Data Mining algorithms

2017-2018 spring

02.14-16.2016

1. Evaluation II.
2. Decision Trees
3. Linear Separator

Plan

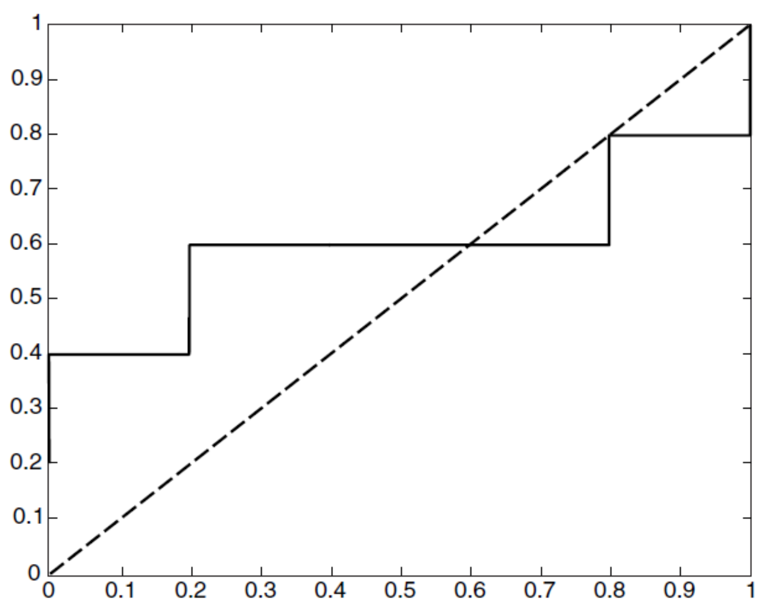
W1 Februar 7-9: Introduction, kNN, evaluation

W2 Februar 14-16: Evaluation, Decision Trees

W3 Februar 21-23: Linear separators, iPython, VC theorem
W4 Februar 28-march 2: SVM, VC theorem and Bottou-Bousquet
W5 March 7-9: clustering (hierarchical, density based etc.), GMM
W6 March 14-16: GMM, MRF, Apriori and association rules
W7 March 21-23: Recommender systems and generative models
W8 March 28-30: basics of neural networks, Sontag-Maas-Bartlett theorems, Bayes networks
W9 April 4-6: BN, CNN, MLP
W10 April 11-13: Dropout, Batch normalization
W11 April 18-20: midterm, RNN
W12 April 25-27: LSTM, GRU, attention, Image caption, Turing Machine
W13 May 2-4: RBM, DBN, VAE, GAN, Boosting, Time series
W14 May 9-11: TS, Projects on Friday

ROC: Receiver Operating Characteristic

Class	+	−	+	−	−	−	+	−	+	+	
	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0



AUC=?

+	+	-	+	-	-	+	+	-	+	
0.16	0.32	0.42	0.44	0.45	0.51	0.78	0.87	0.91	0.93	Score
										TP
										FN
										TN
										FP
										TPR
										FPR

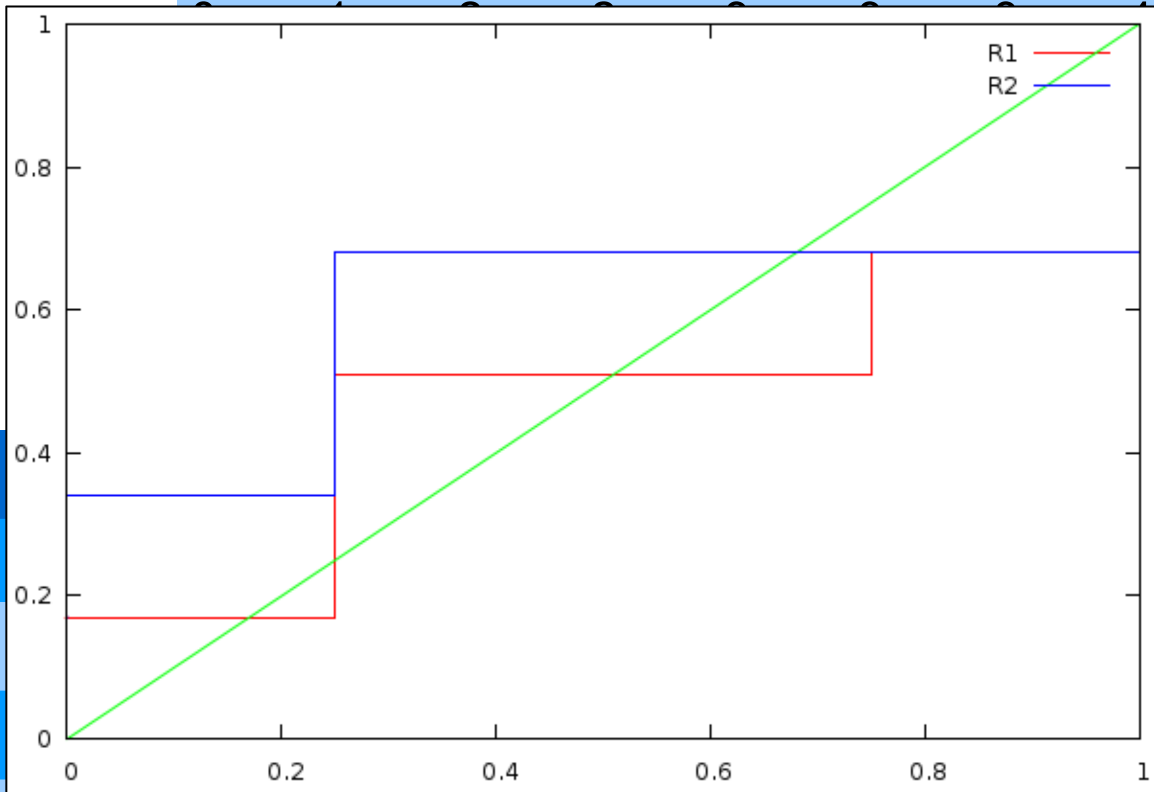
Some exercise:
How to compare models?
AUC?

+	+	-	-	-	+	+	-	+	+	
0.43	0.56	0.62	0.78	0.79	0.86	0.89	0.89	0.91	0.96	Score
										TP
										FN
										TN
										FP
										TPR
										FPR

+	+	-	+	-	-	+	+	-	+	
6	5	4	4	3	3	3	2	1	1	TP
0	1	2	2	3	3	3	4	5	5	FN
0	0	0	1	1	2	3	3	3	4	TN
4	4	4	3	3	2	1	1	1	0	FP
1	5/6	4/6	4/6	3/6	3/6	3/6	2/6	1/6	1/6	TPR
1	1	1	3/4	3/4	2/4	1/4	1/4	1/4	0	FPR

+	+	-	-	-	+	+	-	+	+	
6	5	4	4	4	4	3	2	2	1	TP
0	1	2	2	2	2	3	4	4	5	FN
0	0	0	1	2	3	3	3	4	4	TN
4	4	4	3	2	1	1	1	0	0	FP
1	5/6	4/6	4/6	4/6	4/6	3/6	2/6	2/6	1/6	TPR
1	1	1	3/4	2/4	1/4	1/4	1/4	0	0	FPR

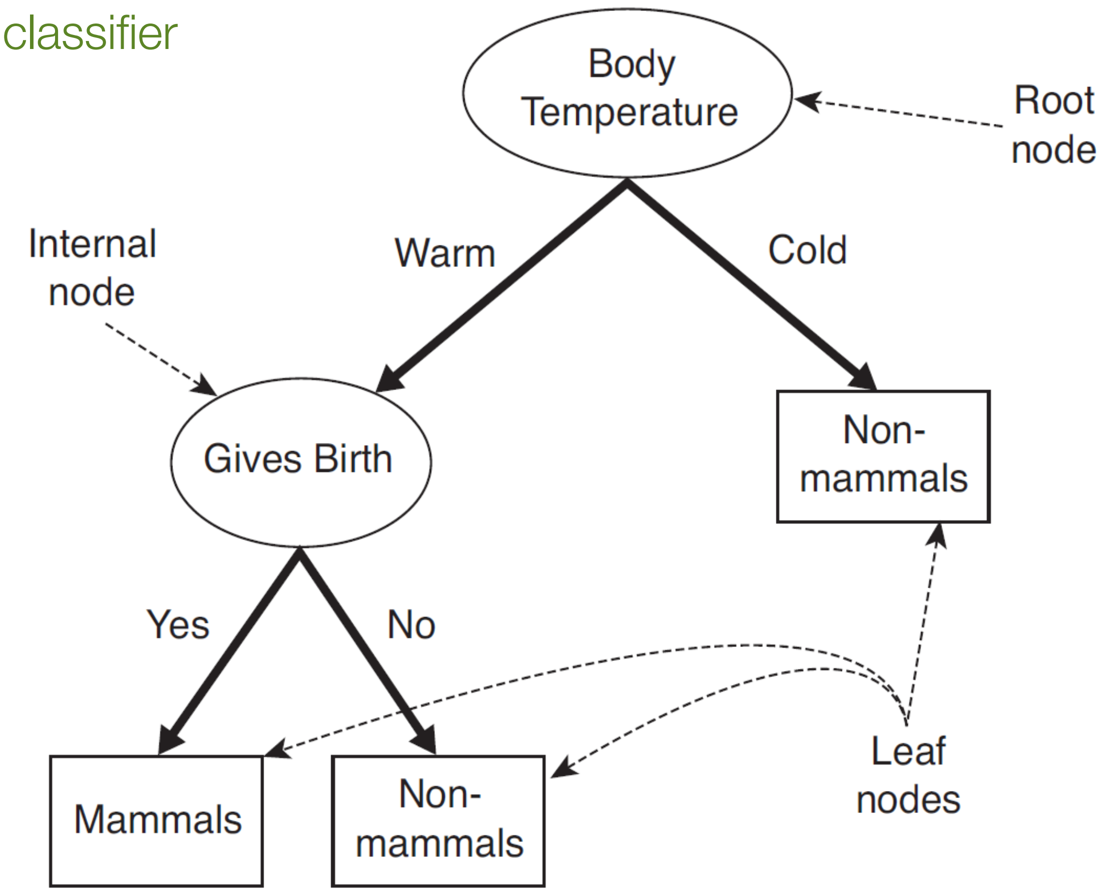
+	+	-	+	-	-	+	+	-	+	
6	5	4	4	3	3	3	2	1	1	TP
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	5	5	FN
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3	4	TN
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1	0	FP
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1/6	1/6	TPR
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1/4	0	FPR



