Machine learning in computer vision

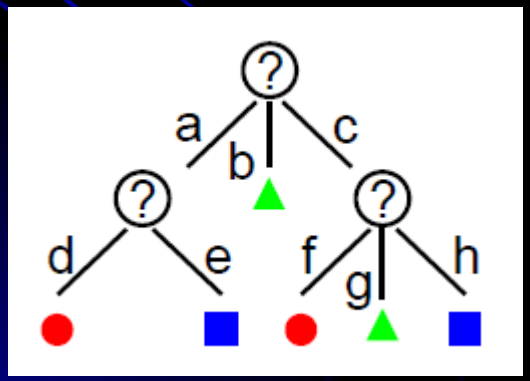
Lesson 6

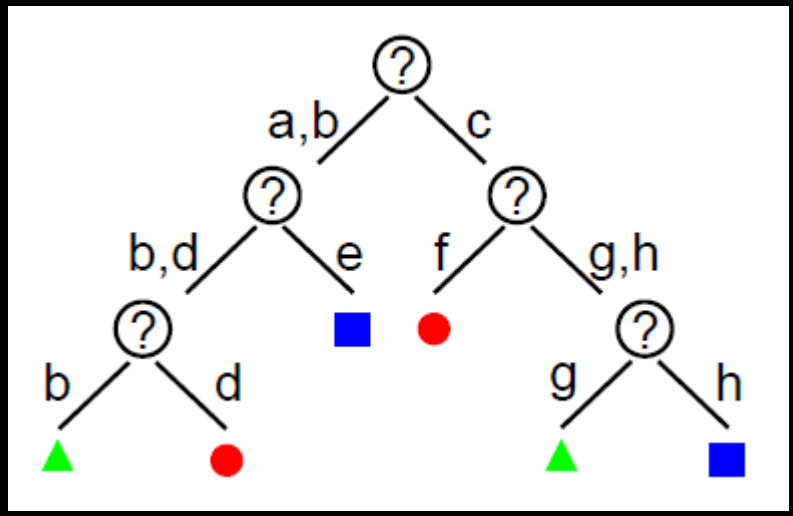
Decision trees

Nominal data – no interpretation of distance

Rules

Tree:
node = test, branches = possible outcomes
leaf = object class

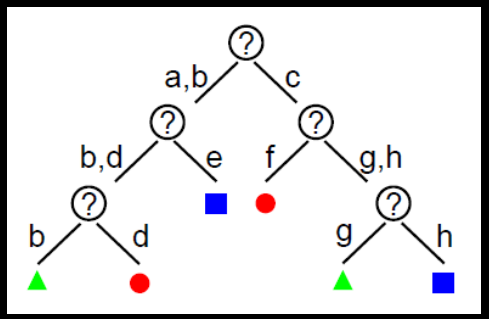




DT classification

to classify an example:

1. start at the root
2. perform the test
3. follow the edge corresponding to the outcome
4. goto 2. unless leaf
5. predict the class associated with the leaf



DT learning

Set of tests

Splitting criterion

Stop-splitting rule

Classification rules

DT learning

Set of tests

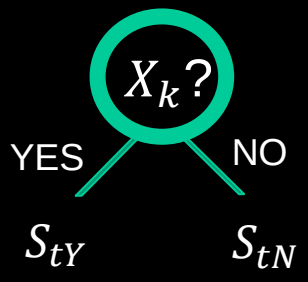
Set of all possible tests e.g. $X_k \leq \theta$

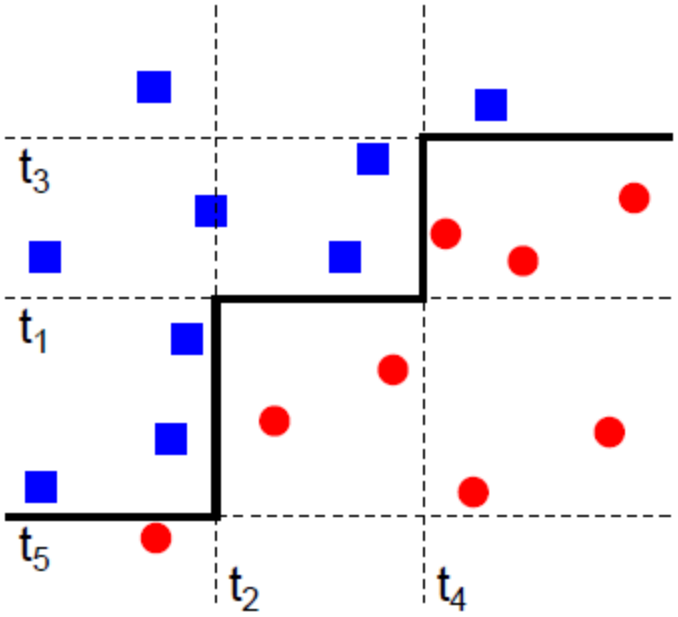
In each node

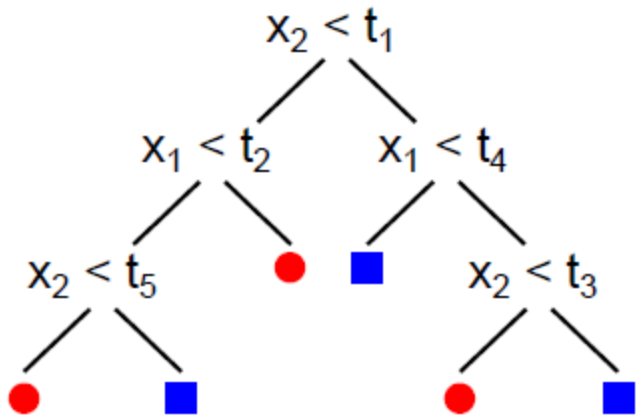
Binary node t analyzes a subset $S_t \subset X$ and splits it into S_{tY}, S_{tN}:

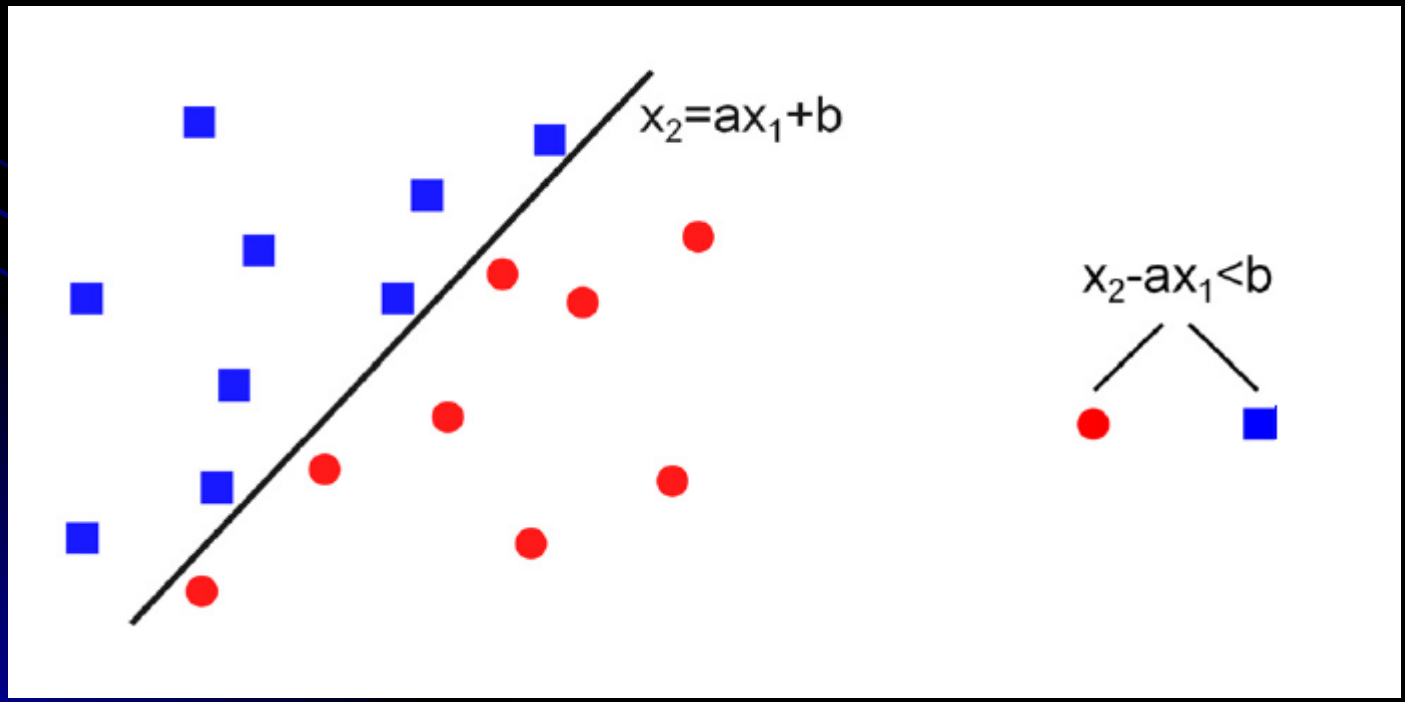
$S_{tY} \cap S_{tN} = \emptyset$

$S_{tY} \cup S_{tN} = S_t$




t_3
t_1
t_5
t_2
t_4


$x_2 < t_1$
$x_1 < t_2$
$x_1 < t_4$
$x_2 < t_5$
$x_2 < t_3$


$x_2=ax_1+b$
$x_2-ax_1<b$

DT learning

Splitting criterion

Specifies the feature X_k and threshold θ to get the best split of S_t

In child nodes we want to decrease impurity

In pure node all data belong to one class

Measure of impurity?

