_i, \mathbf{x}))$$

There are two strategies to compute $K(\mathbf{x}_i, \mathbf{x})$

1. If we know the projection function: $\phi(\mathbf{x})$.

$$K(\mathbf{x}_i, \mathbf{x}) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) \equiv \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) \equiv < \phi(\mathbf{x}_i), \phi(\mathbf{x}) >$$



original space kernel space

Example:

$\mathbf{x} = (x_1, x_2)$

non-separable

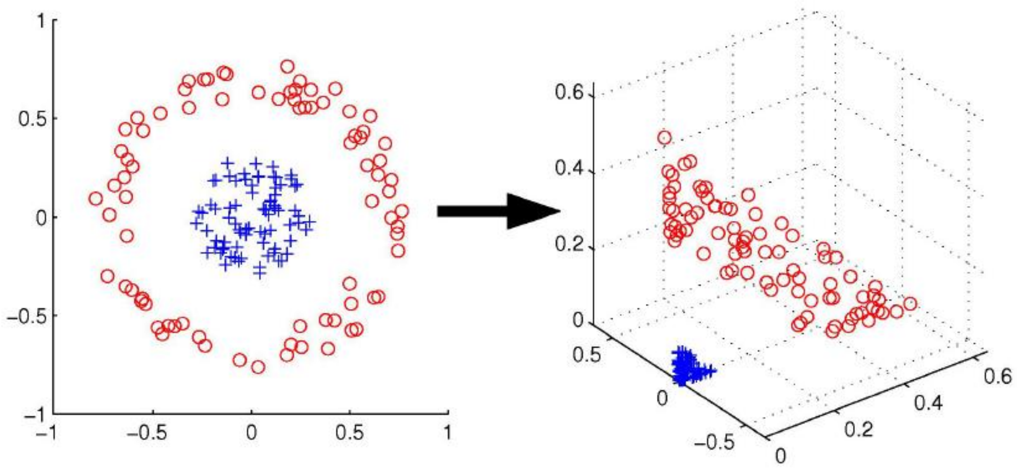
$\phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1 \times x_2)$

separable

The kernel trick

non-linear mapping to F
1. high-D space
2. infinite-D countable space :
3. function space (Hilbert space)

$$\Phi : \mathbf{x} \rightarrow \phi(\mathbf{x}), \Re^d \rightarrow F$$



example: $(x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1 \times x_2)$

SVMs: the kernel trick

Problem: the dimension of $\Phi(\mathbf{x})$ can be very large, making w hard to represent explicitly in memory, and hard for the QP to solve.

The Representer theorem (Kimeldorf & Wahba, 1971) shows that (for SVMs as a special case):

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i)$$

for some variables α. Instead of optimizing w directly we can thus optimize α.

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x})$$

We call $K(\mathbf{x}_i, \mathbf{x}) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) =< \phi(\mathbf{x}_i), \phi(\mathbf{x}) >$ the kernel function.

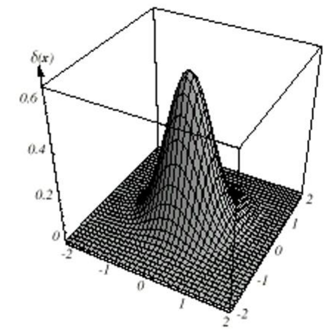
Defining kernels

$$sign(\sum_i \alpha_i \times K(\mathbf{x}_i, \mathbf{x}))$$

There are two strategies to compute $K(\mathbf{x}_i, \mathbf{x})$

2. If we do not know the projection function, then e.g.

$$K(\mathbf{x}_1, \mathbf{x}_2) = e^{-||\mathbf{x}_1 - \mathbf{x}_2||^2/c}$$

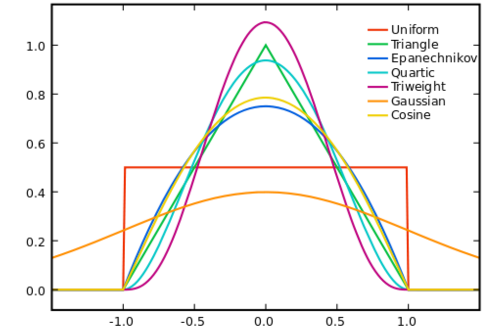


More kernel functions:
http://crsouza.com/2010/03/17/kernel-functions-for-machine-learning-applications/

$$K(\mathbf{x}_1, \mathbf{x}_2) = (< \mathbf{x}_1, \mathbf{x}_2 > + \theta)^d$$

$$K(\mathbf{x}_1, \mathbf{x}_2) = tanh(\alpha < \mathbf{x}_1, \mathbf{x}_2 > + \theta)$$

$$K(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{||\mathbf{x}_1 - \mathbf{x}_2||^2 + c^2}}$$



https://en.wikipedia.org/wiki/Kernel_(statistics)

Understanding the kernel

Turn a parametric (explicit) representation into a non-parametric (implicit) form.

$$sign(\mathbf{w}^T\mathbf{x} + b) \longrightarrow sign(\textstyle\sum_i \alpha_i \times K(\mathbf{x}_i, \mathbf{x}))$$

Next: to see how is the solution in Kernel SVM obtained.

This can be simply illustrated in the Ridge Regression Classifier.

An example

Let's look at a simpler case (ridge regression) with constant λ:

$$\text{Find: } \arg\min_{\mathbf{w}} \ \ \lambda ||\mathbf{w}||^2 + \sum_{i=1}^{n} (y_i - \mathbf{w} \cdot \mathbf{x}_i)^2$$

A. It is convex and also has a closed-form (analytic) solution. Using gradient descent is fine too.

B. It is convex but without a closed-form solution.

C. It is non-convex but can be solved using gradient descent.

D. It is non-convex and has no clear solutions.

Ridge regression and kernel

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

$$\mathbf{w}^* = (X^T X + \lambda I_m)^{-1} X^T Y$$

$$\mathbf{w}^* = \arg min_{\mathbf{w}} = (X\mathbf{w} - Y)^T (X\mathbf{w} - Y) + \lambda \mathbf{w}^T \mathbf{w}$$

$$(X^T X + \lambda I)\mathbf{w}^* = X^T Y$$

$$\mathbf{w}^* = \frac{1}{\lambda}(X^T Y - X^T X \mathbf{w}^*)$$

$$\Downarrow$$

$$= X^T \mathbf{a}$$

$$\Downarrow$$

$$\mathbf{a} = \frac{1}{\lambda}(Y - X\mathbf{w}^*)$$

$$\Downarrow$$

$$\lambda \mathbf{a} = (Y - X X^T \mathbf{a})$$

$$\Downarrow$$

$$X X^T \mathbf{a} + \lambda \mathbf{a} = Y$$

$$\Downarrow$$

$$(X X^T + \lambda I)\mathbf{a} = Y$$

$$\Downarrow$$

$$\mathbf{a} = (X X^T + \lambda I)^{-1} Y$$

$$\Downarrow$$

$$\mathbf{w}^* = X^T (X X^T + \lambda I_n)^{-1} Y$$

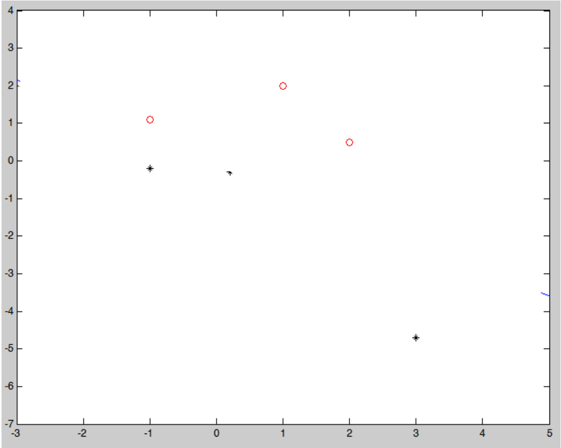
The difference between

$$X^T X \text{ and } X X^T$$

$$S_{training} = \{(\mathbf{x}_i, y_i), i = 1..n\}$$

Feature space: $\mathbf{w}^* = (X^T X + \lambda I_m)^{-1} X^T Y$

Kernel space: $\mathbf{w}^* = X^T (X X^T + \lambda I_n)^{-1} Y$



$$X = \begin{pmatrix} 1.0 & 2.0 \\ 2.0 & 0.5 \\ -1.0 & 1.1 \\ -1.0 & -0.2 \\ 3.0 & -4.5 \\ 0.2 & -0.29 \end{pmatrix}$$