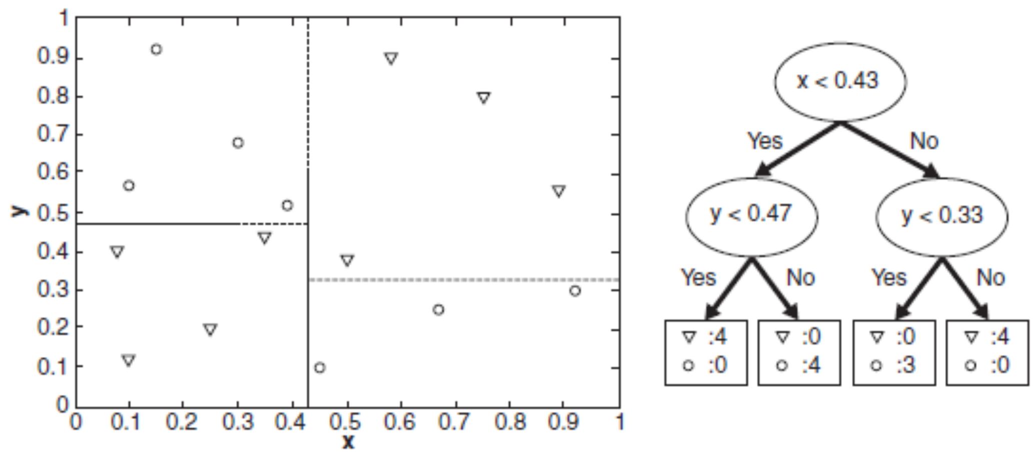
+	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	
6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	TP
0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	FN
0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	TN
4	4	4	3	2	1	1	1	0	0	FP
1	5/6	4/6	4/6	4/6	4/6	3/6	2/6	2/6	1/6	TPR
1	1	1	3/4	2/4	1/4	1/4	1/4	0	0	FPR

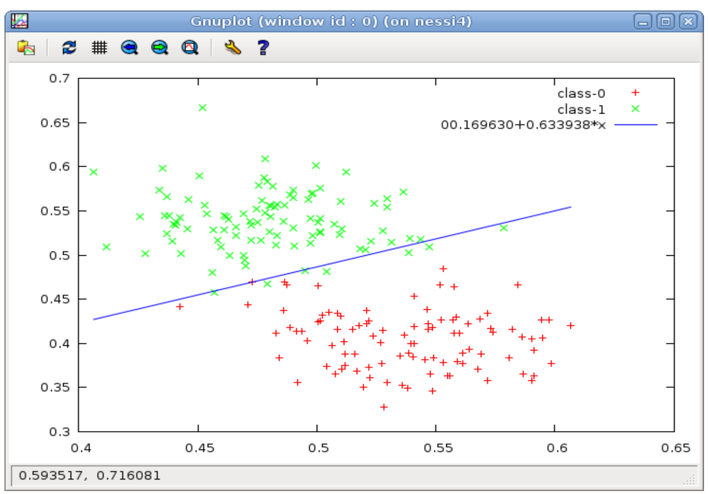
Decision trees

E.g. mammal classifier



Decision tree

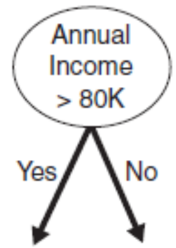


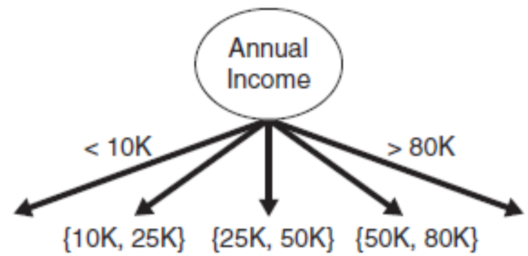


DT?

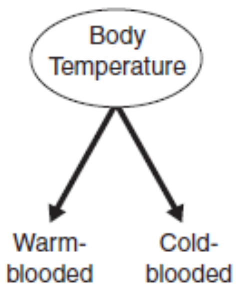
Decision tree

Multiple types of splits

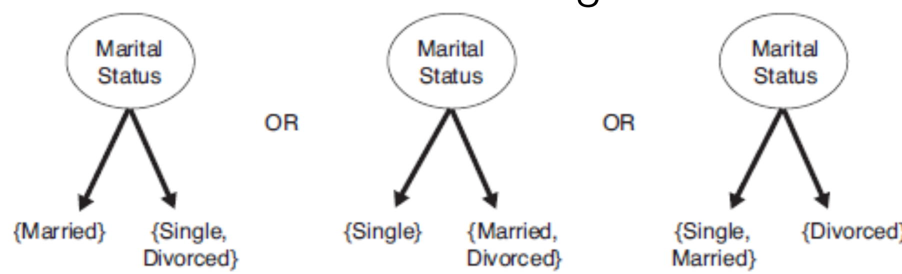




According a scale

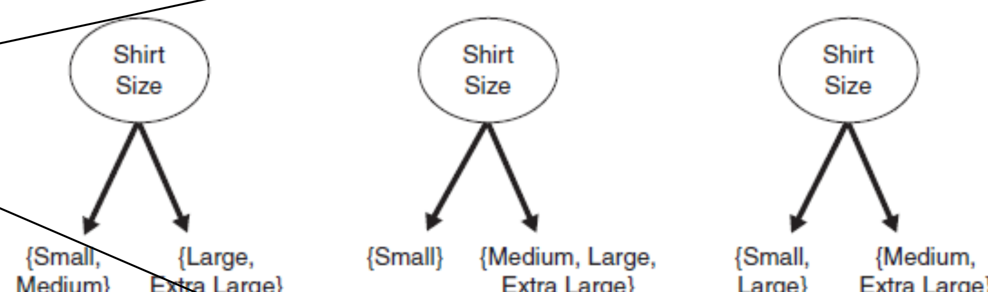


binary



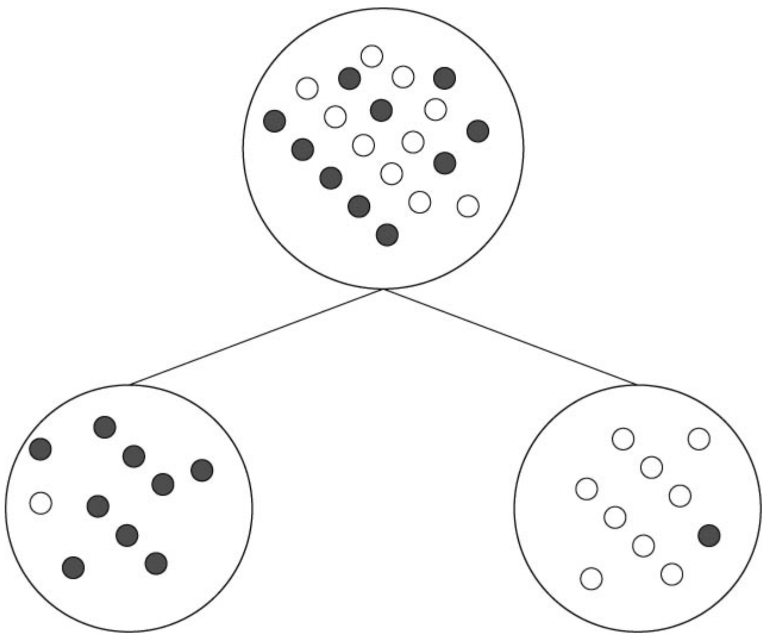
nominal

What is
the difference?