Variable Quality Measures

Let S be a sample of training instances and p_c be the proportions of instances of class ω_c (c=1,…,C) in S.

Define an **impurity** measure $I(S)$ that satisfies:

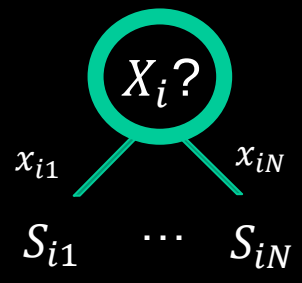
- $I(S)$ is minimal only when p_i=1 and p_c=0 for $c \neq i$
  (all objects are of the same class);
- $I(S)$ is maximal only when p_c =1/C
  (there is exactly the same number of objects of all classes);
- $I(S)$ is symmetric with respect to $p_1,\ldots,p_C$.

Reduction of Impurity: Discrete Variables

The "best" variable is the variable X_i that determines a split maximizing the expected reduction of impurity:

$$\Delta I(S, X_i) = I(S) - \sum_j \frac{|S_{ij}|}{|S|} I(S_{ij})$$

where S_{ij} is the subset of instances from S s.t. $X_i = x_{ij}$.



Impurity measure

Gini-index – how often will a random object be misclassified if classified randomly according to the class distribution

$$p_i = \frac{|\{\mathbf{X}|\mathbf{X}\in\omega_i\}|}{|X_t|}, \qquad GI = \sum_{i=1}^{m} p_i(1-p_i) = 1 - \sum_{i=1}^{m} p_i^2$$

Probability of choosing

Probability of misclassification

$$GI(S,F) = GI(S) - \sum_{f\in \mathrm{val}(F)} P(F=f)GI(S_f)$$

Mutual information (information gain)

$$IG(S,F) = H(S) - \sum_{f\in \mathrm{val}(F)} P(F=f)H(S_f)$$

…

DT learning

Stop-splitting rule

When to stop splitting a node and declare it as a leaf?

Impurity measure of the best feature is less than a threshold *T*

The cardinality of S_t is small enough

The node is pure

DT learning

Rules

Once a node is declared to be a leaf, then it has to be given a class label

majority rule:

the leaf is labeled as ω_j, where

$$j = \arg\max_c p(\omega_c | S_t)$$

DT learning

S - Training Set
A - Input Feature Set
y - Target Feature

```
TreeGrowing (S,A,y)
    Create a new tree T with a single root node t
    IF One of the Stopping Criteria is fulfilled THEN
        Mark the root node in T as a leaf with the most common value of y in S as a label
    ELSE
        Find F – a best split of S => outcomes (o1,...,on)
        IF best splitting metric > threshold THEN
            label t with F
            FOR each outcome oi:
                Subtree_i= TreeGrowing (S_oi,A,y)
                Connect the root node of t to Subtree_i with an edge that is labelled as oi
            END FOR
        ELSE
            Mark the root node t as a leaf with the most common value of y in S as a label
        END IF
    END IF
    RETURN T
```

DT example

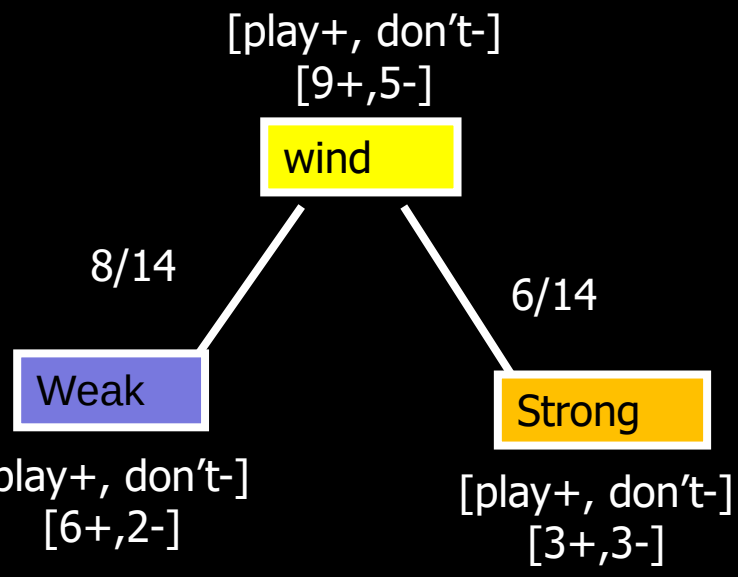
day	outlook	temp	humidity	wind	play
D01	Sunny	Hot	High	Weak	***No***
D02	Sunny	Hot	High	Strong	***No***
D03	Overcast	Hot	High	Weak	***Yes***
D04	Rain	Mild	High	Weak	***Yes***
D05	Rain	Cool	Normal	Weak	***Yes***
D06	Rain	Cool	Normal	Strong	***No***
D07	Overcast	Cool	Normal	Strong	***Yes***
D08	Sunny	Mild	High	Weak	***No***
D09	Sunny	Cool	Normal	Weak	***Yes***
D10	Rain	Mild	Normal	Weak	***Yes***
D11	Sunny	Mild	Normal	Strong	***Yes***
D12	Overcast	Mild	High	Strong	***Yes***
D13	Overcast	Hot	Normal	Weak	***Yes***
D14	Rain	Mild	High	Strong	***No***

DT example with information gain

$$IG(S,F) = H(S) - \sum_{f \in \text{val}(F)} P(F=f) H(S_f)$$

Compute IG(S,F) for all features in the root
Take the best feature
Compute IG(S_{split},F) for all possible features in 1st level
Take the best feature
…

day	outlook	temp	humidity	wind	play
D02	Sunny	Hot	High	Strong	*No*
D06	Rain	Cool	Normal	Strong	*No*
D07	Overcast	Cool	Normal	Strong	*Yes*
D11	Sunny	Mild	Normal	Strong	*Yes*
D12	Overcast	Mild	High	Strong	*Yes*
D14	Rain	Mild	High	Strong	*No*
D01	Sunny	Hot	High	Weak	*No*
D03	Overcast	Hot	High	Weak	*Yes*
D04	Rain	Mild	High	Weak	*Yes*
D05	Rain	Cool	Normal	Weak	*Yes*
D08	Sunny	Mild	High	Weak	*No*
D09	Sunny	Cool	Normal	Weak	*Yes*
D10	Rain	Mild	Normal	Weak	*Yes*
D13	Overcast	Hot	Normal	Weak	*Yes*

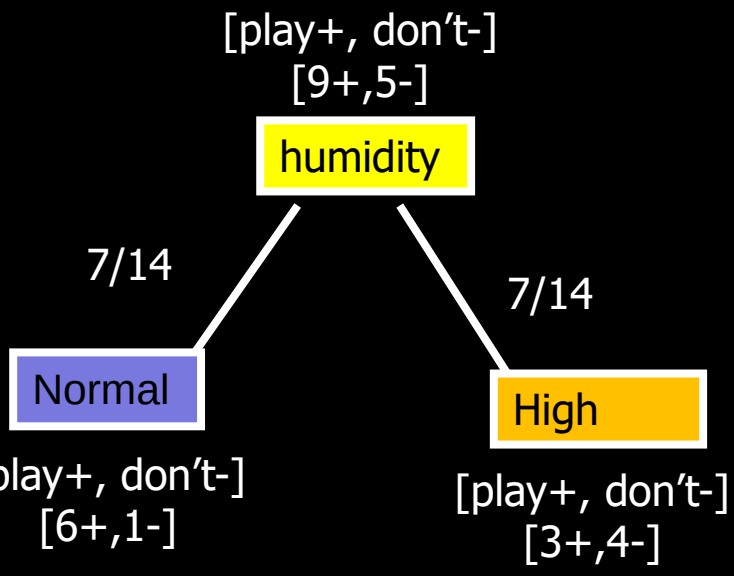


$$IG(S, wind\ ) = H(S) - P(weak)H(S_{weak}) - P(strong)H(S_{strong})$$

H(S) = -(9/14)*log(9/14) – (5/14)*log(5/14) = 0.940
$H(S_{weak})$ = -(6/8)*log(6/8) – (2/8)*log(2/8) = 0.811
$H(S_{strong})$ = -(3/6)*log(3/6) – (3/6)*log(3/6) = 1.0