$$Y = \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix}$$

$$X^T X = \begin{pmatrix} 1.0 & 2.0 & -1.0 & -1.0 & 3.0 & 0.2 \\ 2.0 & 0.5 & 1.1 & -0.2 & -4.5 & -0.29 \end{pmatrix} \begin{pmatrix} 1.0 & 2.0 \\ 2.0 & 0.5 \\ -1.0 & 1.1 \\ -1.0 & -0.2 \\ 3.0 & -4.5 \\ 0.2 & -0.29 \end{pmatrix} = \begin{pmatrix} 16.04 & -11.46 \\ -11.45 & 25.83 \end{pmatrix}$$

$$X X^T = \begin{pmatrix} 1.0 & 2.0 \\ 2.0 & 0.5 \\ -1.0 & 1.1 \\ -1.0 & -0.2 \\ 3.0 & -4.5 \\ 0.2 & -0.29 \end{pmatrix} \begin{pmatrix} 1.0 & 2.0 & -1.0 & -1.0 & 3.0 & 0.2 \\ 2.0 & 0.5 & 1.1 & -0.2 & -4.5 & -0.29 \end{pmatrix} = \begin{pmatrix} 5. & 3 & 1.2 & -1.4 & -6. & -0.38 \\ 3. & 4.25 & -1.45 & -2.1 & 3.75 & 0.255 \\ 1.2 & -1.45 & 2.21 & 0.78 & -7.95 & -0.52 \\ -1.4 & -2.1 & 0.78 & 1.04 & -2.1 & -0.14 \\ -6. & 3.75 & -7.95 & -2.1 & 29.25 & 1.91 \\ -0.38 & 0.26 & -0.52 & -0.14 & 1.9 & 0.12 \end{pmatrix}$$

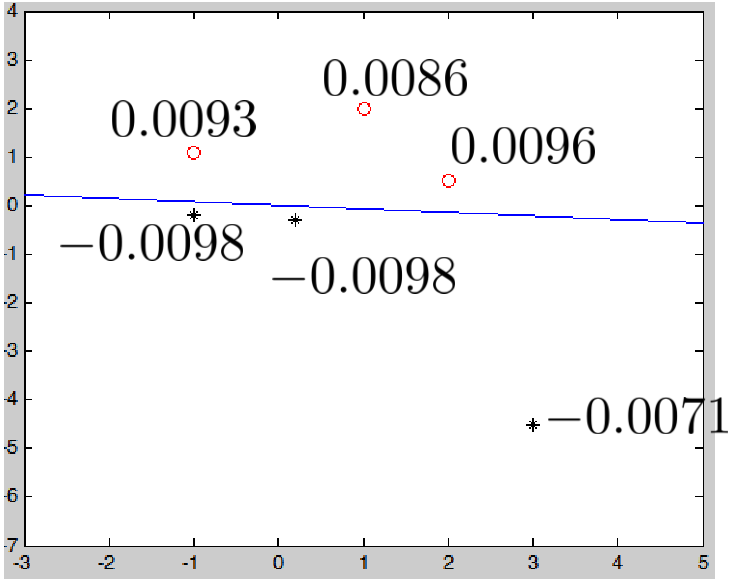
An example

Let's look at a simpler case (ridge regression) with constant λ:

Find: $\arg\min_{\mathbf{w}} \quad \lambda||\mathbf{w}||^2 + \sum_{i=1}^{n}(y_i - \mathbf{w}\cdot\mathbf{x}_i)^2$

$$\mathbf{w}^* = X^T(G + \lambda I_n)^{-1}Y$$

$$\mathbf{w}^* = \sum_i \alpha_i \times \mathbf{x}_i$$



$$X = \begin{pmatrix} 1.0 & 2.0 \\ 2.0 & 0.5 \\ -1.0 & 1.1 \\ -1.0 & -0.2 \\ 3.0 & -4.5 \\ 0.2 & -0.29 \end{pmatrix} \qquad Y = \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix} \qquad \lambda = 100$$

```
w=X'*inv(X*X'+100*eye(6))*Y;
```

$$\mathbf{w}^* = \begin{pmatrix} 0.0051 \\ 0.0687 \end{pmatrix} \qquad \alpha = \begin{pmatrix} 0.0086 \\ 0.0096 \\ 0.0093 \\ -0.0098 \\ -0.0071 \\ -0.0098 \end{pmatrix}$$

Explanations of duality

Let's look at a simpler case (ridge regression) with constant λ:

Find: $\arg\min_{\mathbf{w}} \quad \lambda||\mathbf{w}||^2 + \times \sum_{i=1}^{n}(y_i - \mathbf{w}\cdot\mathbf{x}_i)^2$

Ridge regression: $\alpha = (C \times (XX^T) + I)^{-1}Y$

$$\mathbf{w}^* = \sum_i \alpha_i \times \mathbf{x}_i$$

Learded classifier: $\text{sign}(\mathbf{w}^* \cdot \mathbf{x}) = sign(\sum_i \alpha_i \times \mathbf{x}_i \cdot \mathbf{x})$

Explanations of duality

$$\mathbf{w}^* = \sum_i \alpha_i \times \mathbf{x}_i$$

Learded classifier: $\mathrm{sign}(\mathbf{w} \cdot \mathbf{x}) = sign(\sum_i \alpha_i \times \mathbf{x}_i \cdot \mathbf{x})$

A magic here is in training:
we only need to know $\mathbf{x_i} \cdot \mathbf{x_j} =< \mathbf{x_i}, \mathbf{x_j} >, \forall i, j$

In testing:
we only need to know $\mathbf{x_i} \cdot \mathbf{x} =< \mathbf{x_i}, \mathbf{x} >, \forall i$

The original feature representation of $\mathbf{x}_i$ and $\mathbf{x}$ can be implicit.

SVM

Primal: Find: $\arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + C \times \sum_{i=1}^{n}(1 - y_i \times (\mathbf{w} \cdot \mathbf{x}_i + b))_+$

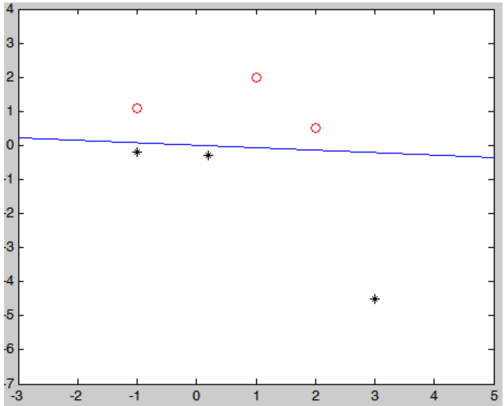
Dual: Find $\arg\max_{\alpha_1,\ldots\alpha_n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} \alpha_j \alpha_i Q_{ij}$

where $Q_{ji} = y_j y_i K(\mathbf{x}_j, \mathbf{x}_i)$ note: $\mathbf{x}_j \cdot \mathbf{x}_i$ is replaced by a more general form, kernel

Subject to constraints: $0 \leq \alpha_i \leq C, \forall i$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$

$$\boxed{\begin{array}{l} \mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \\[2em] b^* = y_k(1 - \varepsilon_k) - \mathbf{w}^* \cdot \mathbf{x}_k \quad \text{where } k = \arg\max_k \alpha_k \end{array}}$$

Note α_i^* and y_i are scalar. $\mathbf{x}_i$ is data vector.