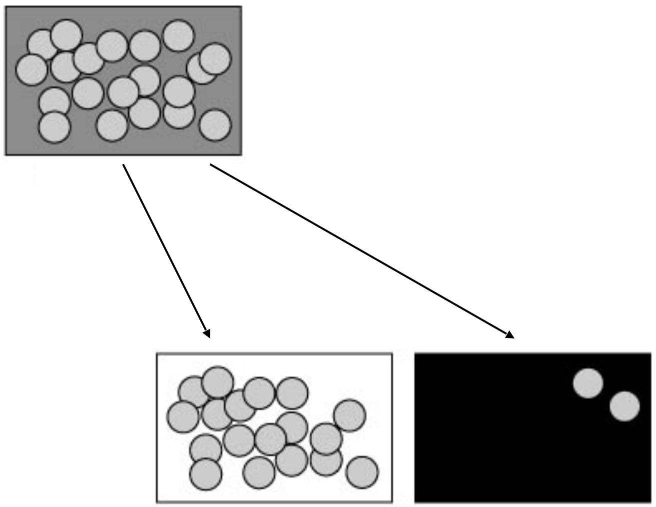


ordinal

Decision tree



A good split



A not so good one

Decision tree

Presumption: attributes are nominal

Procedure TreeBuilding (D)

If (the data is not classified correctly)

Find best splitting attribute A

For each a element in A

Create child node N_a

D_a all instances in D where A=a

TreeBuilding (D_a)

Endfor

Endif

EndProcedure

Decision tree

What is a good splitting attribute?

- results homogeneous child nodes (separates instances with different class labels)
- balanced (splits into similarly sized nodes)

To measure it we use “purity” measures:

- missclassification error
- entropy
- Gini
- or whatever works/suitable

Decision tree

Missclassification error:

p(i,t): the proportion of instances with class label i in node t

Classification error: 1-max(p(i,t))

Gain:

$$\Delta = I(\text{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

where I(parent) is the parent nodes purity measure
and the chosen attribute resulted a split into k child nodes

Decision tree

Small example:

Should we choose A or B as a splitting attribute?

A	B	Class Label
T	F	+
T	T	+
T	T	+
T	F	−
T	T	+
F	F	−
F	F	−
F	F	−
T	T	−
T	F	−

Decision tree

Small example:

Should we choose A or B as a splitting attribute?

A	B	Class Label
T	F	+
T	T	+
T	T	+
T	F	–
T	T	+
F	F	–
F	F	–
F	F	–
T	T	–
T	F	–

Error A:

MCE = 0+3/7

Error B:

MCE = 1/4+1/6

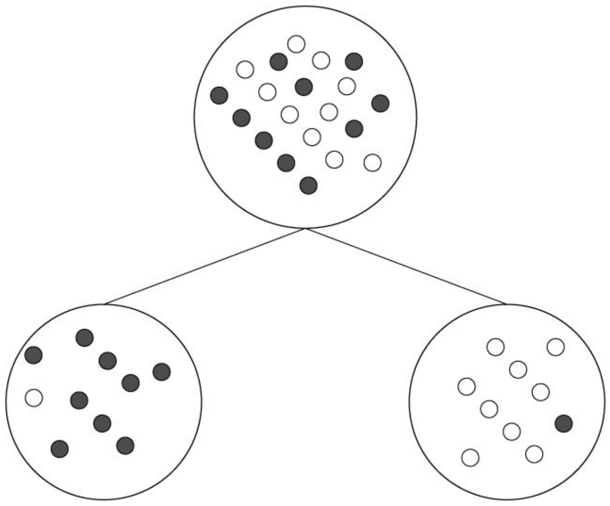
Choose B! or?

Decision tree

Gini (population diversity)

p(i|t) : proportion of instances with class label i in node t

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2$$



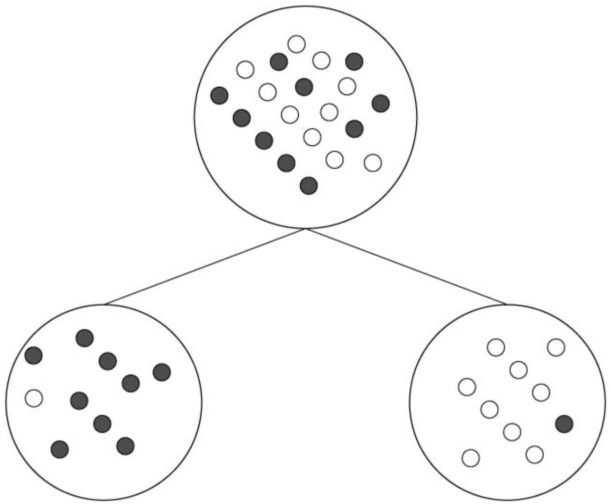
Gini in the parent/root?
Gini in the child nodes?

Decision tree

Gini (population diversity)

p(i|t) : proportion of instances with class label i in node t

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2$$



Gini(parent)= 0.5 = 0.5^2 + 0.5^2

Gini(child)= 0.82 = 0.1^2 + 0.9^2

Decision tree

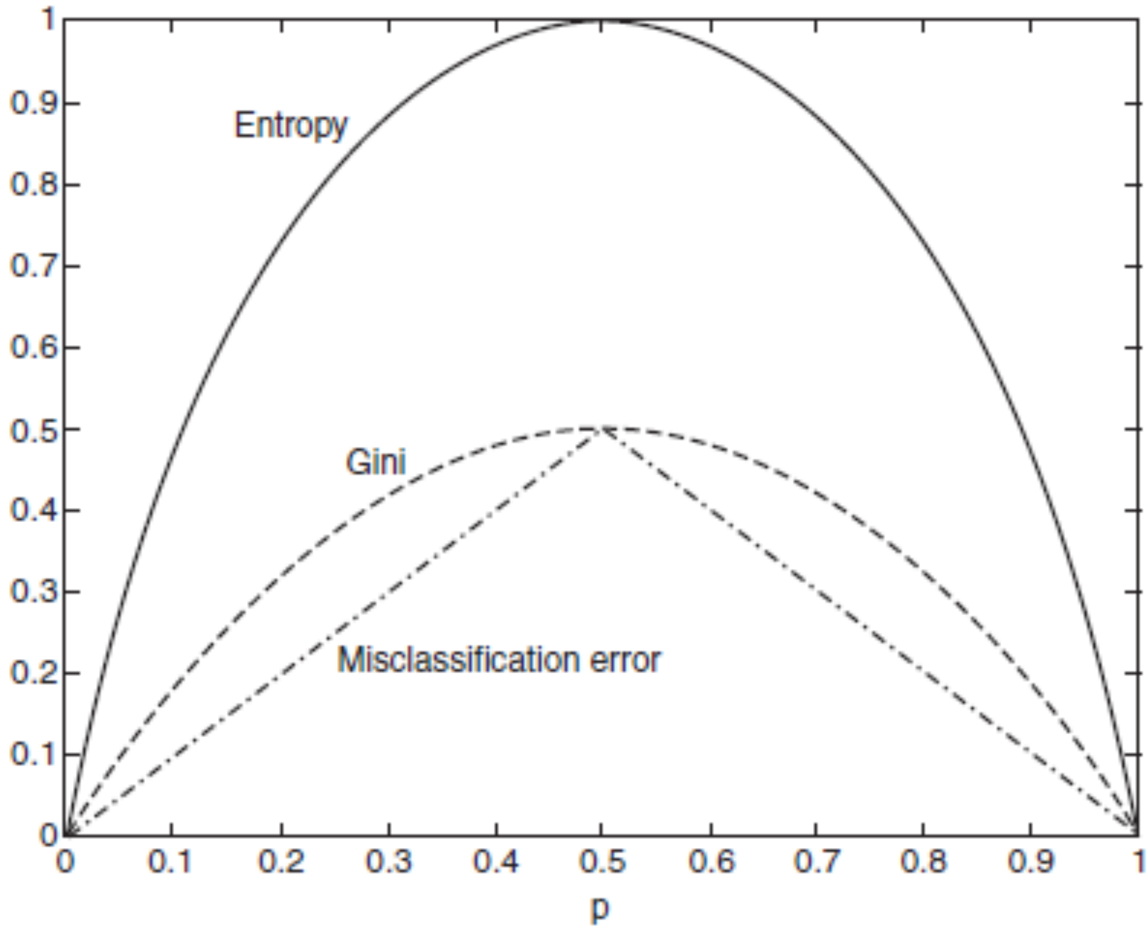
Entropy

p(i|t) : proportion of instances with class label i in node t

$$I(v_j) - \sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t)$$

All three of the measures are having a peak value at 0.5 and they prefer splits into multiple nodes.