IG(S;wind) = 0.940 - (8/14)*0.811 - (6/14)*1.0 = 0.048

day	outlook	temp	humidity	wind	play
D01	Sunny	Hot	High	Weak	*No*
D02	Sunny	Hot	High	Strong	*No*
D03	Overcast	Hot	High	Weak	*Yes*
D04	Rain	Mild	High	Weak	*Yes*
D08	Sunny	Mild	High	Weak	*No*
D12	Overcast	Mild	High	Strong	*Yes*
D14	Rain	Mild	High	Strong	*No*
D05	Rain	Cool	Normal	Weak	*Yes*
D06	Rain	Cool	Normal	Strong	*No*
D07	Overcast	Cool	Normal	Strong	*Yes*
D09	Sunny	Cool	Normal	Weak	*Yes*
D10	Rain	Mild	Normal	Weak	*Yes*
D11	Sunny	Mild	Normal	Strong	*Yes*
D13	Overcast	Hot	Normal	Weak	*Yes*

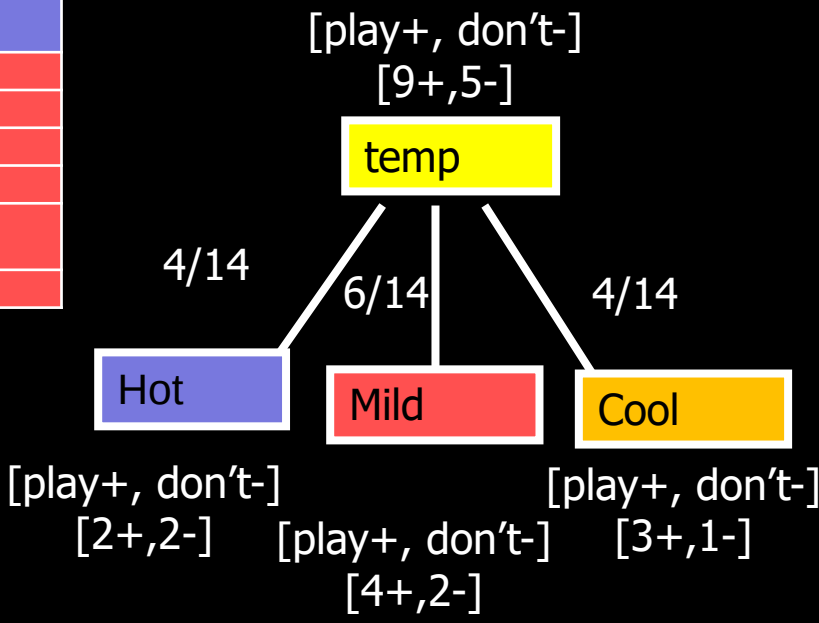


$$IG(S, humidity) = H(S) - P(normal)H(S_{normal}) - P(high)H(S_{high})$$

$H(S_{normal}) = -(6/7)*\log(6/7) - (1/7)*\log(1/7) = 0.592$
$H(S_{high}) = -(3/7)*\log(3/7) - (4/7)*\log(4/7) = 0.985$

$IG(S;humidity) = 0.940 - (7/14)*0.592 - (7/14)*0.985 = 0.152$

day	outlook	temp	humidity	wind	play
D05	Rain	Cool	Normal	Weak	*Yes*
D06	Rain	Cool	Normal	Strong	*No*
D07	Overcast	Cool	Normal	Strong	*Yes*
D09	Sunny	Cool	Normal	Weak	*Yes*
D01	Sunny	Hot	High	Weak	*No*
D02	Sunny	Hot	High	Strong	*No*
D03	Overcast	Hot	High	Weak	*Yes*
D13	Overcast	Hot	Normal	Weak	*Yes*
D04	Rain	Mild	High	Weak	*Yes*
D08	Sunny	Mild	High	Weak	*No*
D10	Rain	Mild	Normal	Weak	*Yes*
D11	Sunny	Mild	Normal	Strong	*Yes*
D12	Overcast	Mild	High	Strong	*Yes*
D14	Rain	Mild	High	Strong	*No*

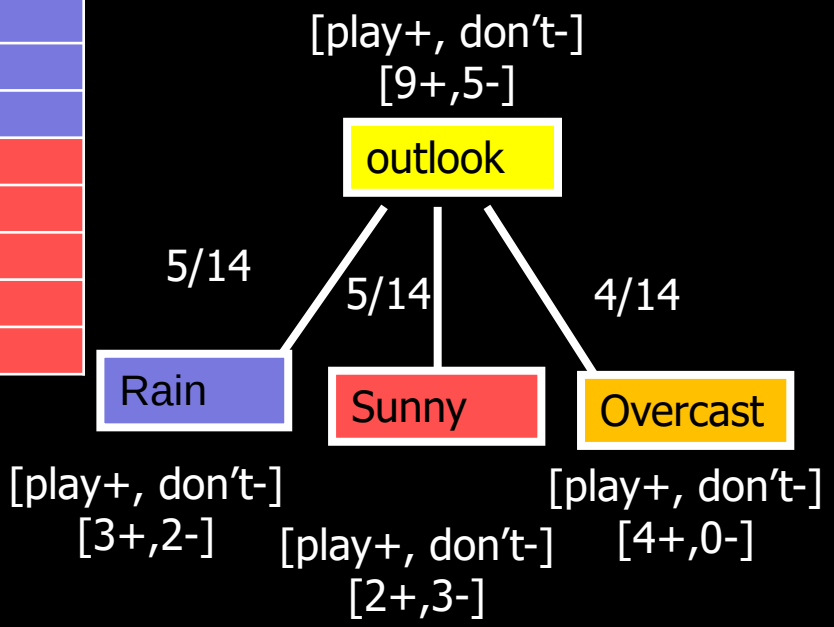


$$IG(S, temp) = H(S) - P(hot)H(S_{hot}) - P(mild)H(S_{mild}) - P(cool)H(S_{cool})$$

$H(S_{hot})$ = -(2/4)*log(2/4) – (2/4)*log(2/4) = 1.0
$H(S_{mild})$ = -(4/6)*log(4/6) – (2/6)*log(2/6) = 0.918
$H(S_{cool})$ = -(3/4)*log(3/4) – (1/4)*log(1/4) = 0.811

IG(S;temp) = 0.940 - (4/14)*1 - (6/14)*0.918 - (4/14)*0.811 = 0.029

day	outlook	temp	humidity	wind	play
D03	Overcast	Hot	High	Weak	*Yes*
D07	Overcast	Cool	Normal	Strong	*Yes*
D12	Overcast	Mild	High	Strong	*Yes*
D13	Overcast	Hot	Normal	Weak	*Yes*
D04	Rain	Mild	High	Weak	*Yes*
D05	Rain	Cool	Normal	Weak	*Yes*
D06	Rain	Cool	Normal	Strong	*No*
D10	Rain	Mild	Normal	Weak	*Yes*
D14	Rain	Mild	High	Strong	*No*
D01	Sunny	Hot	High	Weak	*No*
D02	Sunny	Hot	High	Strong	*No*
D08	Sunny	Mild	High	Weak	*No*
D09	Sunny	Cool	Normal	Weak	*Yes*
D11	Sunny	Mild	Normal	Strong	*Yes*



$$IG(S, outlook) = H(S) - P(rain)H(S_{rain}) - P(sunny)H(S_{sunny}) - P(overcast)H(S_{overcast})$$

$H(S_{rain}) = -(3/5)*\log(3/5) - (2/5)*\log(2/5) = 0.971$
$H(S_{sunny}) = -(2/5)*\log(2/5) - (3/5)*\log(3/5) = 0.971$
$H(S_{overcast}) = -(4/4)*\log(4/4) - 0 = 0$

IG(S;outlook) = 0.940 - (5/14)*0.971 - (5/14)*0.971 - (4/14)*0 = 0.247

DT example with information gain

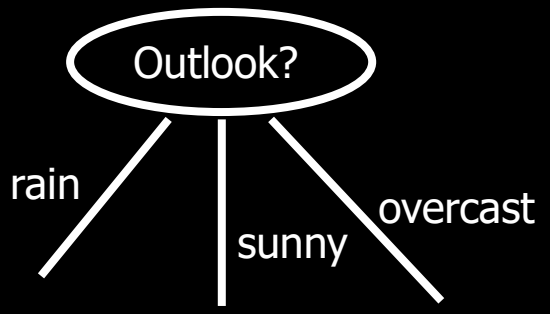
Compute IG(S,F) for all features in the root
Take the best feature:

IG(S;wind) = 0.048

IG(S;humidity) = 0.152

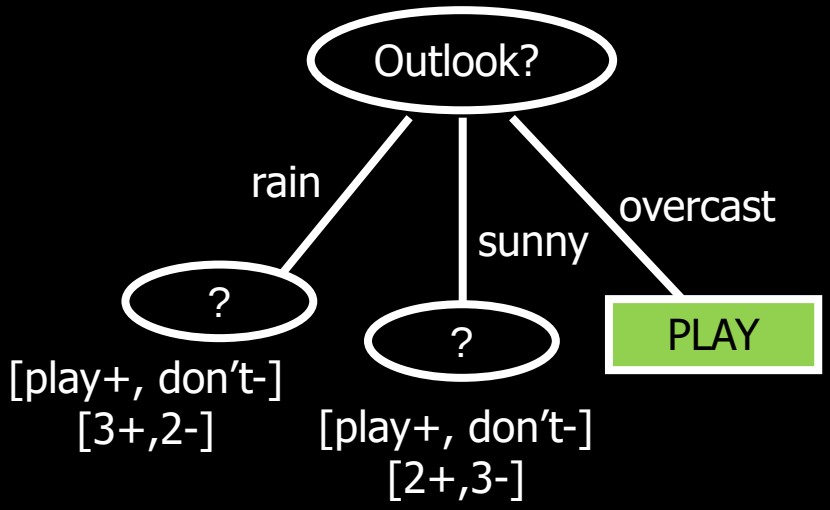
IG(S;temp) = 0.029

IG(S;outlook) = 0.247



DT example with information gain

We have solved the root. What next?