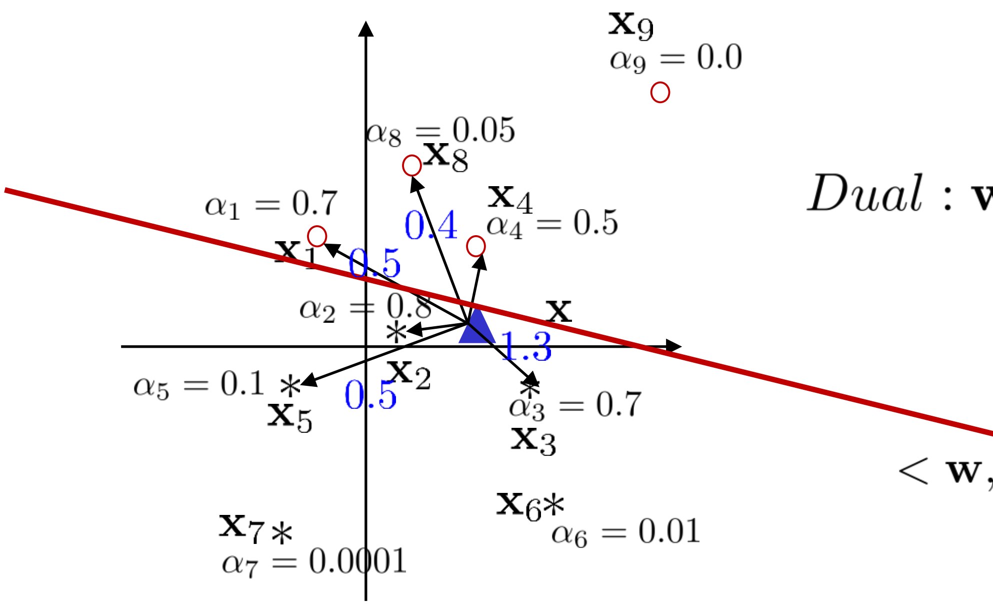


Most α_is are 0 and we only save non-zero data samples, which are the support vectors of our learned classifier.

Learned classifier: $sign(\sum_i \alpha_i y_i \times K(\mathbf{x}_i, \mathbf{x}))$

Understanding SVM?

Find: $\arg\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C \times \sum_{i=1}^{n}(1 - y_i \times (\mathbf{w}\cdot\mathbf{x}_i + b))_+$



$$Dual: \mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i$$

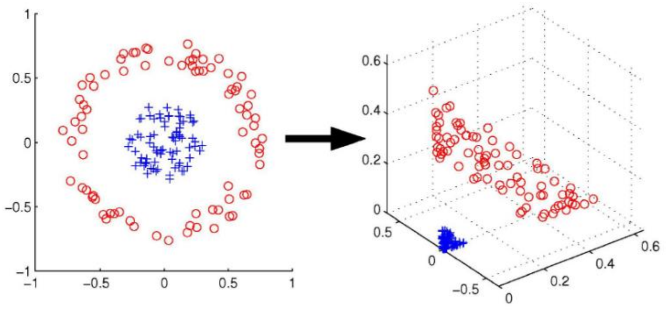
In testing: given an input data $\mathbf{x}$, make the prediction based on $sign(< \mathbf{w}, \mathbf{x} > +b) = sign(\sum_{i=1}^{n} \alpha_i y_i < \mathbf{x}_i, \mathbf{x} >)$

$$< \mathbf{w}, \mathbf{x} > +b = 0.7 \times (+1) \times 0.5 + 0.8 \times (-1) \times 1.5 + 0.7 \times (-1) \times 1.3 + 0.5 \times (+1) \times 0.7 + 0.1 \times (-1) \times 0.5 + 0.05 \times (+1) \times 0.4$$
$$= -1.44$$

$$sign(< \mathbf{w}, \mathbf{x} > +b) = -1$$

Main motivations for applying kernels in machine learning

1. Turn non-separable/difficulty classification problems into separable/easy ones by projecting the original feature space into non-linear (typically higher) dimensions.

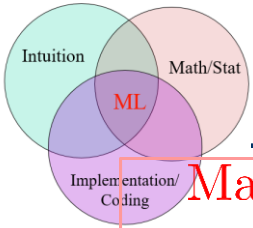


original space (non-separable)

kernel space (separable)

2. Turn a parametric (explicit) representation into a non-parametric (implicit) form.

$$sign(\mathbf{w}^T\mathbf{x} + b) \longrightarrow sign(\textstyle\sum_i \alpha_i \times K(\mathbf{x}_i, \mathbf{x}))$$



Recap: Kernel-based Support Vector Machine

Math:

Training : Minimize $\mathcal{L}(\mathbf{w}, b) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}(1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))_+$

$\Longrightarrow$ Find $\arg\max_{\alpha_1,\ldots\alpha_n} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n}\alpha_j\alpha_i Q_{ij}$

$$\mathbf{w}^* = \sum_{i=1}^{n}\alpha_i\mathbf{x}_i$$

$$b^* = y_k(1-\varepsilon_k) - \mathbf{w}\cdot\mathbf{x}_k \qquad \text{where } k = \arg\max_i \alpha_i$$

Ridge regression: $\alpha = (C \times (XX^T) + I)^{-1}Y$

Testing : Learned classifier: $sign(\sum_i \alpha_i K(\mathbf{x}_i, \mathbf{x}))$

Pros:

- It is very robust.
- Works very well in practice.
- Mathematically well-defined and can be extended to many places.

Cons:

- No intrinsic feature selection stage.
- May not be able to deal with large amount training data with high dimension due to its kernel.

Empirical Comparisons of Different Algorithms

Caruana and Niculesu-Mizil, ICML 2006

MODEL	1ST	2ND	3RD	4TH	5TH	6TH	7TH	8TH	9TH	10TH
BST-DT	0.580	0.228	0.160	0.023	0.009	0.000	0.000	0.000	0.000	0.000
RF	0.390	0.525	0.084	0.001	0.000	0.000	0.000	0.000	0.000	0.000
BAG-DT	0.030	0.232	0.571	0.150	0.017	0.000	0.000	0.000	0.000	0.000
SVM	0.000	0.008	0.148	0.574	0.240	0.029	0.001	0.000	0.000	0.000
ANN	0.000	0.007	0.035	0.230	0.606	0.122	0.000	0.000	0.000	0.000
KNN	0.000	0.000	0.000	0.009	0.114	0.592	0.245	0.038	0.002	0.000
BST-STMP	0.000	0.000	0.002	0.013	0.014	0.257	0.710	0.004	0.000	0.000
DT	0.000	0.000	0.000	0.000	0.000	0.000	0.004	0.616	0.291	0.089
LOGREG	0.000	0.000	0.000	0.000	0.000	0.000	0.040	0.312	0.423	0.225
NB	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.030	0.284	0.686

Overall rank by mean performance across problems and metrics (based on bootstrap analysis).

BST-DT: boosting with decision tree weak classifier

RF: random forest

BAG-DT: bagging with decision tree weak classifier

SVM: support vector machine

ANN: neural nets

KNN: k nearest neighboorhood

BST-STMP: boosting with decision stump weak classifier

DT: decision tree

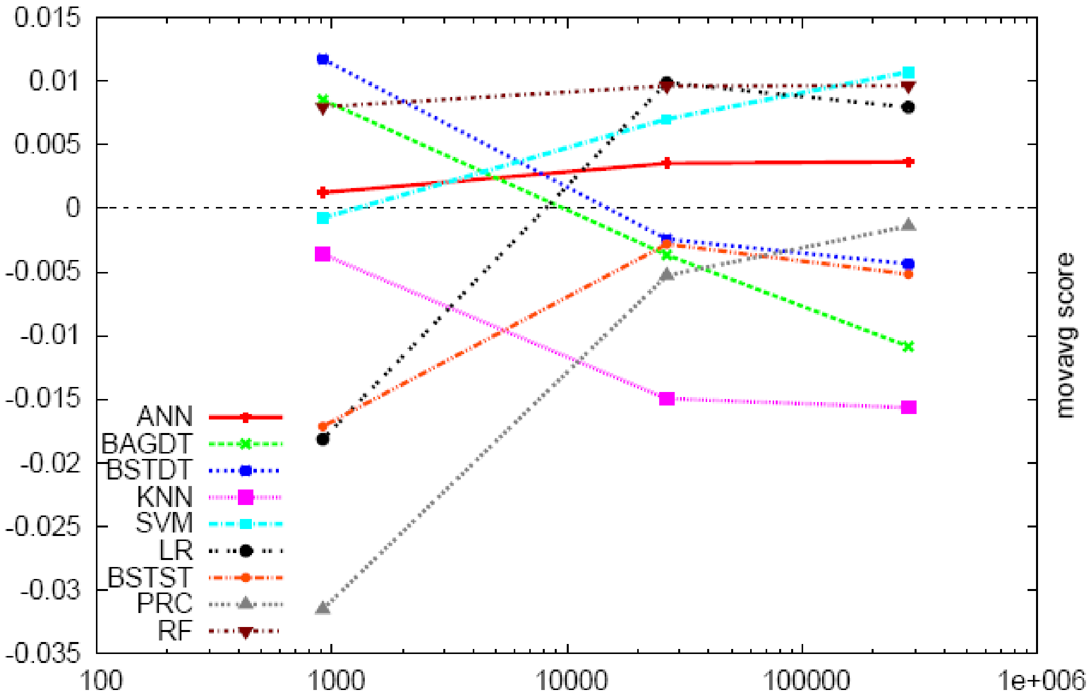
LOGREG: logistic regression

NB: naïve Bayesian

It is informative, but by no means final.