Decision tree



Example

$$\text{Entropy}(t) = -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t),$$

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1}[p(i|t)]^2,$$

$$\text{Classification error}(t) = 1 - \max_i[p(i|t)],$$

Node N_1	Count
Class=0	0
Class=1	6

$\text{Gini} = 1 - (0/6)^2 - (6/6)^2 = 0$
$\text{Entropy} = -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$
$\text{Error} = 1 - \max[0/6, 6/6] = 0$

Node N_2	Count
Class=0	1
Class=1	5

$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$
$\text{Entropy} = -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$
$\text{Error} = 1 - \max[1/6, 5/6] = 0.167$

Node N_3	Count
Class=0	3
Class=1	3

$\text{Gini} = 1 - (3/6)^2 - (3/6)^2 = 0.5$
$\text{Entropy} = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$
$\text{Error} = 1 - \max[3/6, 3/6] = 0.5$

Decision tree

Notes:

DTs can handle both nominal and numerical features (dates and strings?)
easily interpretable
Robust to noise (is it?)
But some subtrees can occur multiple times
Overfitting is a real issue

Why?

Typical problems:

- too deep and wide trees with less train instances in the leaves
- unbalanced training set (not just DT issue)

Solution: pruning!

Some preliminaries:

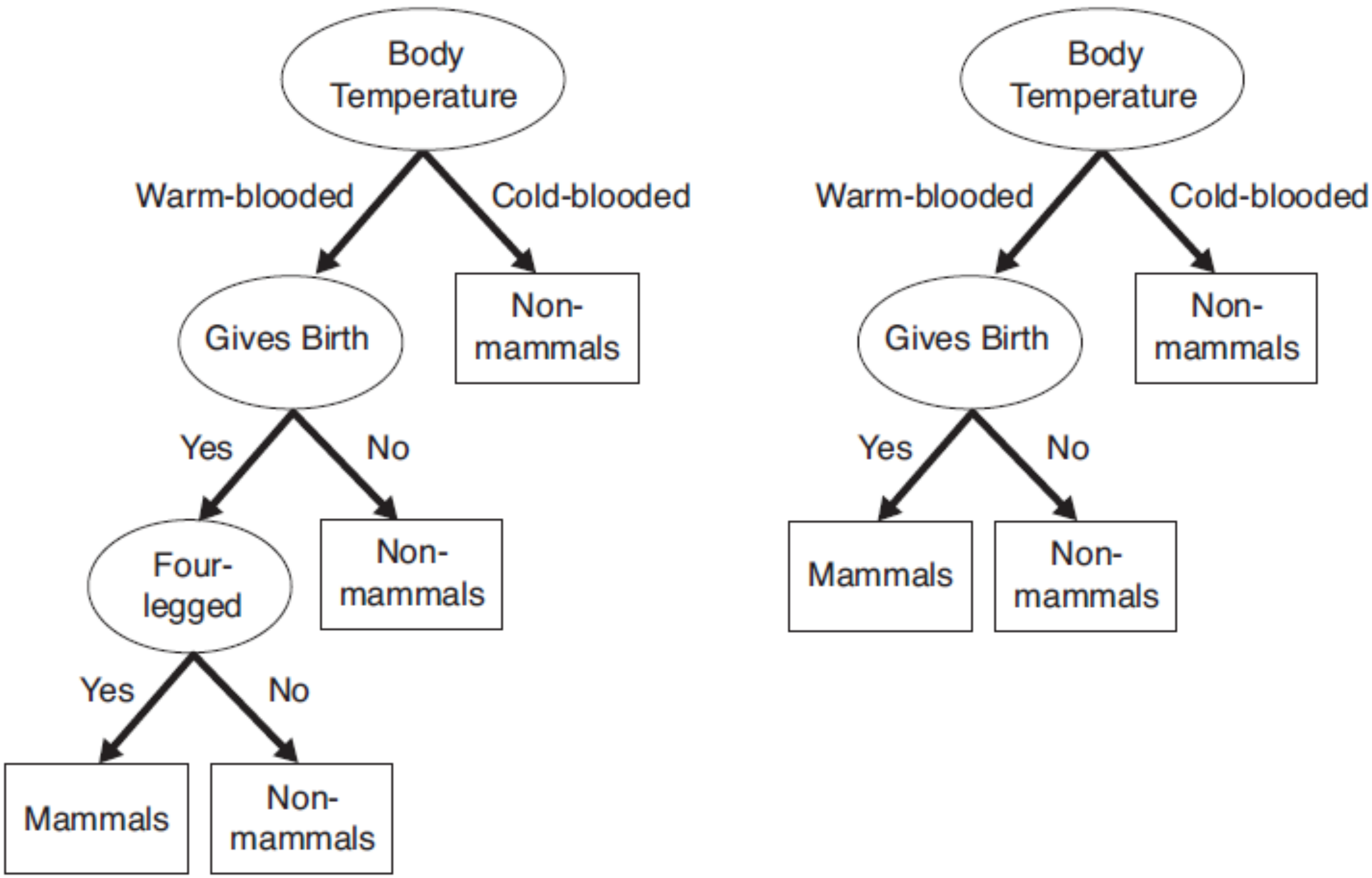
- always prefer the less complex model with the same performance (Minimum description length, MDL)
- early and post-pruning
- MDL(Minimum Description Length):

E.g.: what will happen with a dolphin?

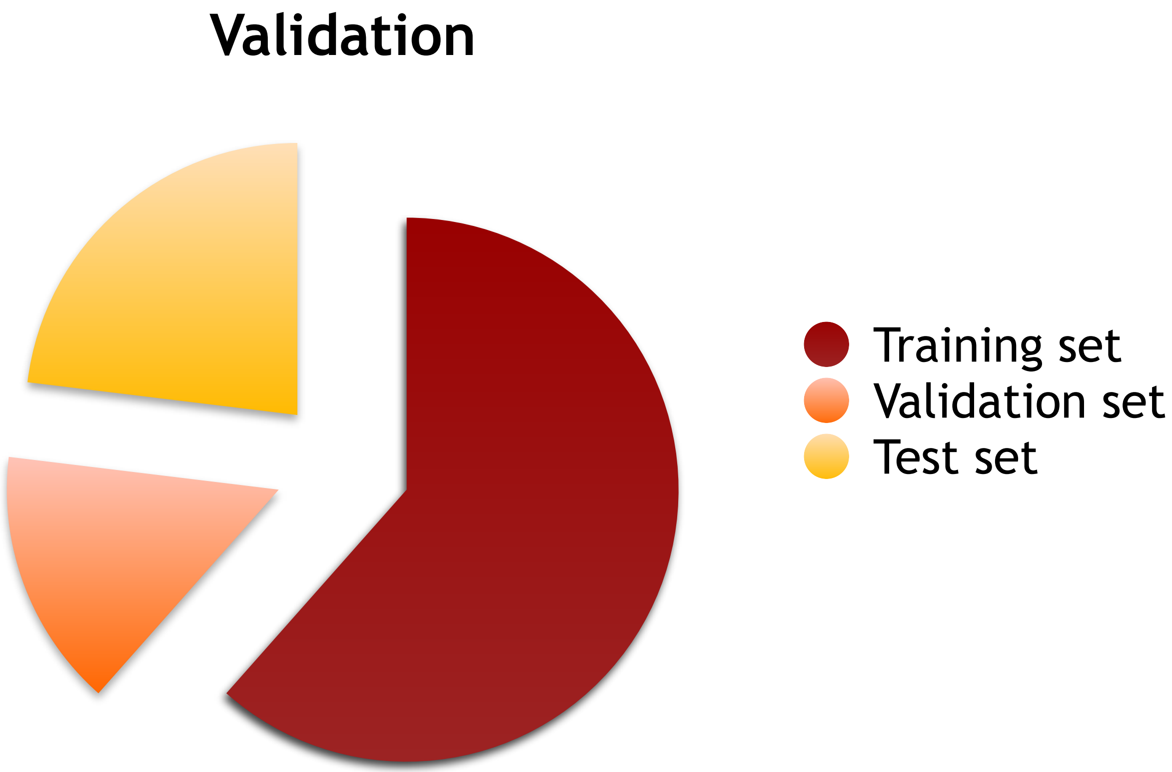
Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
porcupine	warm-blooded	yes	yes	yes	yes
cat	warm-blooded	yes	yes	no	yes
bat	warm-blooded	yes	no	yes	no*
whale	warm-blooded	yes	no	no	no*
salamander	cold-blooded	no	yes	yes	no
komodo dragon	cold-blooded	no	yes	no	no
python	cold-blooded	no	no	yes	no
salmon	cold-blooded	no	no	no	no
eagle	warm-blooded	no	no	no	no
guppy	cold-blooded	yes	no	no	no