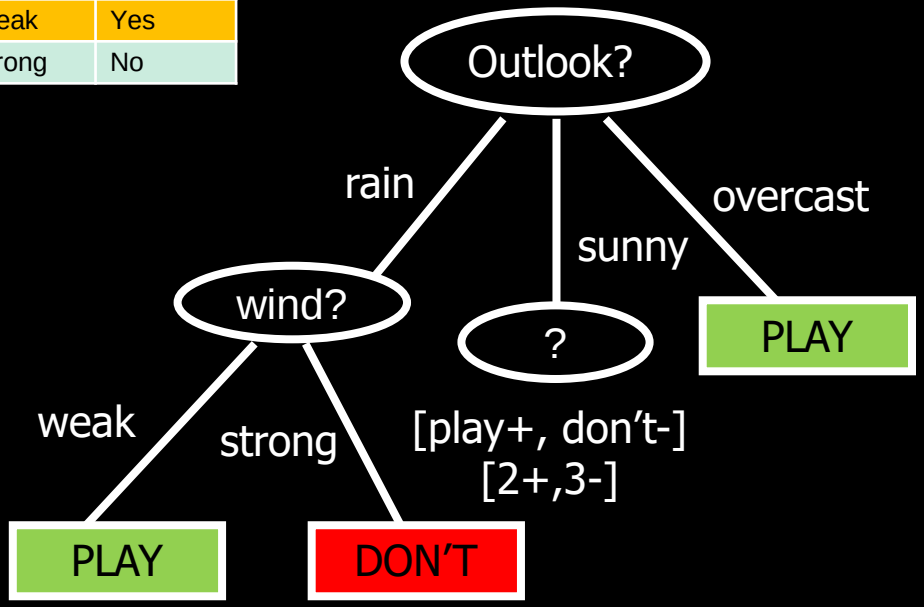


day	outlook	temp	humidity	wind	play
D04	Rain	Mild	High	Weak	Yes
D05	Rain	Cool	Normal	Weak	Yes
D06	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

day	outlook	temp	humidity	wind	play
D01	Sunny	Hot	High	Weak	No
D02	Sunny	Hot	High	Strong	No
D08	Sunny	Mild	High	Weak	No
D09	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

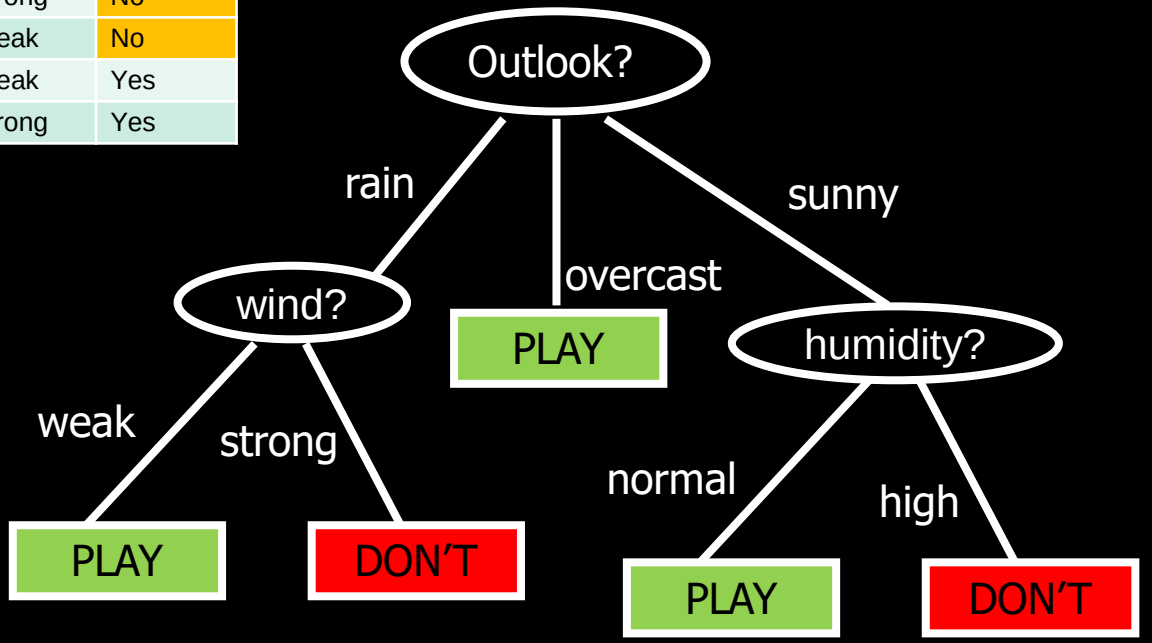
DT example with information gain

day	outlook	temp	humidity	wind	play
D04	Rain	Mild	High	Weak	Yes
D05	Rain	Cool	Normal	Weak	Yes
D06	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



DT example with information gain

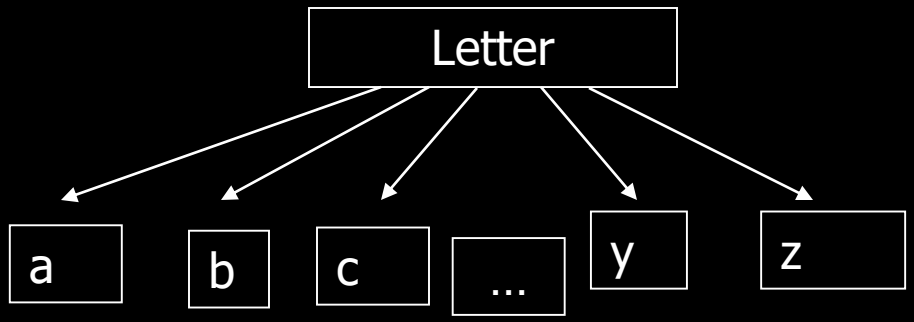
day	outlook	temp	humidity	wind	play
D01	Sunny	Hot	High	Weak	No
D02	Sunny	Hot	High	Strong	No
D08	Sunny	Mild	High	Weak	No
D09	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes



Information gain

favors attributes with many possible values

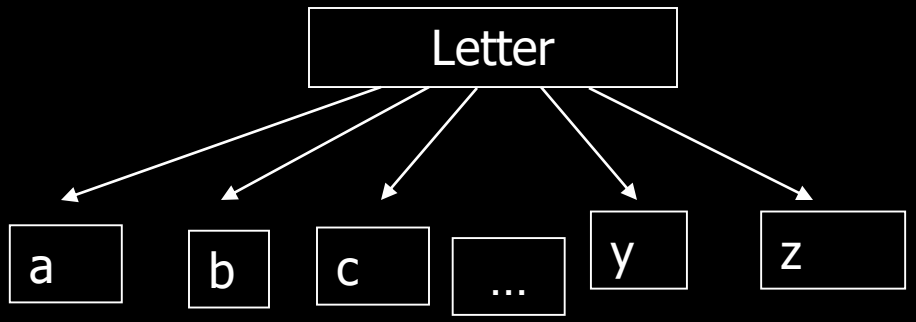
Variables with Many Values



Problem:

- Not good splits: they fragment the data too quickly, leaving insufficient data at the next level
- The reduction of impurity of such test is often high (example: split on the object id).

Variables with Many Values



Two solutions:

- Change the splitting criterion to penalize variables with many values
- Consider only binary splits

Variables with Many Values

$$SplitInfo(S,F) = -\sum_{f \in \text{val}(F)} \frac{|S_f|}{|S|} \log_2 \frac{|S_f|}{|S|}$$

$$GainRatio(S,F) = \frac{IG(S,F)}{SplitInfo(S,F)}$$

Example: outlook in the play_tennis

- IG (outlook) = 0.247
- SplitInfo (outlook) = 1.577
- GainRatio (outlook) = 0.247/1.577=0.157 < 0.247

Problem: the gain ratio favours unbalanced tests

DT example with information gain

Binary tree?