Empirical Study on High-dimension

Caruana et al., ICML 2008



Moving average standardized scores of each learning algorithm as a function of the dimension.

The rank for the algorithms to perform consistently well:

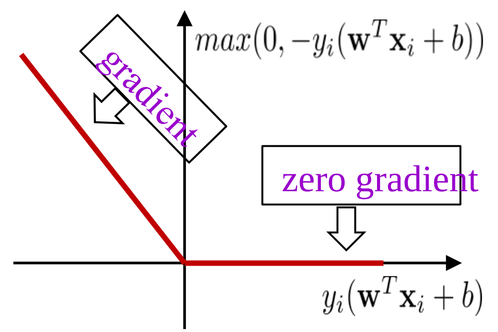
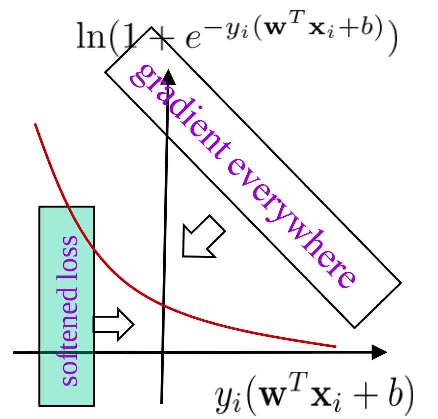
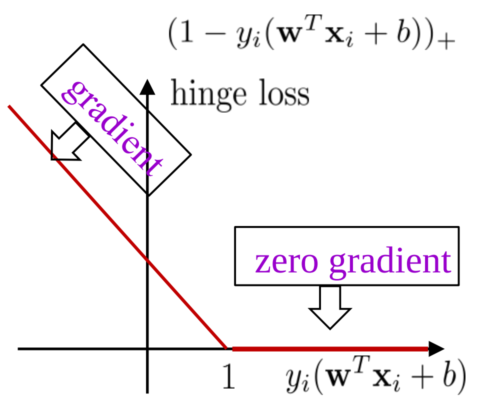
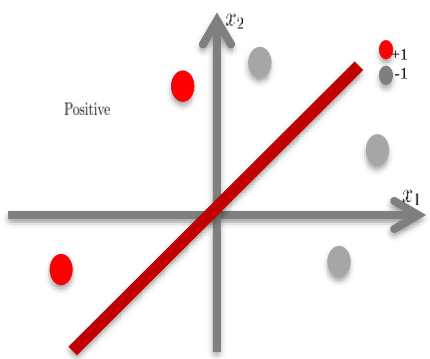
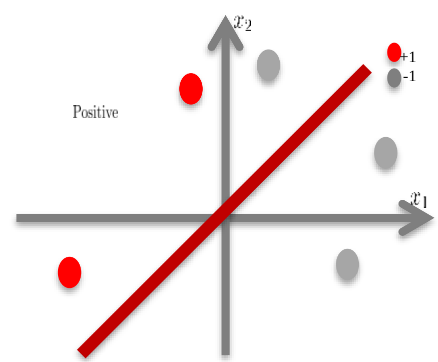
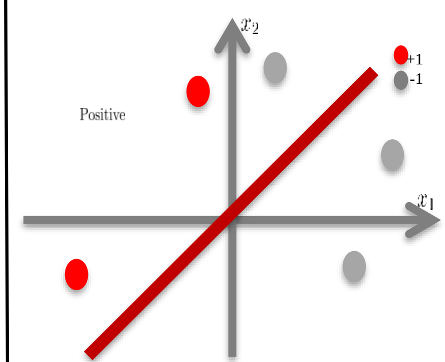
(1) random forest (2) neural nets (3) boosted tree (4) SVMs

Decision Tree Classifier

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

http://www.r2d3.us/visual-intro-to-machine-learning-part-2/

A Summary

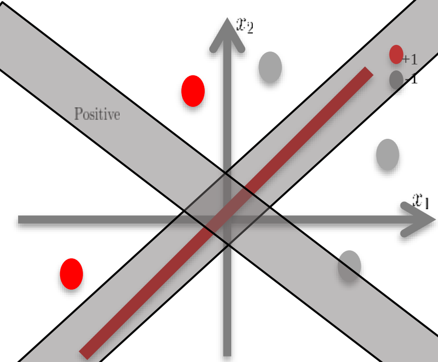
	Perceptron	Logistic Regression	SVM
Training	Training: Minimize $\mathcal{L}(\mathbf{w},b)=\sum_i max(0,-y_i(\mathbf{w}^T\mathbf{x}_i+b))$  convex optimization	Training: Minimize $\mathcal{L}(\mathbf{w},b)=\sum_{i=1}^{n}\ln(1+e^{-y_i(\mathbf{w}^T\mathbf{x}_i+b)})$  convex optimization	Training: Minimize $\mathcal{L}(\mathbf{w},b)=\frac{1}{2}\|\|\mathbf{w}\|\|^2 +C\sum_{i=1}^{n}(1-y_i(\mathbf{w}^T\mathbf{x}_i+b))_+$  convex optimization
Testing	 Test: $f(\mathbf{x};\mathbf{w},b)=\begin{cases}+1 & if\ \mathbf{w}^T\mathbf{x}+b\geq 0\\ -1 & otherwise\end{cases}$	 Test: $f(\mathbf{x};\mathbf{w},b)=\begin{cases}+1 & if\ \mathbf{w}^T\mathbf{x}+b\geq 0\\ -1 & otherwise\end{cases}$	 Test: $f(\mathbf{x};\mathbf{w},b)=\begin{cases}+1 & if\ \mathbf{w}^T\mathbf{x}+b\geq 0\\ -1 & otherwise\end{cases}$

The main difference with the previous classifiers.

Training

Training: Minimize $\mathcal{L}(\mathbf{w}, b) = \sum_i max(0, -y_i(\mathbf{w}^T\mathbf{x}_i + b))$

Testing

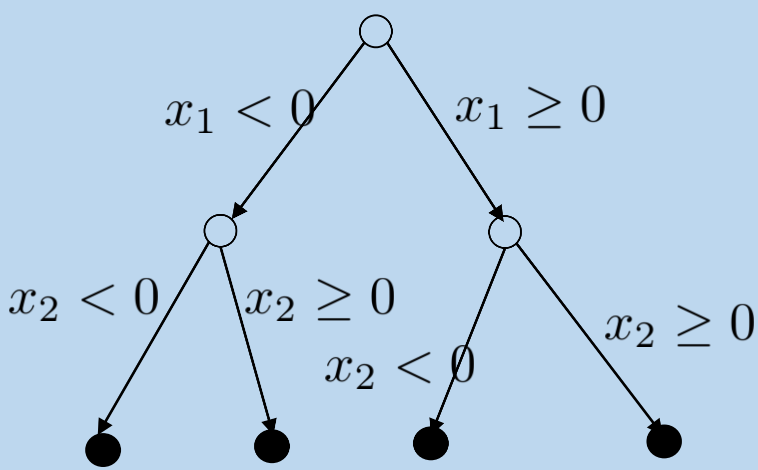


Test: $f(\mathbf{x}; \mathbf{w}, b) = \begin{cases} +1 & if\ \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & otherwise \end{cases}$

Training

Training: Minimize
an objective function that is recursively defined for splitting.
No explicit error/loss is minimized here!