Decision tree

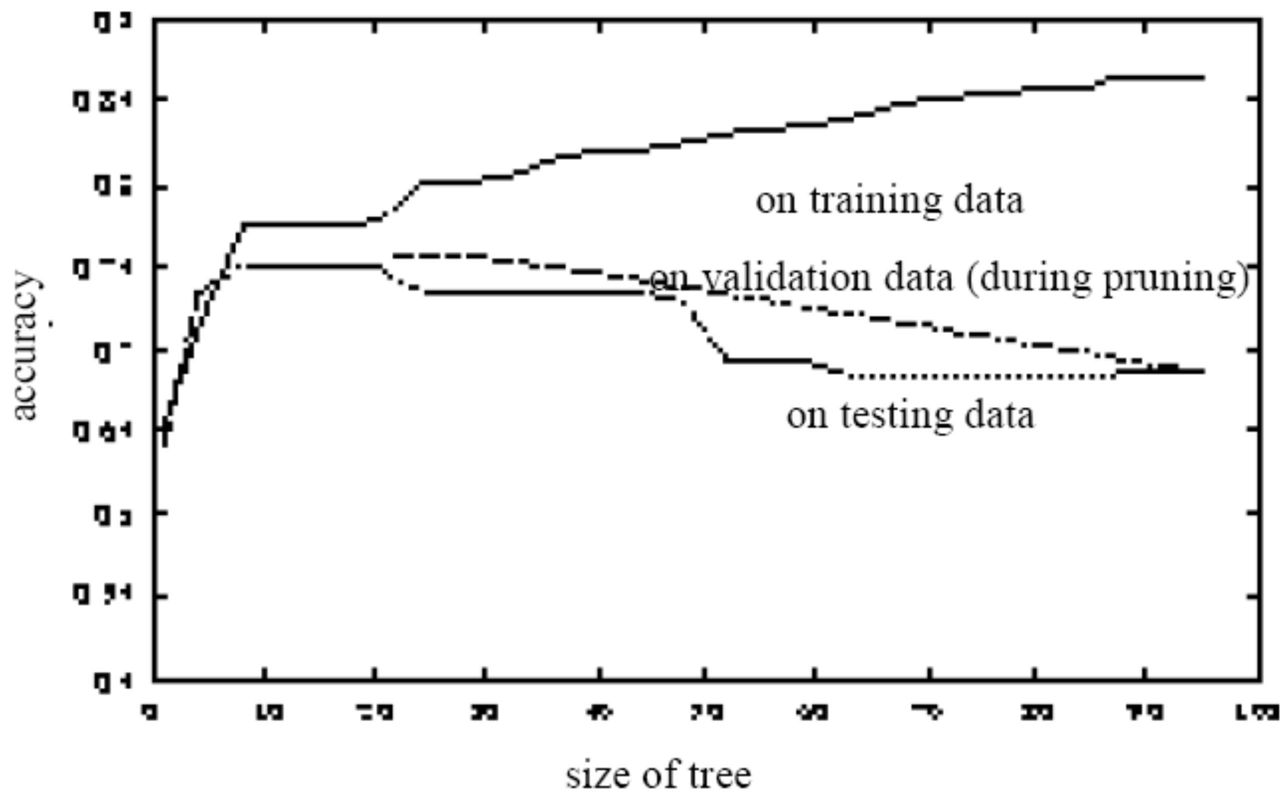
Noise?



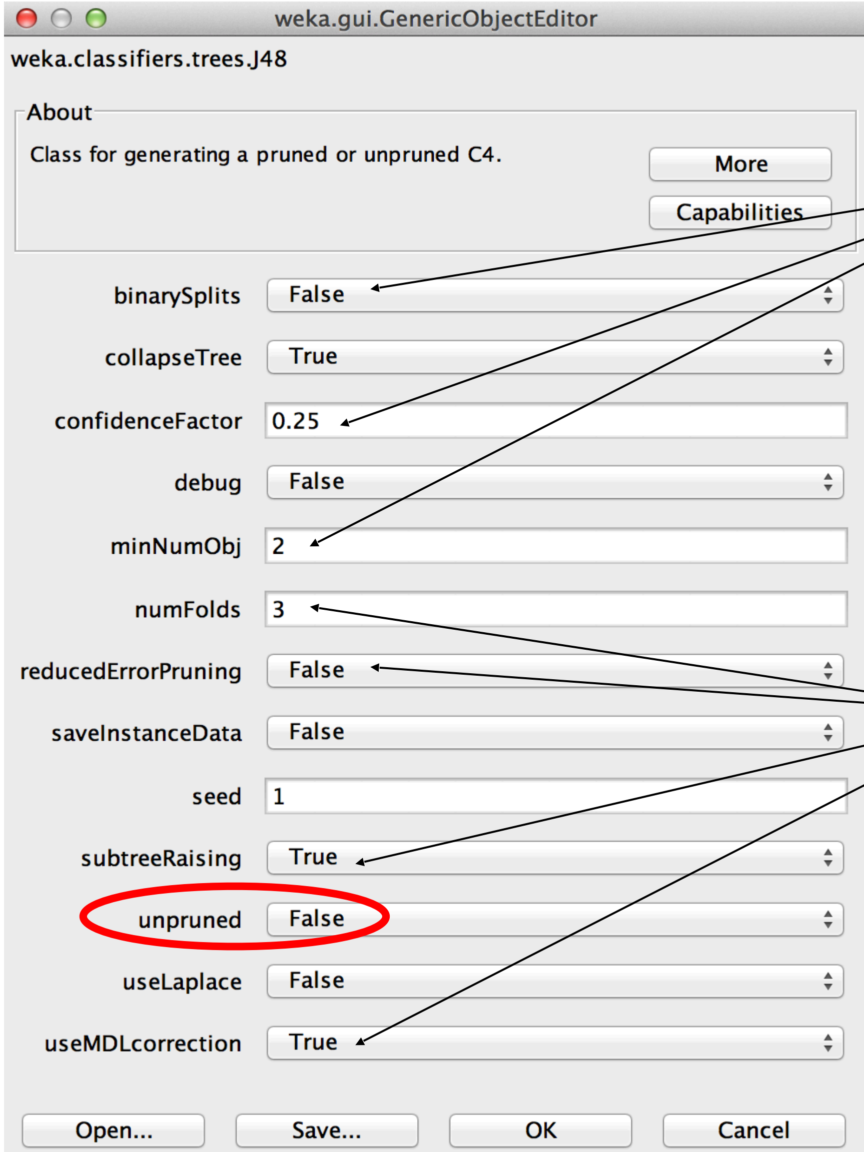
Validation



Validation


accuracy
on training data
on validation data (during pruning)
on testing data
size of tree