Check all possible binary splits

Temp:

- {{Cool}, {Hot, Mild}}
- {{Hot}, {Cool, Mild}}
- {{Mild}, {Hot, Cool}}

Outlook:

- {{Sunny},{Rain, Overcast}}
- {{Rain},{Sunny, Overcast}}
- {{Overcast},{Rain, Sunny}}

DT example with information gain

Numerical values?

Check all possible thresholds

$\{\{v_1, \ldots v_i\}, \{v_{i+1}, \ldots v_k\}\}$

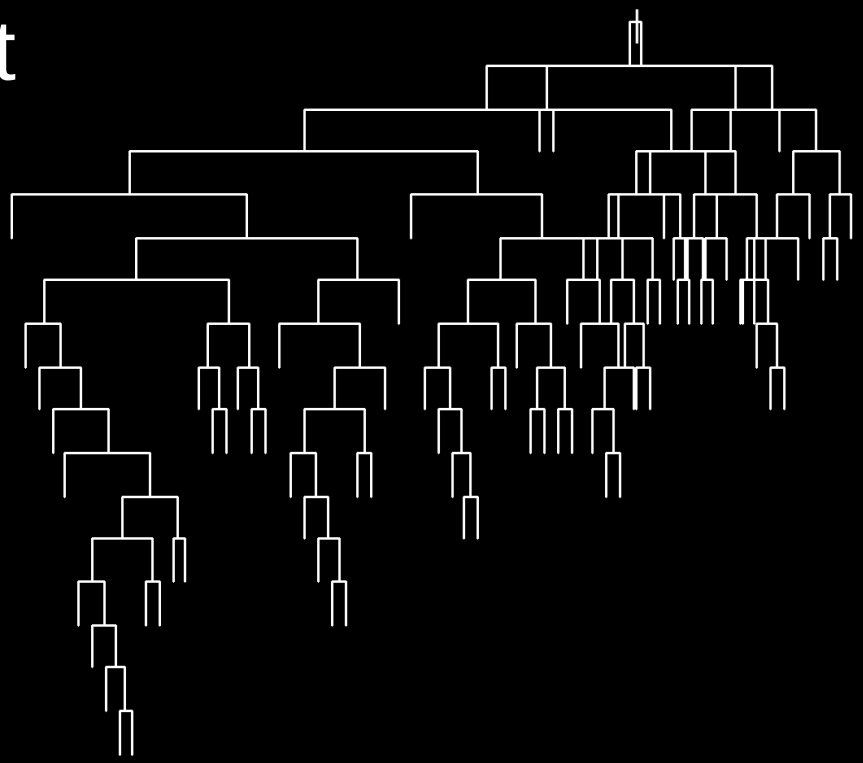
$\theta_i = \frac{v_i + v_{i+1}}{2}$

Discretization to form an ordinal categorical attribute

- Static – discretize once at the beginning
- Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering

Tree pruning

Trees tend to grow a lot



Solution: pruning

Pruning

Pre-pruning: stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data (Stop-splitting rule)

Post-pruning: Allow the tree to overfit the data, and then post-prune the tree.

Reduced-error pruning

Each node is a candidate for pruning

Pruning consists in removing a subtree rooted in a node: the node becomes a leaf and is assigned the most common classification

Nodes are removed only if the resulting tree performs no worse on the validation set.

Nodes are pruned iteratively: at each iteration the node whose removal most increases accuracy on the validation set is pruned.

Pruning stops when no pruning increases accuracy

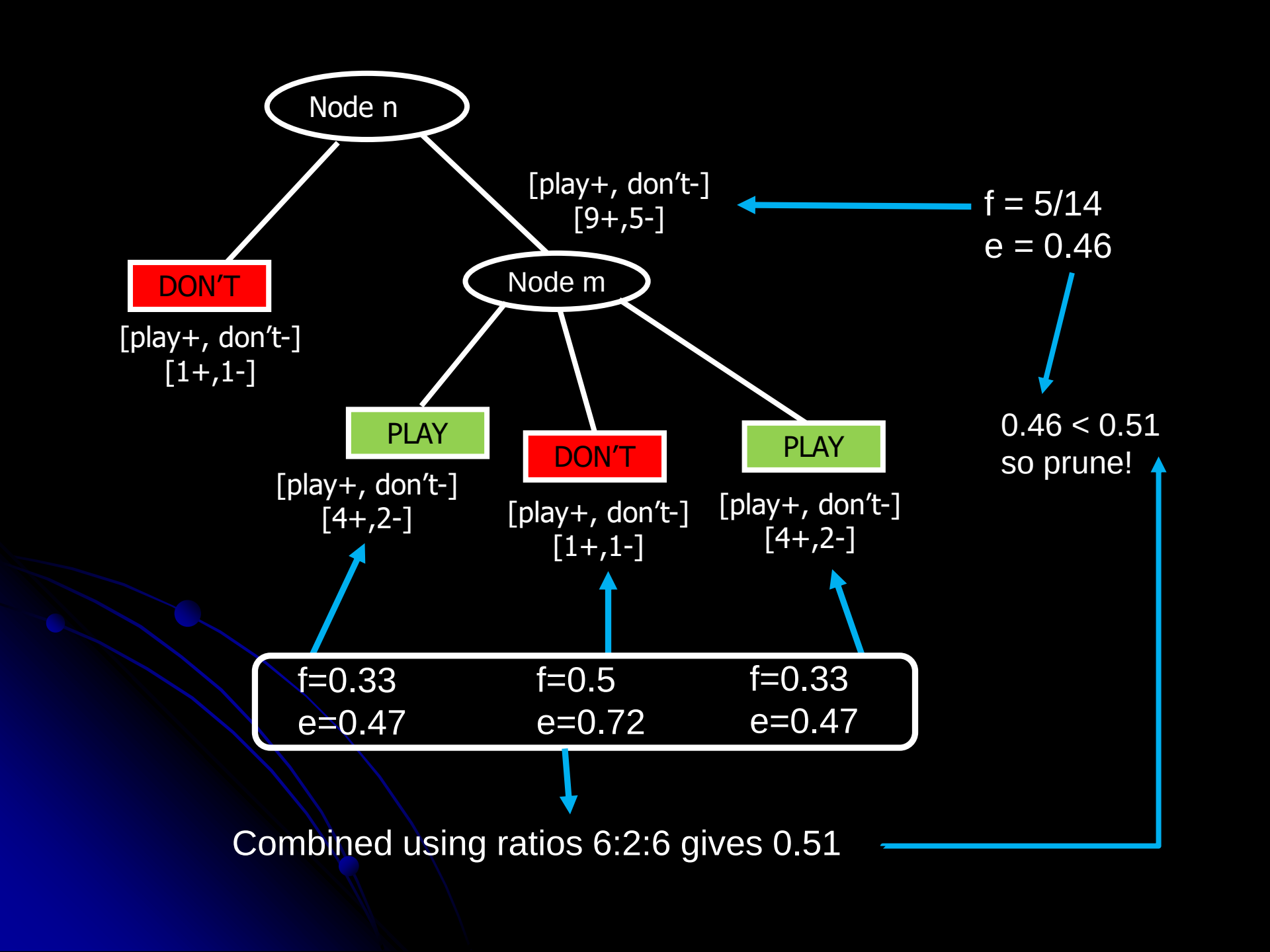
C4.5's Pruning Method

Given the error f on the training data, the upper bound for the error estimate for a node is computed as

$$e = \left(f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N}} \right) \Big/ \left(1 + \frac{z^2}{N} \right)$$

If CI = 50% then z = 0.69 (from normal distribution)

N is the number of instances covered by the leaf


Node n
[play+, don't-]
[9+,5-]
f = 5/14
e = 0.46
DON'T
[play+, don't-]
[1+,1-]
Node m
PLAY
[play+, don't-]
[4+,2-]
DON'T
[play+, don't-]
[1+,1-]
PLAY
[play+, don't-]
[4+,2-]
0.46 < 0.51
so prune!
f=0.33
e=0.47
f=0.5
e=0.72
f=0.33
e=0.47
Combined using ratios 6:2:6 gives 0.51

DT summary

Pros:

- ✓ simple to understand and interpret
- ✓ little data preparation and little computation
- ✓ indicates which attributes are most important for classification