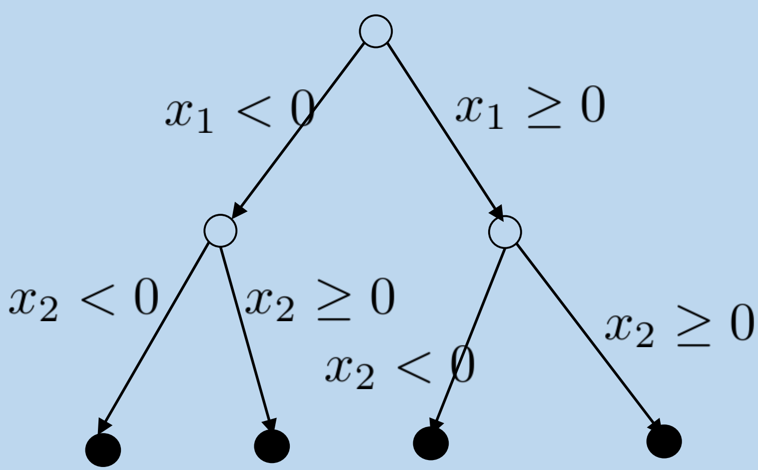
Decision Tree



Testing

The prediction is obtained by running a sequence decisions to go to the leaf node to obtain the classification.

Decision Tree

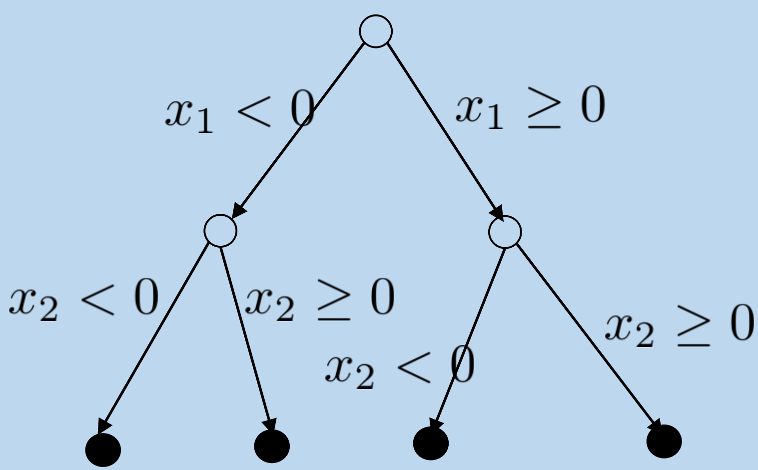


Why are we NOT able to define an explicit loss function to minimize like in Perceptron, Logisitic Regression, and SVM?

A. To complex to define.
B. It's a recursive function that has no intermediate loss.
C. The tree depths are not fixed.
D. This is a clustering task that is not suitable for classification.
E. None of the above.

Why are we NOT able to define an explicit loss function to minimize like in Perceptron, Logisitic Regression, and SVM?

Decision Tree



A. To complex to define.

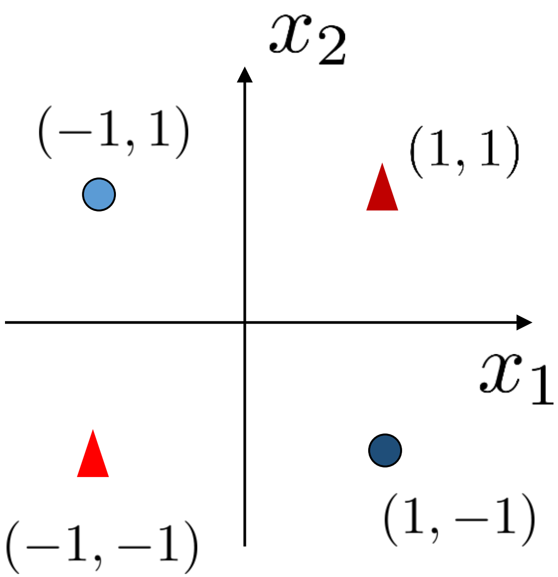
B. It's a recursive function that has no intermediate loss.

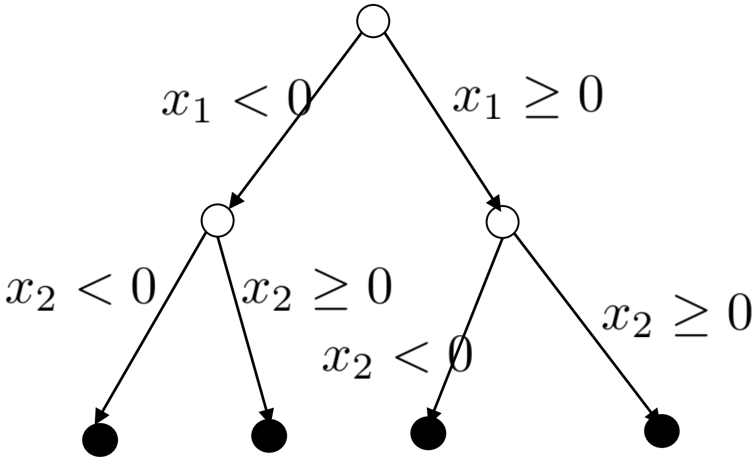
C. The tree depths are not fixed.

D. This is a clustering task that is not suitable for classification.

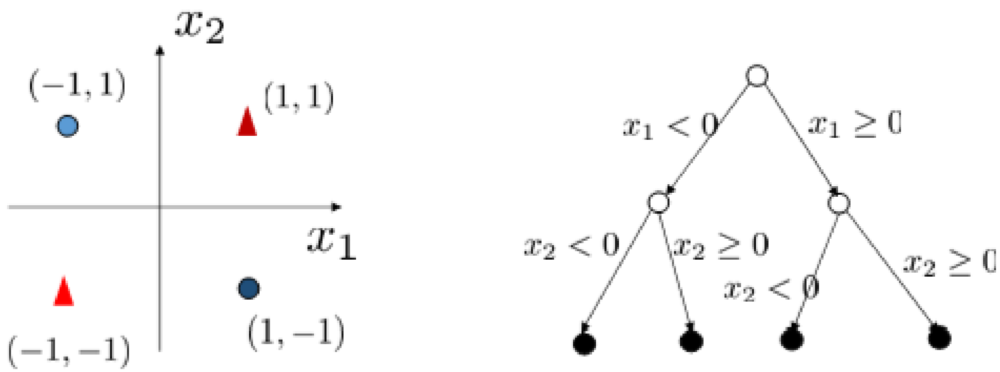
E. None of the above.

Decision Tree for XOR





Decision Tree for XOR



$(x_1, x_2) \to \boxed{(x_1, x_2, x_1 x_2)}$

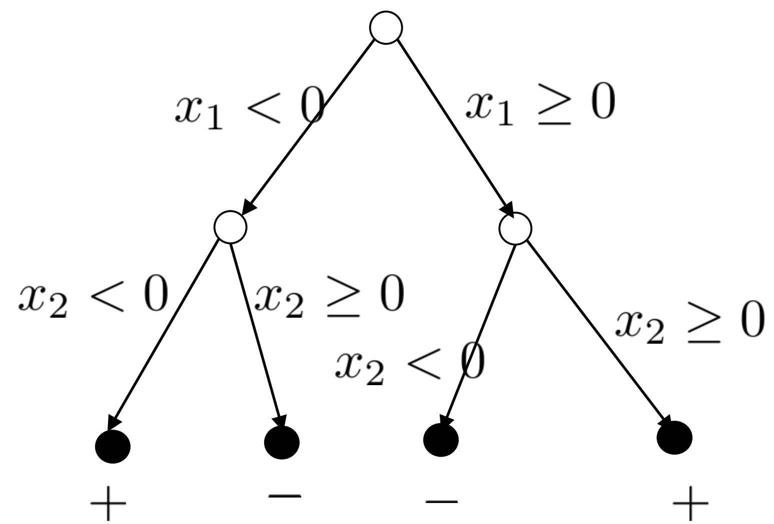
Decision Tree

The general rule is: **divide-and-conquer**

Decision node:○ decision to which path to pass the data.