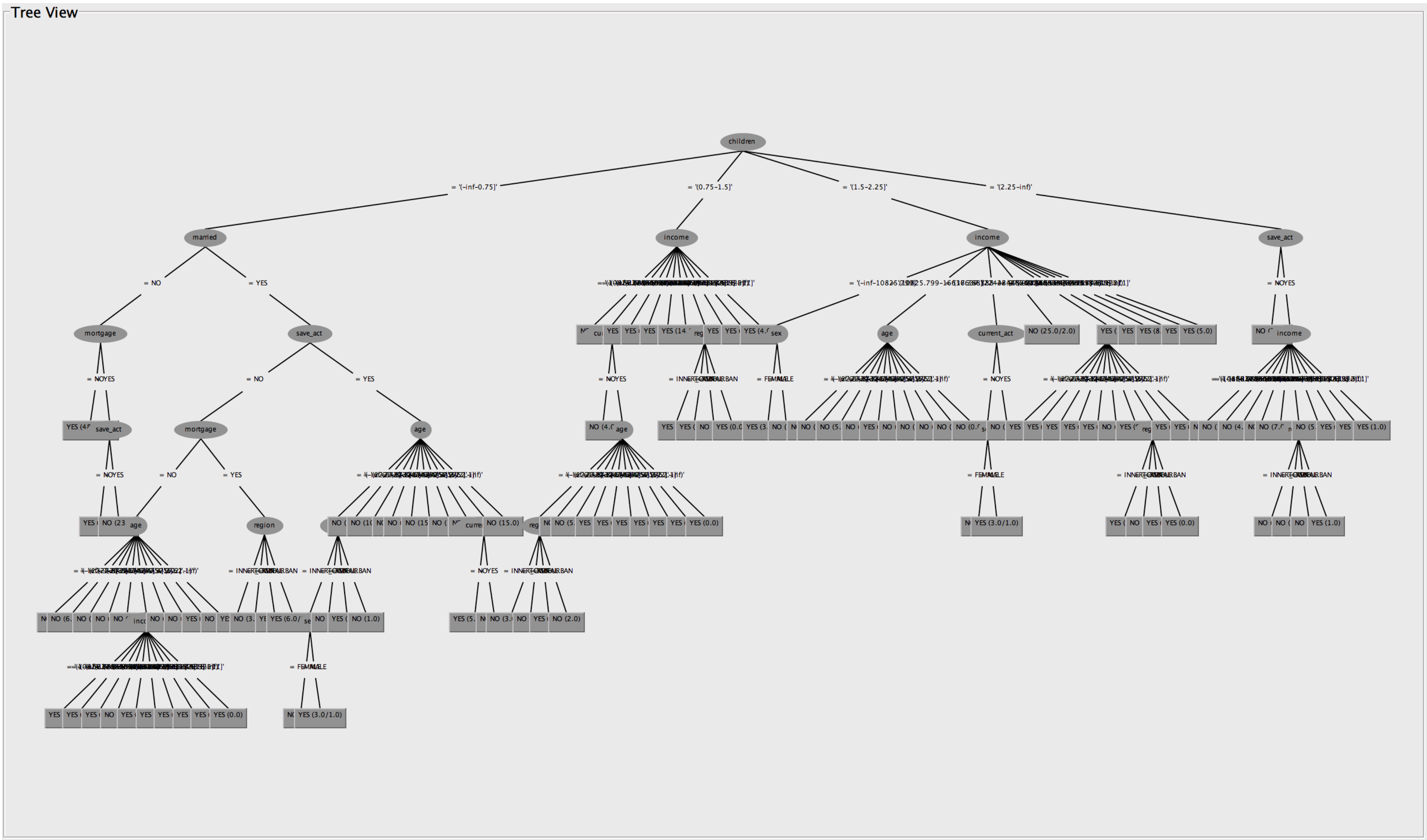
Example: Quinlan C4.5 in Weka



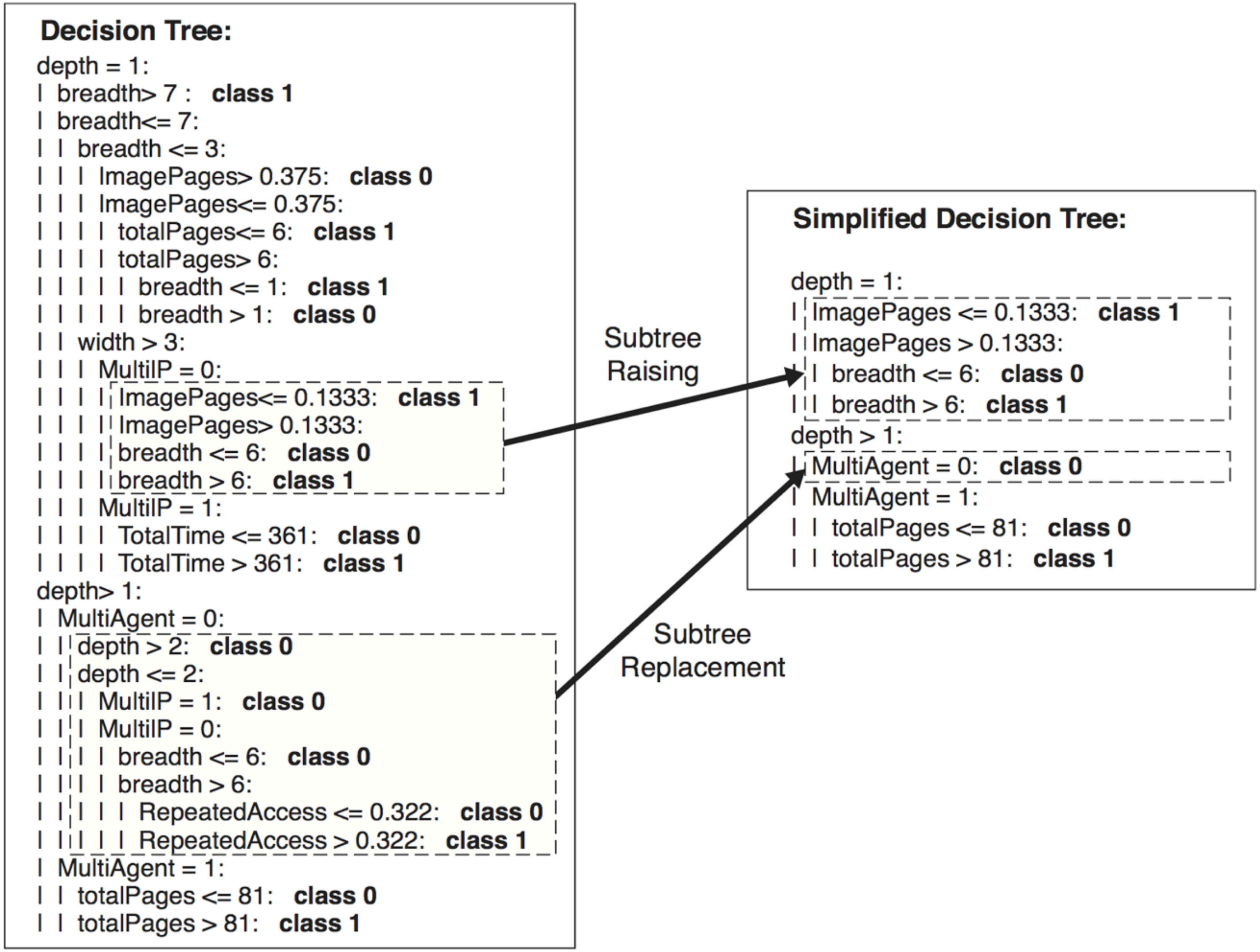
Pre-pruning:
Stop growing

Post-pruning:
After building the tree we remove or even change some parts of the tree

Unpruned tree



Subtree raising vs. replacement (rep)



Subtree raising

Number of leaves: 20
Size of the tree: 39

```
income <= 51284.3
|   children <= 1
|   |   children <= 0
|   |   |   married = NO
|   |   |   |   mortgage = NO: YES (43.0/3.0)
|   |   |   |   mortgage = YES
|   |   |   |   |   save_act = NO: YES (12.0)
|   |   |   |   |   save_act = YES: NO (22.0)
|   |   |   married = YES
|   |   |   |   save_act = NO
|   |   |   |   |   mortgage = NO
|   |   |   |   |   |   income <= 21506.2
|   |   |   |   |   |   |   age <= 41: NO (11.0/1.0)
|   |   |   |   |   |   |   age > 41: YES (5.0/1.0)
|   |   |   |   |   |   income > 21506.2: NO (20.0)
|   |   |   |   |   mortgage = YES: YES (25.0/3.0)
|   |   |   |   save_act = YES: NO (115.0/12.0)
|   |   children > 0
|   |   |   income <= 15538.8
|   |   |   |   age <= 41
|   |   |   |   |   income <= 12683.6: NO (14.0)
|   |   |   |   |   income > 12683.6
|   |   |   |   |   |   income <= 13381: YES (2.0)
|   |   |   |   |   |   income > 13381: NO (6.0)
|   |   |   |   age > 41: YES (2.0)
|   |   |   income > 15538.8: YES (101.0/5.0)
|   children > 1
|   |   income <= 30404.3: NO (124.0/12.0)
|   |   income > 30404.3
|   |   |   children <= 2: YES (36.0/5.0)
|   |   |   children > 2
|   |   |   |   income <= 44288.3: NO (19.0/2.0)
|   |   |   |   income > 44288.3: YES (6.0)
income > 51284.3
|   children <= 0
|   |   age <= 62: YES (4.0)
|   |   age > 62: NO (6.0/1.0)
|   children > 0: YES (27.0)
```

Number of leaves: 17
Size of the tree: 33

```
children <= 1
|   children <= 0
|   |   married = NO
|   |   |   mortgage = NO: YES (48.0/3.0)
|   |   |   mortgage = YES
|   |   |   |   save_act = NO: YES (12.0)
|   |   |   |   save_act = YES: NO (23.0)
|   |   married = YES
|   |   |   save_act = NO
|   |   |   |   mortgage = NO
|   |   |   |   |   income <= 21506.2
|   |   |   |   |   |   age <= 41: NO (11.0/1.0)
|   |   |   |   |   |   age > 41: YES (5.0/1.0)
|   |   |   |   |   income > 21506.2: NO (20.0)
|   |   |   |   mortgage = YES: YES (25.0/3.0)
|   |   |   save_act = YES: NO (119.0/12.0)
|   children > 0
|   |   income <= 15538.8
|   |   |   age <= 41
|   |   |   |   income <= 12683.6: NO (14.0)
|   |   |   |   income > 12683.6
|   |   |   |   |   income <= 13381: YES (2.0)
|   |   |   |   |   income > 13381: NO (6.0)
|   |   |   age > 41: YES (2.0)
|   |   income > 15538.8: YES (111.0/5.0)
children > 1
|   income <= 30404.3: NO (124.0/12.0)
|   income > 30404.3
|   |   children <= 2: YES (51.0/5.0)
|   |   children > 2
|   |   |   income <= 44288.3: NO (19.0/2.0)
|   |   |   income > 44288.3: YES (8.0)
```

Effect of pruning

Subtree replacement vs. raising

Training set

ID	vehicle	color	acceleration
Train 1	motorbike	red	high
Train 2	motorbike	blue	high
Train 3	car	blue	high
Train 4	motorbike	blue	high
Train 5	car	green	small
Train 6	car	blue	small
Train 7	car	blue	high
Train 8	car	red	small

Validation set

ID	vehicle	color	acceleration
Valid 1	motorbike	red	small
Valid 2	motorbike	blue	high
Valid 3	car	blue	high
Valid 4	car	blue	high

Test set

ID	vehicle	color	acceleration
Teszt 1	motor	piros	small
Teszt 2	motor	zöld	small
Teszt 3	autó	piros	small
Teszt 4	autó	zöld	small

Start with a two-level tree
Prune the tree using the validation set
How the decision affect the performance on the test set?