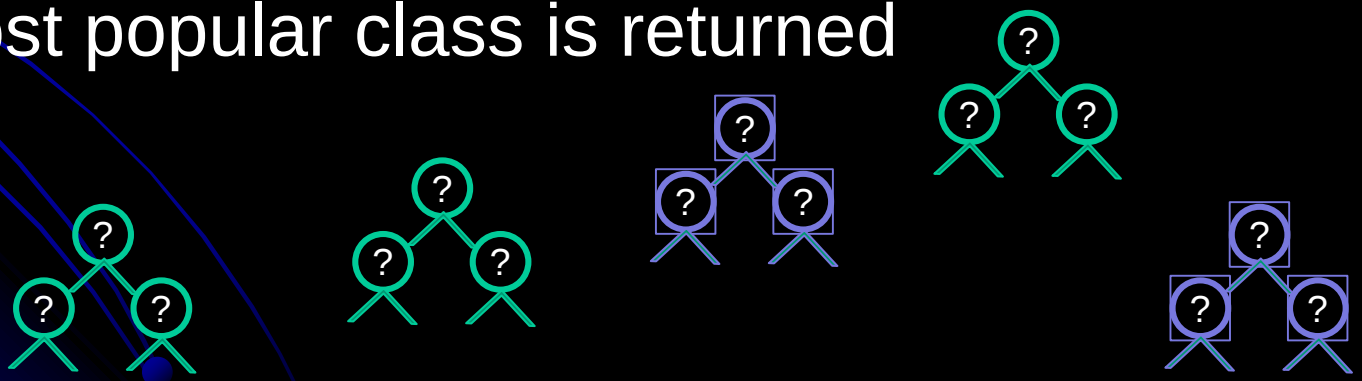
Cons:

- × learning an optimal decision tree is NP-complete
- × perform poorly with many classes and small data
- × computationally expensive to train
- × over-complex trees do not generalize well from the training data (overfitting)

Random Forest (Breiman 2001)

Random Forest:

- Each classifier in the ensemble is a *decision tree* classifier and is generated using a random selection of attributes at each node to determine the split
- During classification, each tree votes and the most popular class is returned



Random Forest (Breiman 2001)

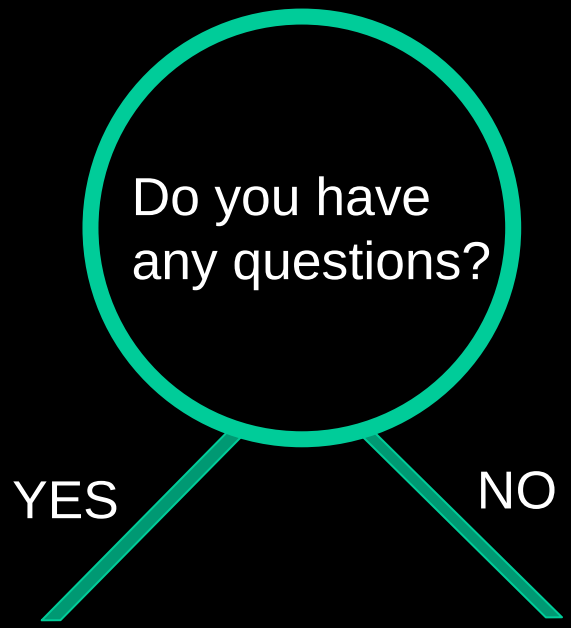
Two Methods to construct Random Forest:

- Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
- Forest-RC (*random linear combinations*)*:* Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)

Random Forest (Breiman 2001)

Comparable in accuracy to Adaboost, but more robust to errors and outliers

Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting


Do you have
any questions?
YES
NO

Linear classifier

Assumption 1: there are two classes of data

Assumption 2: Classes are linearly separable, i.e. ∃ hyperplane that separates the space s.t. data from the two classes lie in different subspaces

Linear classifier

Hyperplane – decision boundary:

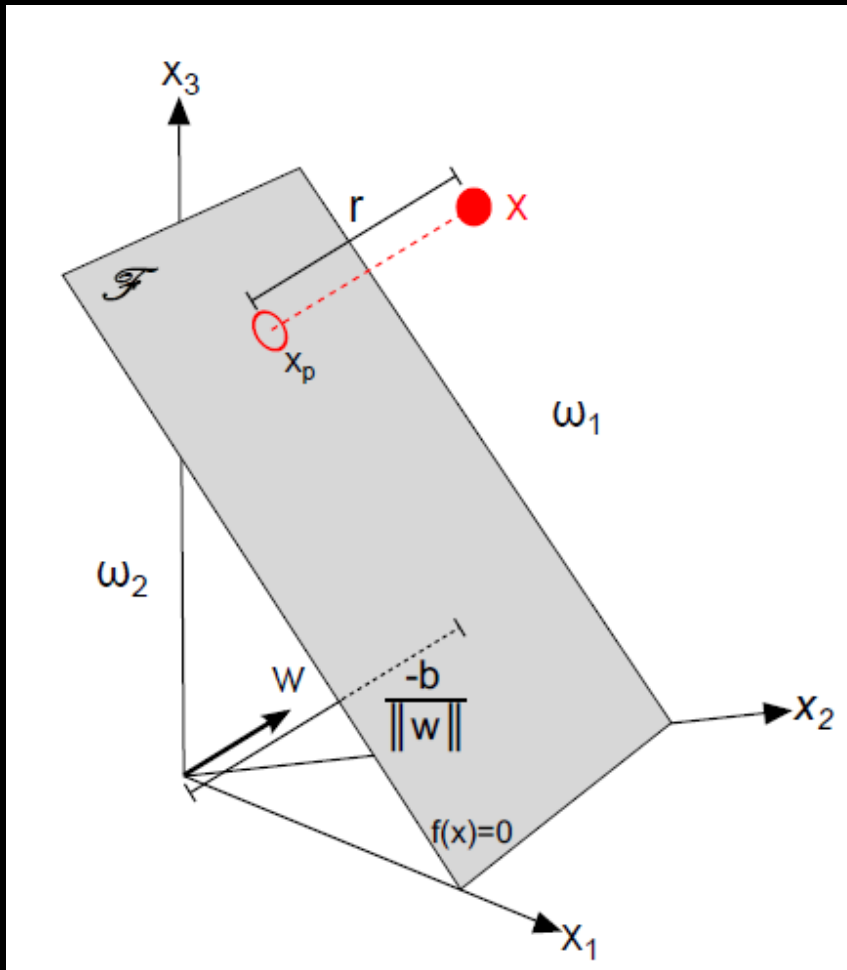
$\mathbf{w}^T\mathbf{x} + b = 0$

w – normal vector

$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$

$f(\mathbf{x}) \geq 0$ for $\mathbf{x} \in \omega_1$

$f(\mathbf{x}) < 0$ for $\mathbf{x} \in \omega_2$



Linear classifier

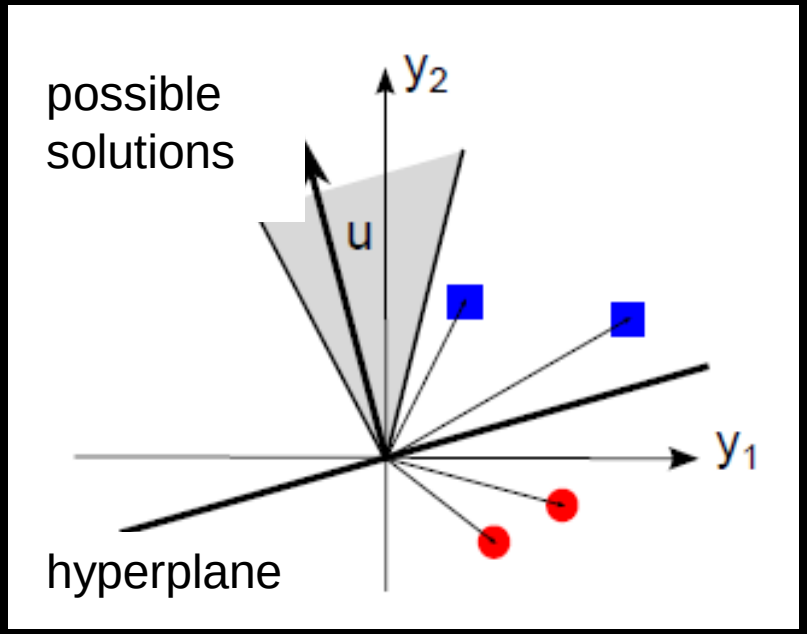
Little change

$f(\mathbf{x}) = \mathbf{u}^T\mathbf{y}$

$\mathbf{u} = [b, \mathbf{w}^T]^T$ and $\mathbf{y} = [1, \mathbf{x}^T]^T$

$\mathbf{y}_i \in \omega_1$ if $\mathbf{u}^T\mathbf{y}_i \geq 0$

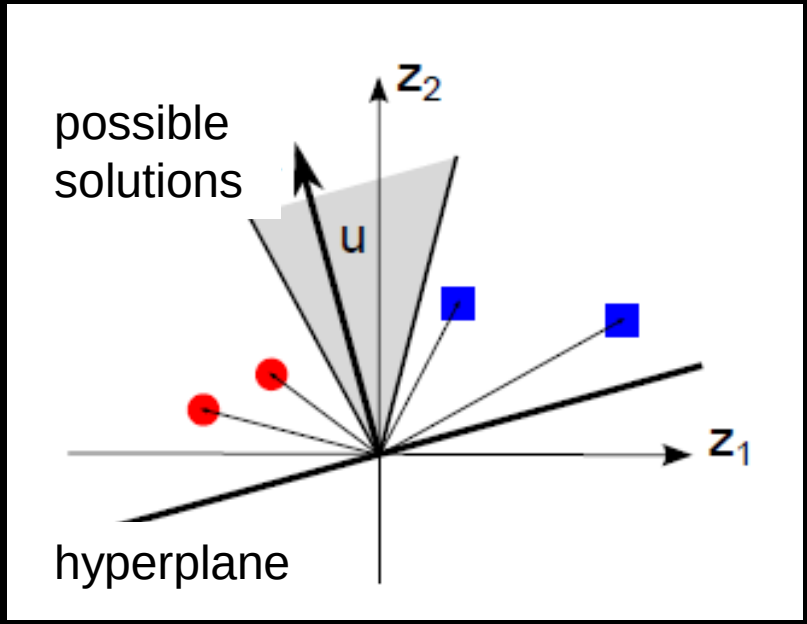
$\mathbf{y}_i \in \omega_2$ if $\mathbf{u}^T\mathbf{y}_i < 0$