Leaf (end) node:● which class (or class probability)

$$p(y|x)$$



C4.5 (J. Quinlan)

1.1 EXAMPLE: LABOR NEGOTIATION SETTLEMENTS

```
good, bad.

duration:                        continuous.
wage increase first year:        continuous.
wage increase second year:       continuous.
wage increase third year:        continuous.
cost of living adjustment:       none, tcf, tc.
working hours:                   continuous.
pension:                         none, ret_allw, empl_contr.
standby pay:                     continuous.
shift differential:              continuous.
education allowance:             yes, no.
statutory holidays:              continuous.
vacation:                        below average, average, generous.
longterm disability assistance:  yes, no.
contribution to dental plan:     none, half, full.
bereavement assistance:          yes, no.
contribution to health plan:     none, half, full.
```

Figure 1–1. File defining labor-neg classes and attributes

if wage increase first year $\leq$ 2.5 *then*
    if working hours $\leq$ 36 *then class good*
    else if working hours $>$ 36 *then*
        if contribution to health plan is none then class bad
        else if contribution to health plan is half then class good
        else if contribution to health plan is full then class bad
else if wage increase first year $>$ 2.5 *then*
    if statutory holidays $>$ 10 *then class good*
    else if statutory holidays $\leq$ 10 *then*
        if wage increase first year $\leq$ 4 *then class bad*
        else if wage increase first year $>$ 4 *then class good*

Decision Tree Visualization

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/

http://www.r2d3.us/visual-intro-to-machine-learning-part-2/

C4.5 (J. Quinlan)

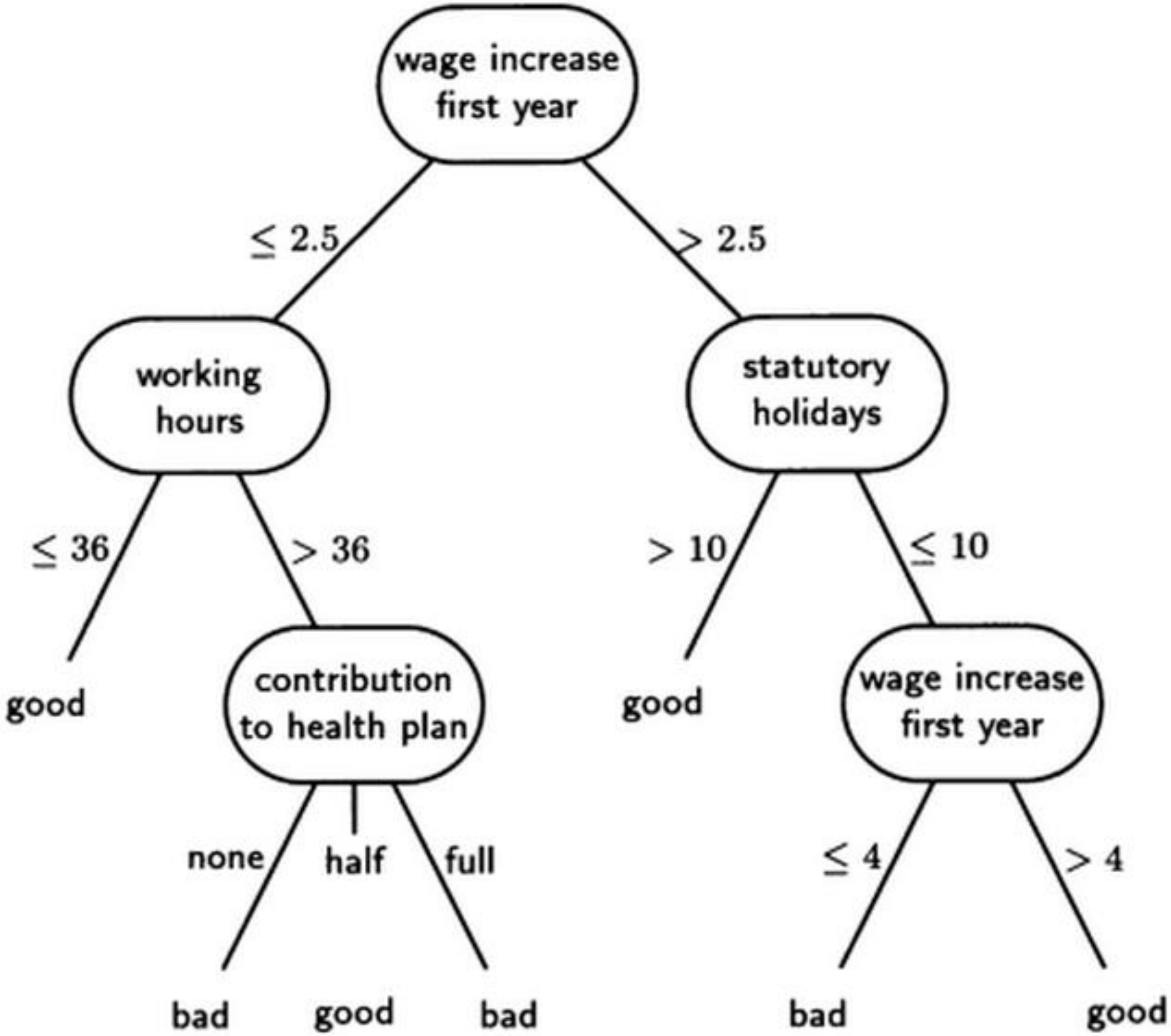


Figure 1–3. labor-neg decision tree in graph form

Is the decision tree classifier a parametric model?

A. Yes

B. In general, no.

C. It depends.

The leaf node of the decision tree classifier typicaly stores the class-labels of the training samples. The depth and the number of the leaf nodes increase when having more training data.

Is the decision tree classifier a parametric model?

A. Yes

B. In general, no.

C. It depends.

The leaf node of the decision tree classifier typicaly stores the class-labels of the training samples. The depth and the number of the leaf nodes increase when having more training data.

C4.5 (J. Quinlan)

1.1 EXAMPLE: LABOR NEGOTIATION SETTLEMENTS

```
C4.5 [release 5] decision tree generator        Fri Dec 6 13:33:54 1991

     Options:
          File stem <labor-neg>
          Trees evaluated on unseen cases

Read 40 cases (16 attributes) from labor-neg.data

Decision Tree:

wage increase first year ≤ 2.5 :
|    working hours ≤ 36 : good (2.0/1.0)
|    working hours > 36 :
|    |    contribution to health plan = none: bad (5.1)
|    |    contribution to health plan = half: good (0.4/0.0)
|    |    contribution to health plan = full: bad (3.8)
wage increase first year > 2.5 :
|    statutory holidays > 10 : good (21.2)
|    statutory holidays ≤ 10 :
|    |    wage increase first year ≤ 4 : bad (4.5/0.5)
|    |    wage increase first year > 4 : good (3.0)

Simplified Decision Tree:

wage increase first year ≤ 2.5 : bad (11.3/2.8)
wage increase first year > 2.5 :
|    statutory holidays > 10 : good (21.2/1.3)
|    statutory holidays ≤ 10 :
|    |    wage increase first year ≤ 4 : bad (4.5/1.7)
|    |    wage increase first year > 4 : good (3.0/1.1)

Tree saved

Evaluation on training data (40 items):

          Before Pruning          After Pruning
         ----------------   ---------------------------
          Size   Errors      Size   Errors     Estimate

            12  1 ( 2.5%)       7  1 ( 2.5%)   (17.4%)   <<

Evaluation on test data (17 items):

          Before Pruning          After Pruning
         ----------------   ---------------------------
          Size   Errors      Size   Errors     Estimate

            12  3 (17.6%)       7  3 (17.6%)   (17.4%)   <<

            (a)   (b)      <-classified as
           ----  ----
             10     1      (a): class good
              2     4      (b): class bad
```

Training C4.5 algorithm (J. Quinlan)

1. Tree construction (divide-and-conquer).

2. Tree pruning (in a way cross-validation to reduce generalization error).

Training C4.5 algorithm (J. Quinlan)

Hunt's method for constructing a decision tree from a set S of training samples. $\{C_1, C_2, ..., C_k\}$

There are three possibilities:

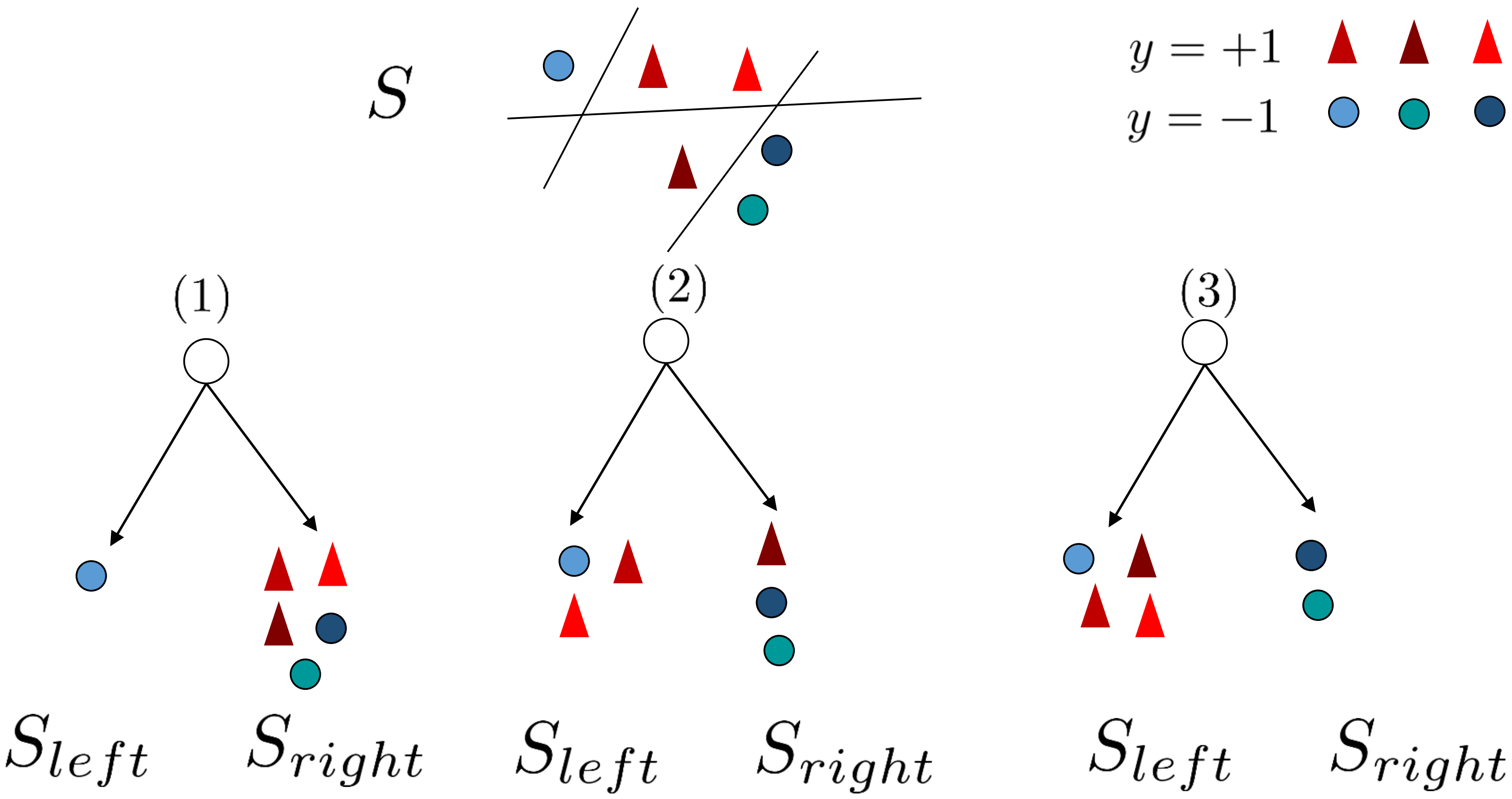
(1) S contains one or more samples that all belong to a single class. C_j

(2) S contains no samples.

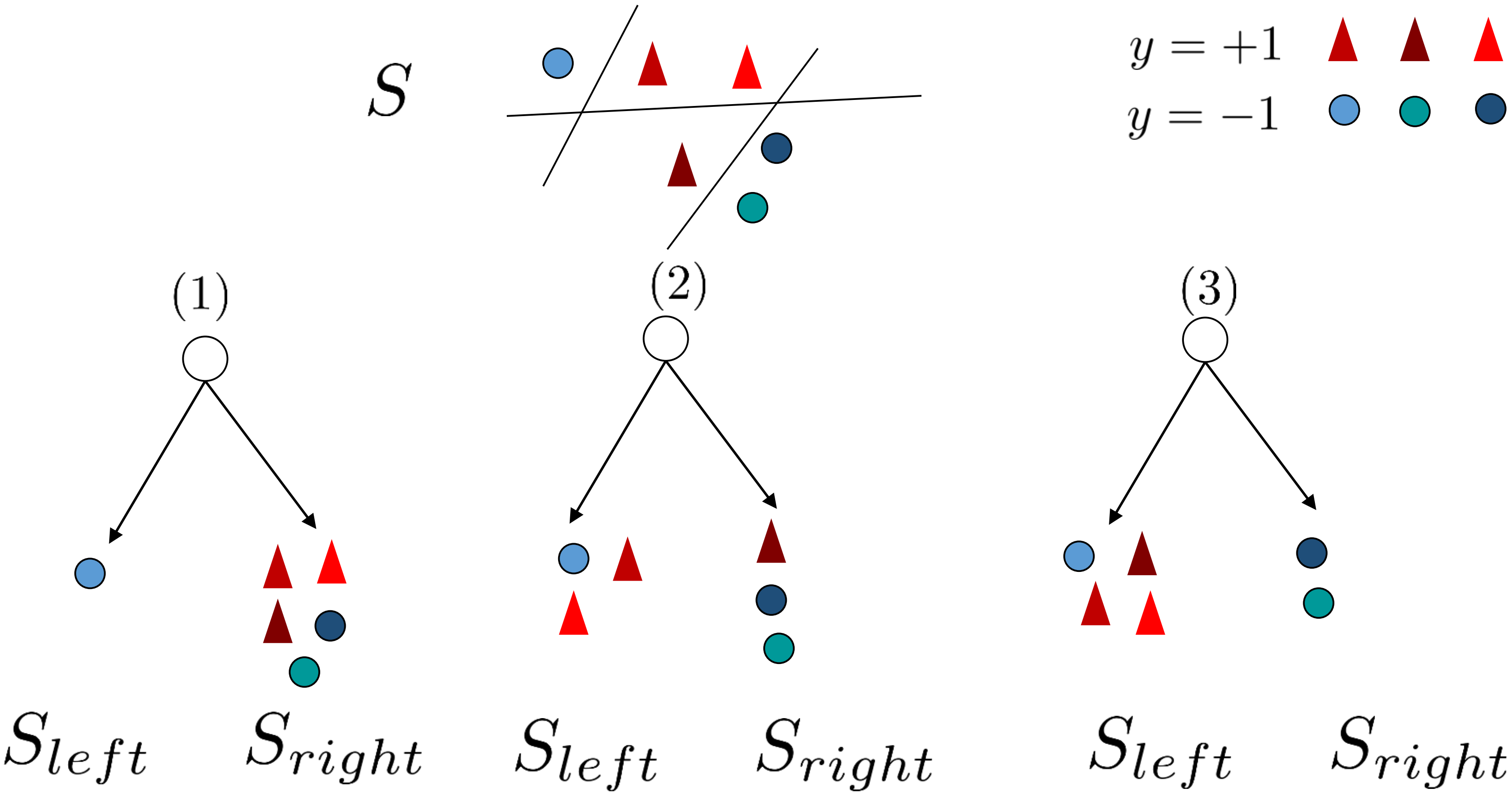
(3) S contains samples that belong to a mixture of classes.

Tree construction (J. Quinlan)

We recursively construct a tree each time to find the feature at a particular value to maximize the gain (minimize the cost).