Linear classifier

Another trick:

$$\mathbf{z}_i = \begin{cases} \mathbf{y}_i, & \mathbf{y}_i \in \omega_1 \\ -\mathbf{y}_i, & \mathbf{y}_i \in \omega_2 \end{cases}$$



- Final problem:

Find $\mathbf{u}$, s.t. for all $\mathbf{z}_i$

$\mathbf{u}^T \mathbf{z}_i > 0$

Gradient method

Let $O(\mathbf{u})$ be scalar objective function reaching minimum when $\mathbf{u}$ is the solution of the problem

Idea:

If a function $\mathrm{F}(\mathbf{x})$ is defined and differentiable in a neighborhood of a point $\mathbf{y}$, then $\mathrm{F}(\mathbf{x})$ decreases fastest if one goes from $\mathbf{y}$ in the direction of the negative gradient of F at $\mathbf{y}$: $\nabla F|_{\mathbf{y}}$

Gradient method

Iterative method

Start with random vector $\mathbf{u}_1$

Compute $\nabla O|_{\mathbf{u}_1}$

Update the value of $\mathbf{u}$ by choosing a vector from the neighborhood of $\mathbf{u}_1$ in the direction of negative gradient

$$\mathbf{u}_{i+1} = \mathbf{u}_i - \eta(i)\, \nabla O\Big|_{\mathbf{u}_i}$$

where $\eta(i)$ is the learning rate in step i

Gradient method

Objective function is differentiable

Taylor expansion:

$O(\mathbf{a})$ is approximated in the neighborhood of $\mathbf{u}_i$

$$O(\mathbf{a}) \approx O(\mathbf{u}_i) + \nabla O^T(\mathbf{a} - \mathbf{u}_i) + \frac{1}{2}(\mathbf{a} - \mathbf{u}_i)^T \mathbf{H}(\mathbf{a} - \mathbf{u}_i)$$

where **H** is the Hessian matrix of second partial derivatives of function *O*

Gradient method

$$O(\mathbf{a}) \approx O(\mathbf{u}_i) + \nabla O^T(\mathbf{a} - \mathbf{u}_i) + \tfrac{1}{2}(\mathbf{a} - \mathbf{u}_i)^T \mathbf{H}(\mathbf{a} - \mathbf{u}_i)$$

$$\mathbf{u}_{i+1} = \mathbf{u}_i - \eta(i)\nabla O|_{\mathbf{u}_i}$$

Let $\mathbf{a} = \mathbf{u}_{i+1}$

$$O(\mathbf{u}_{i+1}) \approx O(\mathbf{u}_i) - \eta(i)\,\|\nabla O\|^2 + \tfrac{1}{2}\eta^2(i)\nabla O^T \mathbf{H} \nabla O$$

Minimized for

$$\eta(i) = \frac{\|\nabla O\|^2}{\nabla O^T \mathbf{H} \nabla O}$$

Objective function

Piece-wise linear perceptron function

$O_P = \sum_{\mathbf{z}\epsilon\mathcal{Z}} -\mathbf{u}^T\mathbf{z}$

$\mathcal{Z}$– set of misclassified objects

Batch processing – classify all objects

$\nabla O_P = \sum_{\mathbf{z}\epsilon\mathcal{Z}} -\mathbf{z}$

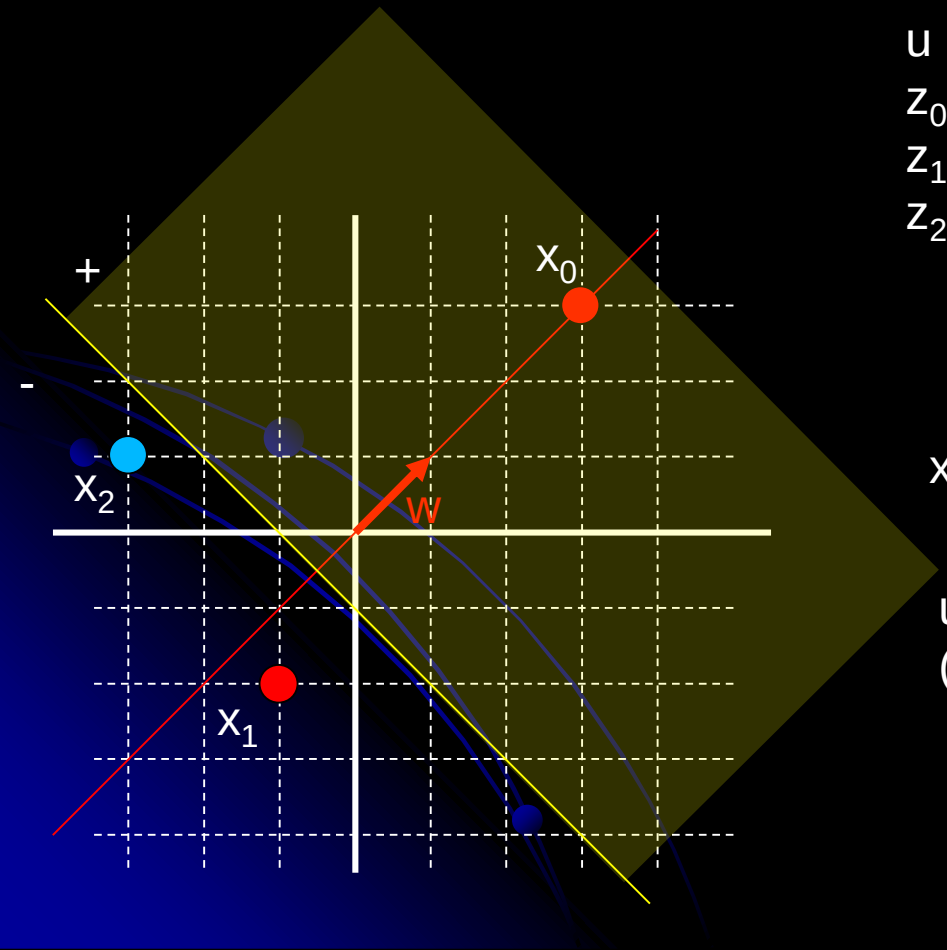
$\mathbf{u}_{i+1} = \mathbf{u}_i + \eta(i) \sum_{\mathbf{z}\epsilon\mathcal{Z}_i} \mathbf{z}$

Sequential learning

Evaluate objects one by one

set $\mathbf{u}_0 = \mathbf{1}$, k = 0, i = 0
repeat
  if $\mathbf{z}_k$ is misclassified
    $\mathbf{u}_{i+1} = \mathbf{u}_i + \mathbf{z}_k$
    i = i+1
  endif
  k = (k+1) mod N
until convergence ($\mathbf{u}^T \mathbf{z}_j \geq 0$ for all $\mathbf{z}_j$)

Sequential example



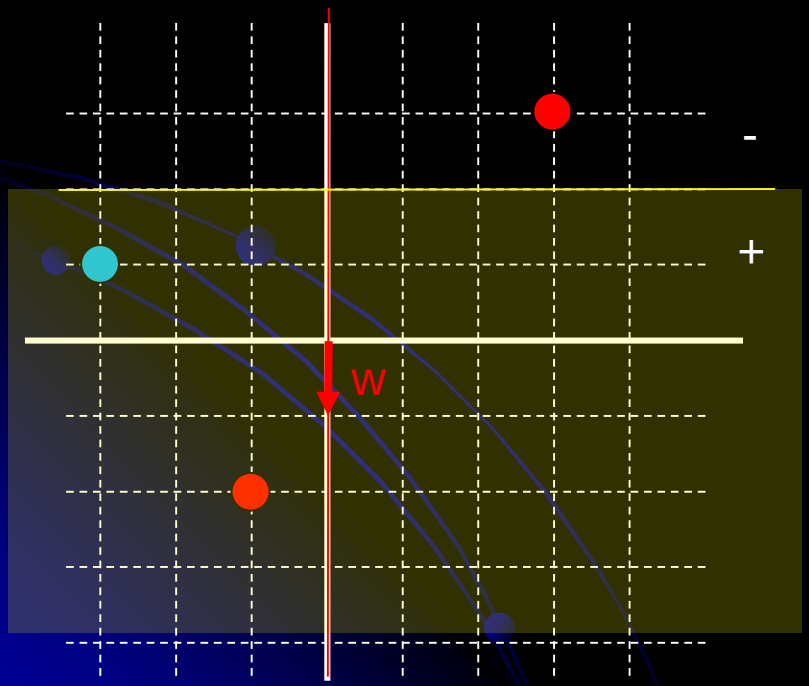
$u = (1,1,1)^T$
$z_0 = (1,3,3)^T$
$z_1 = (1,-1,-2)^T$
$z_2 = -(1,-3,1)^T$

$w = (1,1)^T$
$x_0 = (3,3)^T$
$x_1 = (-1,-2)^T$
$x_2 = (-3,1)^T$

x_1 is misclassified

$u = u + z_1 = (1,1,1) + (1,-1,-2) = (2,0,-1)$

Sequential example

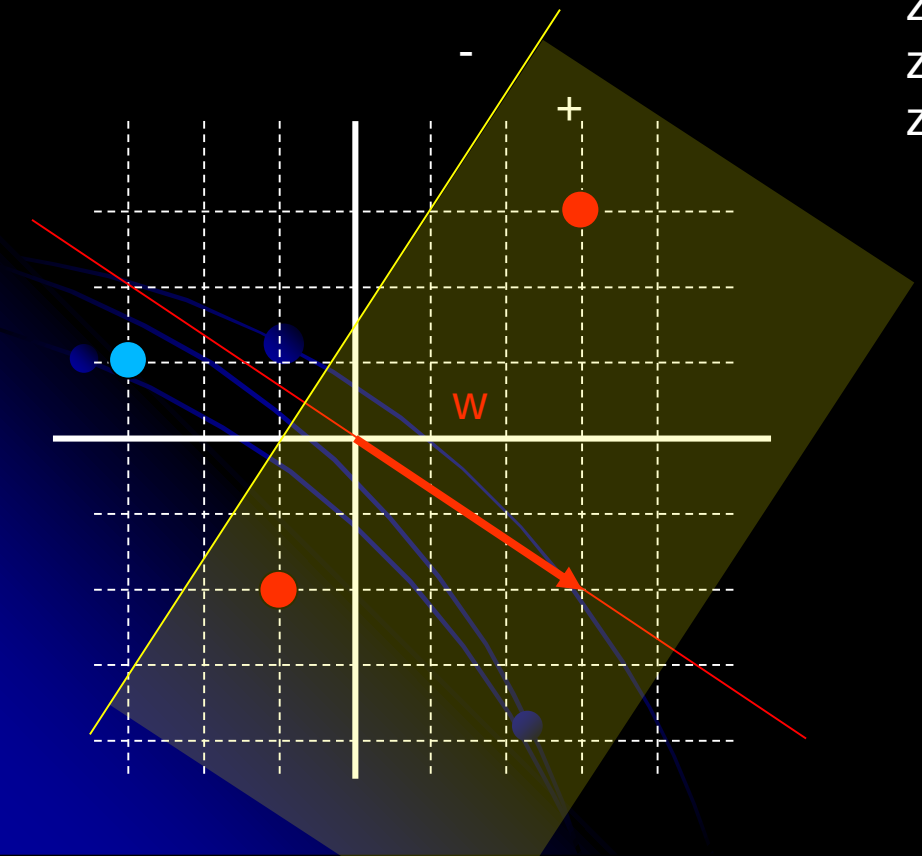


$u = (2,0,-1)^T$
$z_0 = (1,3,3)^T$
$z_1 = (1,-1,-2)^T$
$z_2 = -(1,-3,1)^T$

x_2 is misclassified

$u = u + x_2 = (2,0,-1) - (1,-3,1) = (3,3,-2)$

Sequential example



$u = (3,3,-2)^T$
$z_0 = (1,3,3)^T$
$z_1 = (1,-1,-2)^T$
$z_2 = -(1,-3,1)^T$

Objective function 2

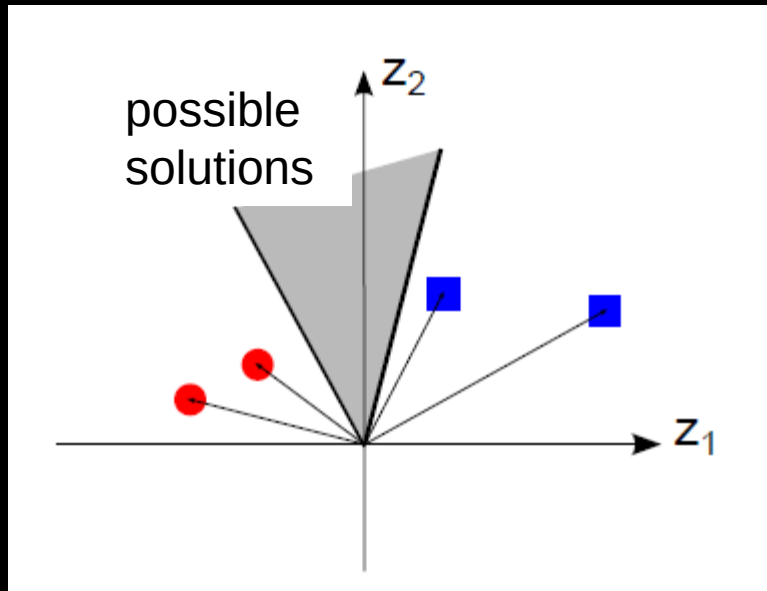
Quadratic function

$O_Q = \sum_{\mathbf{z}\epsilon\mathcal{Z}} (\mathbf{u}^T \mathbf{z})^2$

Continuous gradient

Too smooth

Can converge on the boundary



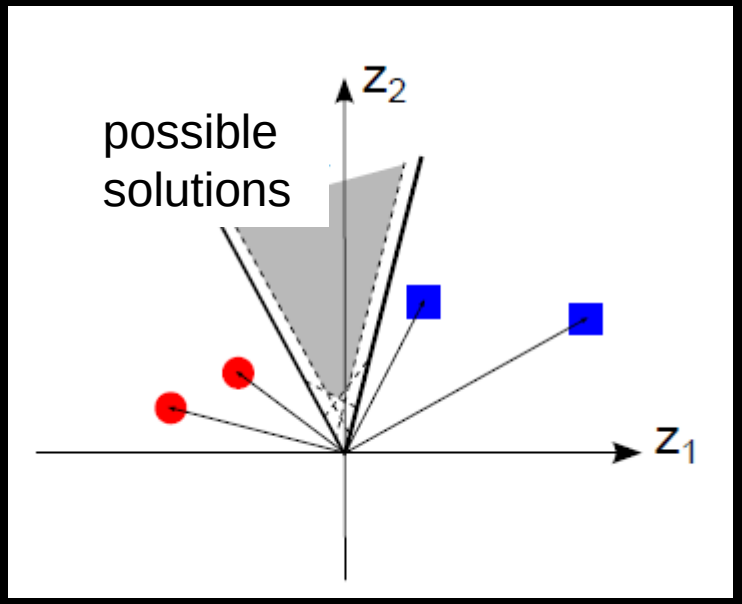
Objective function 3

Normalization: $\|\mathbf{z}\|^2$

Border in the space of possible solutions: ε

$$O_R = \frac{1}{2}\sum_{\mathbf{z}\epsilon\mathcal{Z}} \frac{(\mathbf{u}^T\mathbf{z}-\varepsilon)^2}{\|\mathbf{z}\|^2}$$

$$\nabla O_R = \sum_{\mathbf{z}\epsilon\mathcal{Z}} \frac{\mathbf{u}^T\mathbf{z}-\varepsilon}{\|\mathbf{z}\|^2}\mathbf{z}$$



Sequential learning

set $\mathbf{u}_0 = \mathbf{0}$, k = 0, i = 0
repeat
    if $\mathbf{u}^T \mathbf{z}_k \leq \varepsilon$
        $\mathbf{z} = \mathbf{z}_k$
        $\mathbf{u}_{i+1} = \mathbf{u}_i - \eta(i) \sum_{\mathbf{z} \epsilon \mathcal{Z}} \frac{\mathbf{u}^T \mathbf{z} - \varepsilon}{\|\mathbf{z}\|^2} \mathbf{z}$
        i = i+1
    endif
    k = (k+1) mod N
until $\mathbf{u}^T \mathbf{z}_j > \varepsilon$ for all $\mathbf{z}_j$