Tree construction (J. Quinlan)



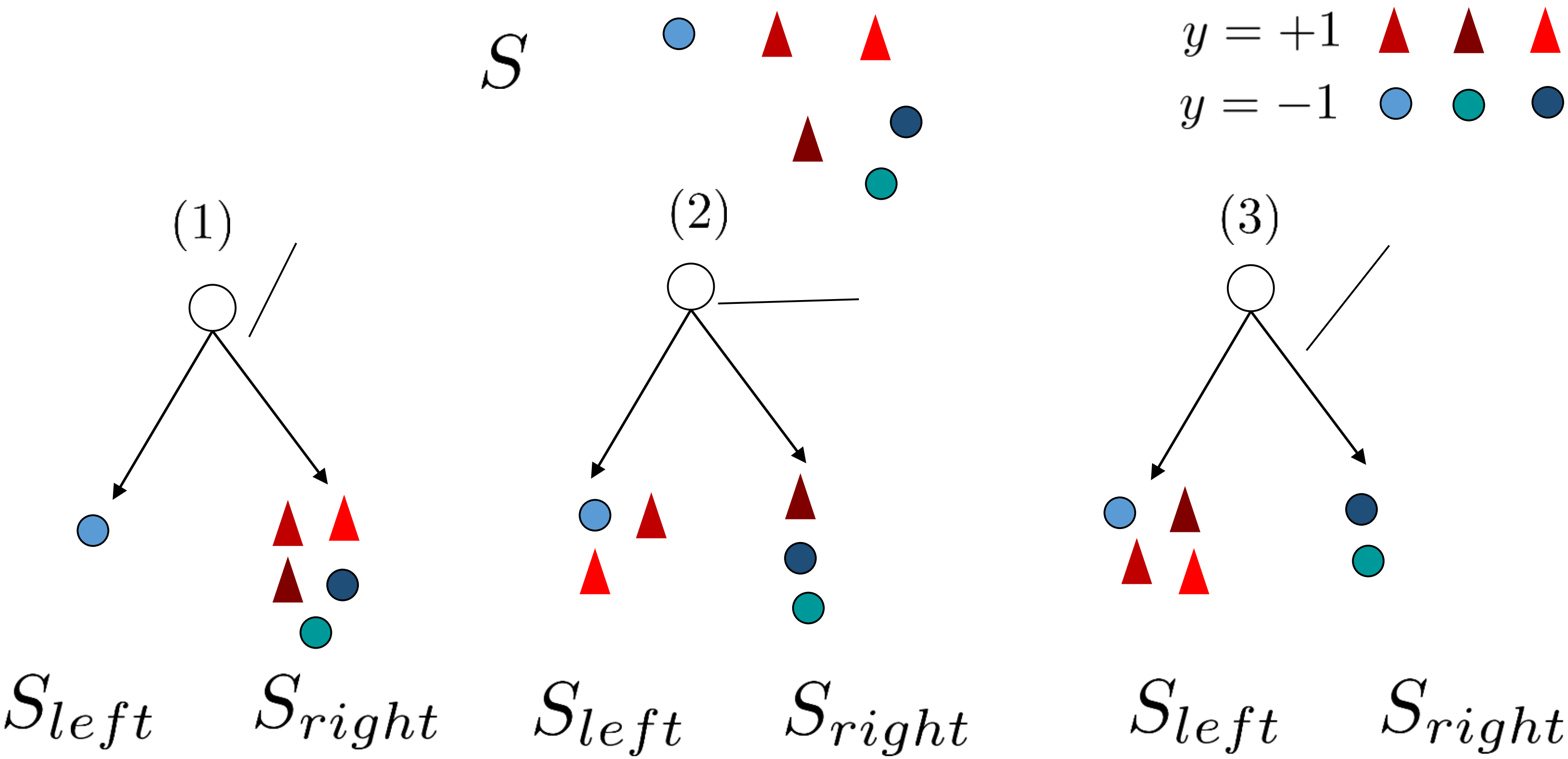
Which one to use:

A: (1)

B: (2)

C: (3)

Tree construction (J. Quinlan)

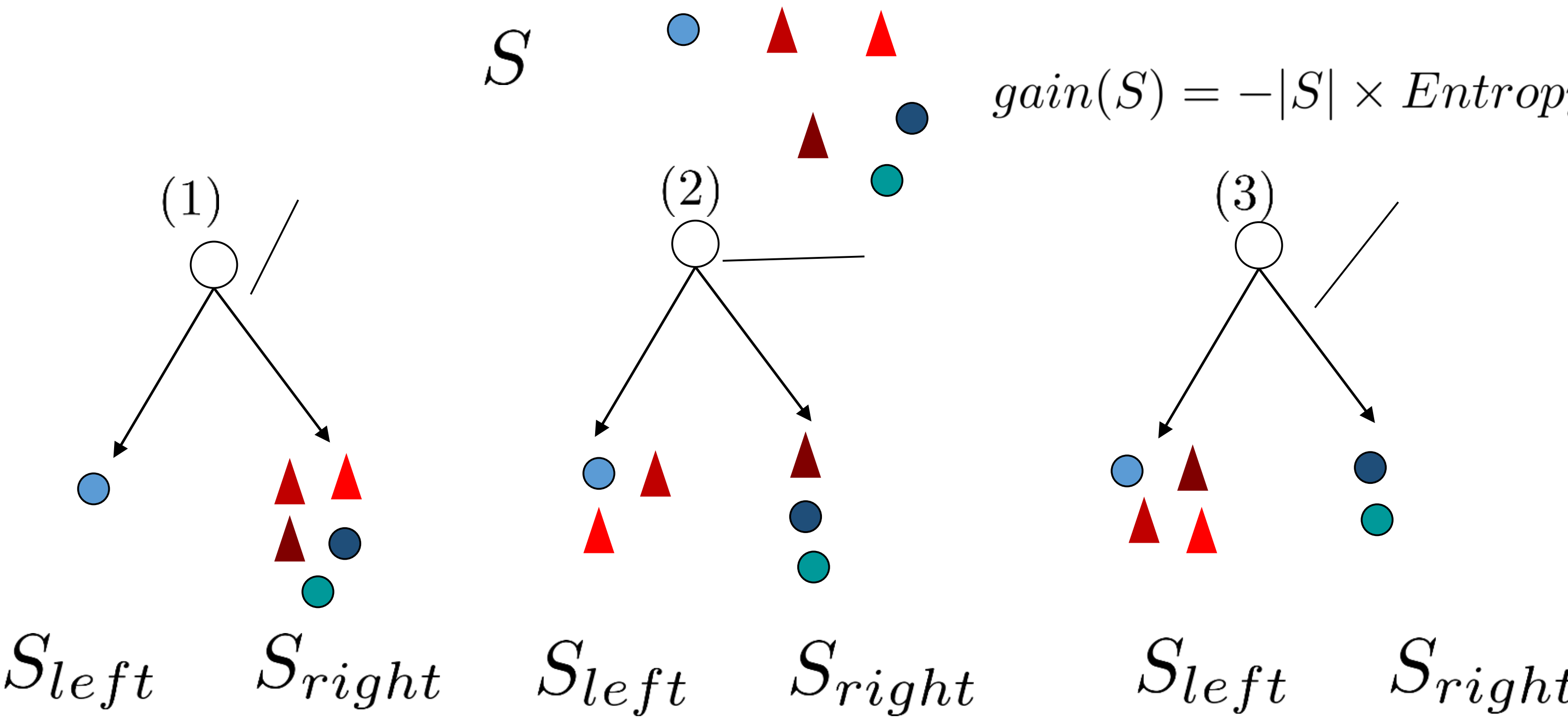


$$f^* = \arg max_f \quad gain(S^{(f)}_{left}) + gain(S^{(f)}_{right}) - gain(S)$$

$$gain(S) = -|S| \times Entropy(Y_S)$$

Tree construction (J. Quinlan)

$$gain(S) = -|S| \times Entropy(Y_S)$$



(1)
$$gain(S_{left}) = 1 \times (1 \times \log(1) + 0 \times \log(0) = 0$$
$$gain(S_{right}) = 5 \times (0.4 \times \log(0.4) + 0.6 \times \log(0.6) = -3.365$$
$$0 - 3.365 = -3.365$$

(2)
$$gain(S_{left}) = 3 \times (0.33 \times \log(0.33) + 0.67 \times \log(0.67) = -1.9095$$
$$gain(S_{right}) = 3 \times (0.67 \times \log(0.67) + 0.33 \times \log(0.33) = -1.9095$$
$$-1.9095 - 1.9095 = -3.819$$

(3)
$$gain(S_{left}) = 4 \times (0.25 \times \log(0.25) + 0.75 \times \log(0.75) = -2.2493$$
$$gain(S_{right}) = 2 \times (0 \times \log(0) + 1 \times \log(1) = 0$$
$$-2.2493 + 0 = -2.2493$$