C4.5 with weighted error (cost)

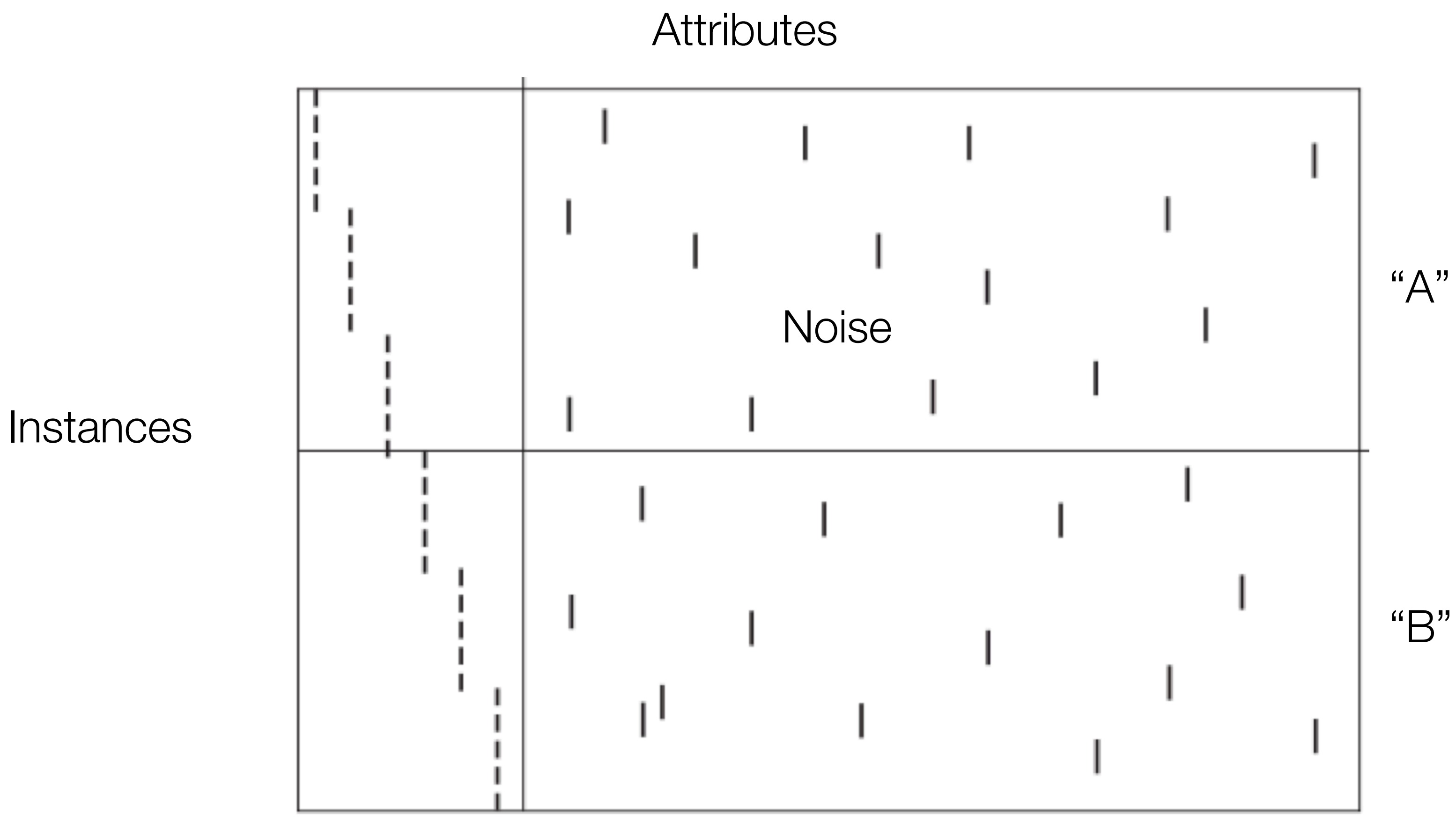
A	B	C	"+"	"-"
I	I	I	5	0
H	I	I	0	20
I	H	I	20	0
H	H	I	0	5
I	I	H	0	0
H	I	H	25	0
I	H	H	0	0
H	H	H	0	25

Start with a two level tree
Is there an ideal two level tree?
Change our decision at the leafs according to the following cost matrix:

Predicted/GT	"+"	"-"
"+"	0	1
"-"	2	0

Noisy attributes

How will perform the kNN and DT?



Missing values

```
?,?,?,?,?,no,auto
xstab,?,?,?,?,yes,noauto
stab,LX,?,?,?,yes,noauto
stab,XL,?,?,?,yes,noauto
stab,MM,nn,tail,?,yes,noauto
?,?,?,?,OutOfRange,yes,noauto
stab,SS,?,?,Low,yes,auto
stab,SS,?,?,Medium,yes,auto
stab,SS,?,?,Strong,yes,auto
stab,MM,pp,head,Low,yes,auto
stab,MM,pp,head,Medium,yes,aut
o
stab,MM,pp,tail,Low,yes,auto
stab,MM,pp,tail,Medium,yes,auto
stab,MM,pp,head,Strong,yes,noaut
o
stab,MM,pp,tail,Strong,yes,auto
```

Shuttle-landing-control

iPython notebook

Anaconda:

```
wget "http://repo.continuum.io/archive/Anaconda3-4.0.0-
Linux-x86_64.sh"
chmod +x Anaconda3-4.0.0-Linux-x86_64.sh
./Anaconda3-4.0.0-Linux-x86_64.sh
source .bashrc
conda update conda
conda update anaconda

conda create -n jupyter-env python=3.5 anaconda
source activate jupyter-env

pip install <module_name>
```

Install packages:

```
pip install pandas
pip install chainer
```

iPython notebook

```
jupyter notebook --generate-config
mcedit .jupyter/jupyter_notebook_config.py
c.NotebookApp.port = 9992
```

If we will work on the server (I hope next week)

Port forward:

```
ssh –L 8888:Localhost:9992
<account>student.ilab.sztaki.hu
```

Final step:

Open in any browser localhost:8888.

Please bring your laptops Friday ☺️

iPython notebook

Small example:

```
import numpy as np
import pandas as pd

v = np.random.random((3))
m = np.random.random((2,3))

v.dot(m.T) # why not v*m?
```

Notes:

```
    - pd.read_csv()
    - dataframe index és values
    - for i in range(10):
        <work>
    - np.linalg.norm(v1-v2) -> L2 distance
    - np.argmax()
```

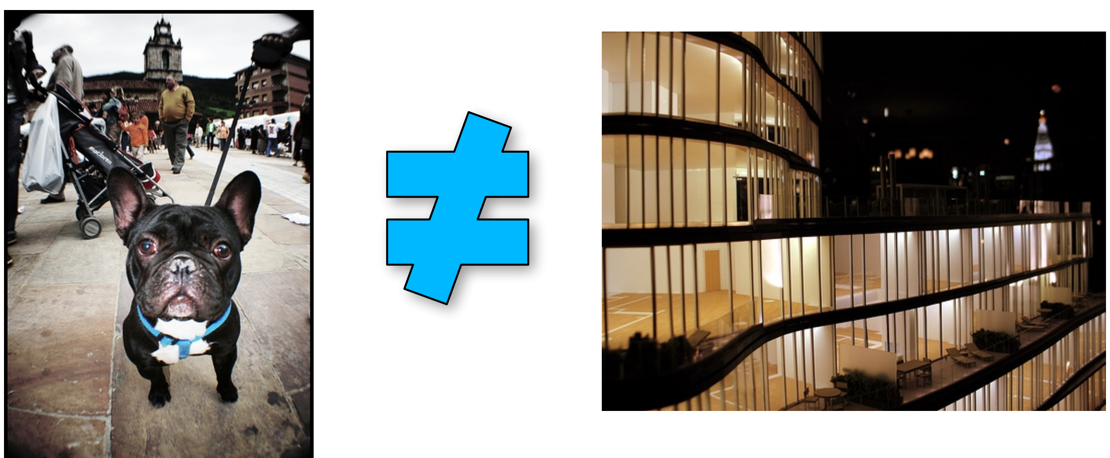
Nearest neighbour

On the web site: NN_data/

image_histograms.txt and sample_histogram.txt:

Input: image histograms 3x8 RGB

Goal: find the closest image to sample image



iPython notebook

```
# Read data

hist = pd.read_csv(’NN_data/image_histograms.txt',sep=' ')
act = pd.read_csv('NN_data/sample_histogram.txt',sep=' ')
```

iPython notebook

```
# Read data

hist = pd.read_csv('NN_data/image_histograms.txt',sep=' ')
act = pd.read_csv('NN_data/sample_histogram.txt',sep=' ')

# distances-> numpy array

dist = np.zeros((len(hist.index)))
dist_norm = np.zeros((len(hist.index)))
```

iPython notebook

```
# Read data

hist = pd.read_csv('NN_data/image_histograms.txt',sep=' ')
act = pd.read_csv('NN_data/sample_histogram.txt',sep=' ')

# distances-> numpy array

dist = np.zeros((len(hist.index)))
dist_norm = np.zeros((len(hist.index)))

# pandas dataframe -> numpy array

hist_vecs = np.array(hist.values[:,1:]).astype(np.float32)
hist_vecs_norm = np.copy(hist_vecs).astype(np.float32)
```

iPython notebook

```
# Read data

img_hists = pd.read_csv('NN_data/image_histograms.txt',sep=' ')
act_hist = pd.read_csv('NN_data/sample_histogram.txt',sep=' ')

# distances -> numpy array

dist = np.zeros((len(hist.index)))
dist_norm = np.zeros((len(hist.index)))

# pandas dataframe -> numpy array

hist_vecs = np.array(hist.values[:,1:]).astype(np.float32)
hist_vecs_norm = np.copy(hist_vecs).astype(np.float32)

# normalization (L2)

act_vec = np.array(act.values[:,1:]).astype(np.float32)
act_vec_norm = act_vec/np.linalg.norm(act_vec).astype(np.float32)
for i in range(hist_vecs[:,0].size):
    norm= np.linalg.norm(hist_vecs[i])
    hist_vecs_norm[i] = hist_vecs[i]/norm

# Norm vs. distance?
```

iPython notebook

```
# compute distances

for i in range(hist_vecs[:,0].size):
    dist[i] = np.linalg.norm(hist_vecs[i]-act_vec)
    dist_norm[i] = np.linalg.norm(hist_vecs_norm[i]-act_vec_norm)
```

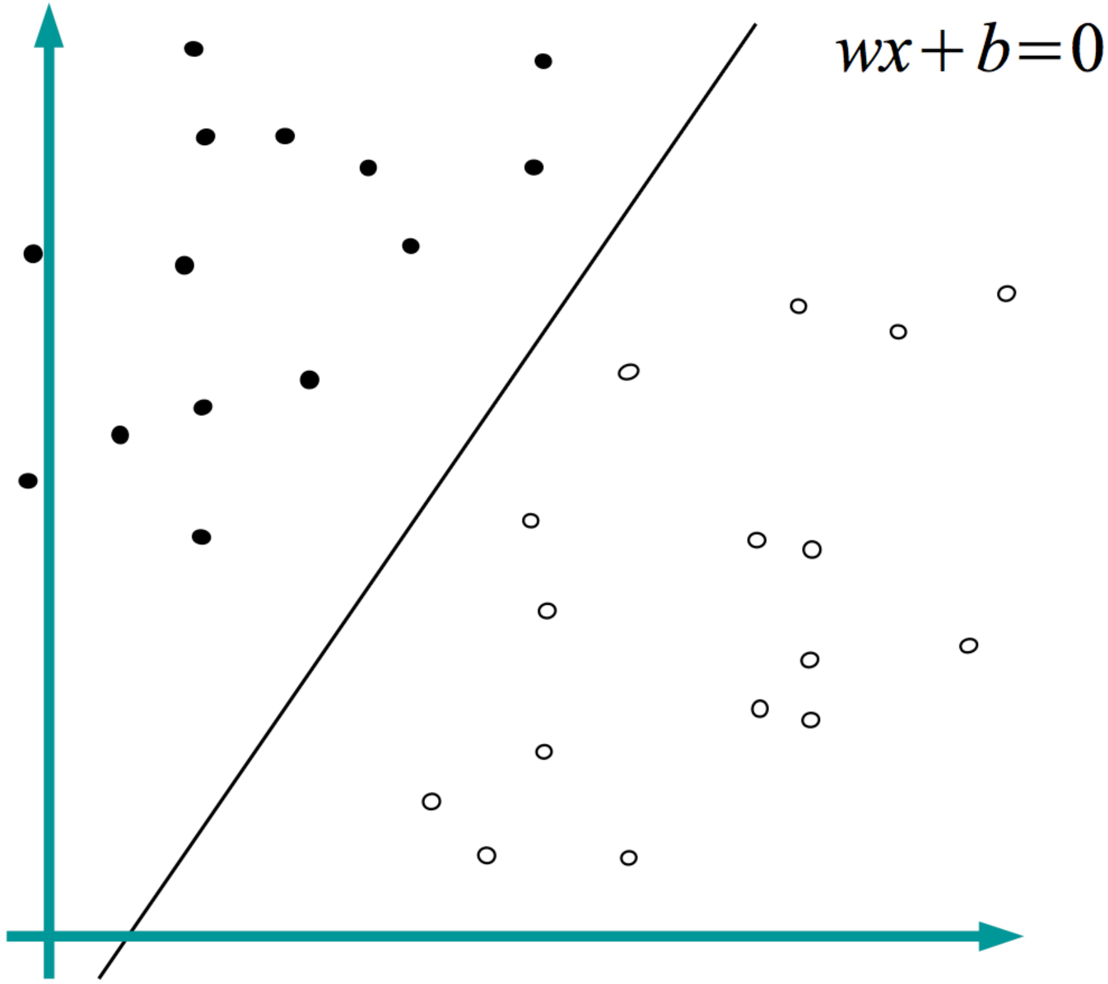
iPython notebook

```
# compute distances

for i in range(hist_vecs[:,0].size):
    dist[i] = np.linalg.norm(hist_vecs[i]-act_vec)
    dist_norm[i] = np.linalg.norm(hist_vecs_norm[i]-act_vec_norm)

# min, max

top = np.argmin(dist)
top_val = np.min(dist)
top_norm = np.argmin(dist_norm)
top_norm_val = np.min(dist_norm)
```

iPython notebook

```
# compute distances

for i in range(hist_vecs[:,0].size):
    dist[i] = np.linalg.norm(hist_vecs[i]-act_vec)
    dist_norm[i] = np.linalg.norm(hist_vecs_norm[i]-act_vec_norm)

# min, max

top = np.argmin(dist)
top_val = np.min(dist)
top_norm = np.argmin(dist_norm)
top_norm_val = np.min(dist_norm)

# evaluation

print('before normalization: %s,%s %f' % (act.values[0,0], hist.values[top,0], 
top_val))
print('after normalization: %s,%s %f' % (act.values[0,0], hist.values[top_norm,0], 
top_norm_val))
```

Linear separator



Linear separator

The problem of learning a half-space or a linear separator consists of n labeled examples $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ in d-dimensional space. The task is to find a d-dimensional vector **w**, if one exists, and a threshold b such that

$$\mathbf{w}\cdot\mathbf{a}_i > b \text{ for each } a_i \text{ labelled} +1$$
$$\mathbf{w}\cdot\mathbf{a}_i < b \text{ for each } a_i \text{ labelled } -1$$

A vector-threshold pair, (**w**, b), satisfying the inequalities is called a linear separator.

Linear separator

If we add an extra dimension to each sample and our norm vector we can rewritten the above formula as

$$(\mathbf{w}' \cdot \mathbf{a}'_i)\, l_i > 0$$

where $1 \leq i \leq n$ and $\mathbf{a}'_l = (\mathbf{a}_i, 1)$, $\mathbf{w}' = (\mathbf{w}, b)$.

Perceptron learning

Let $\mathbf{w} = l_1\mathbf{a}_1$ and $|\mathbf{a}_i|=1$ for each $\mathbf{a}_i$

while exists any $\mathbf{a}_i$ with $(w \cdot a_i)l_i \leq 0$

do

$\mathbf{w}^{t+1} = \mathbf{w}^t + l_i\mathbf{a}_i$

If our problem linearly separable, $(\mathbf{w} \cdot \mathbf{a}_i)l_i > 0$ for all i.

Linear regression

Hypothesis:

$$Y = X^T w$$

Cost (or loss, error) function:

$$error_{square}(f) = E[(Y - f(X))^2]$$

But our dataset is finite:

$$error(f) = \sum_1^N (y_i - x_i^T w_i)^2$$

Linear regression

so:

$$error(f) = \sum_{1}^{N} (y_i - x_i^T w_i)^2 = (y - Xw)^T (y - Xw)$$

There exist a minimum

$$-2X^T y + 2X^T Xw = 0 \qquad X^T (y - Xw) = 0$$

And if the determinant is non-zero (non singular):

$$w = (X^T X)^{-1} X^T y$$

Logistic regression

What are the obvious constrains of lin. reg.?

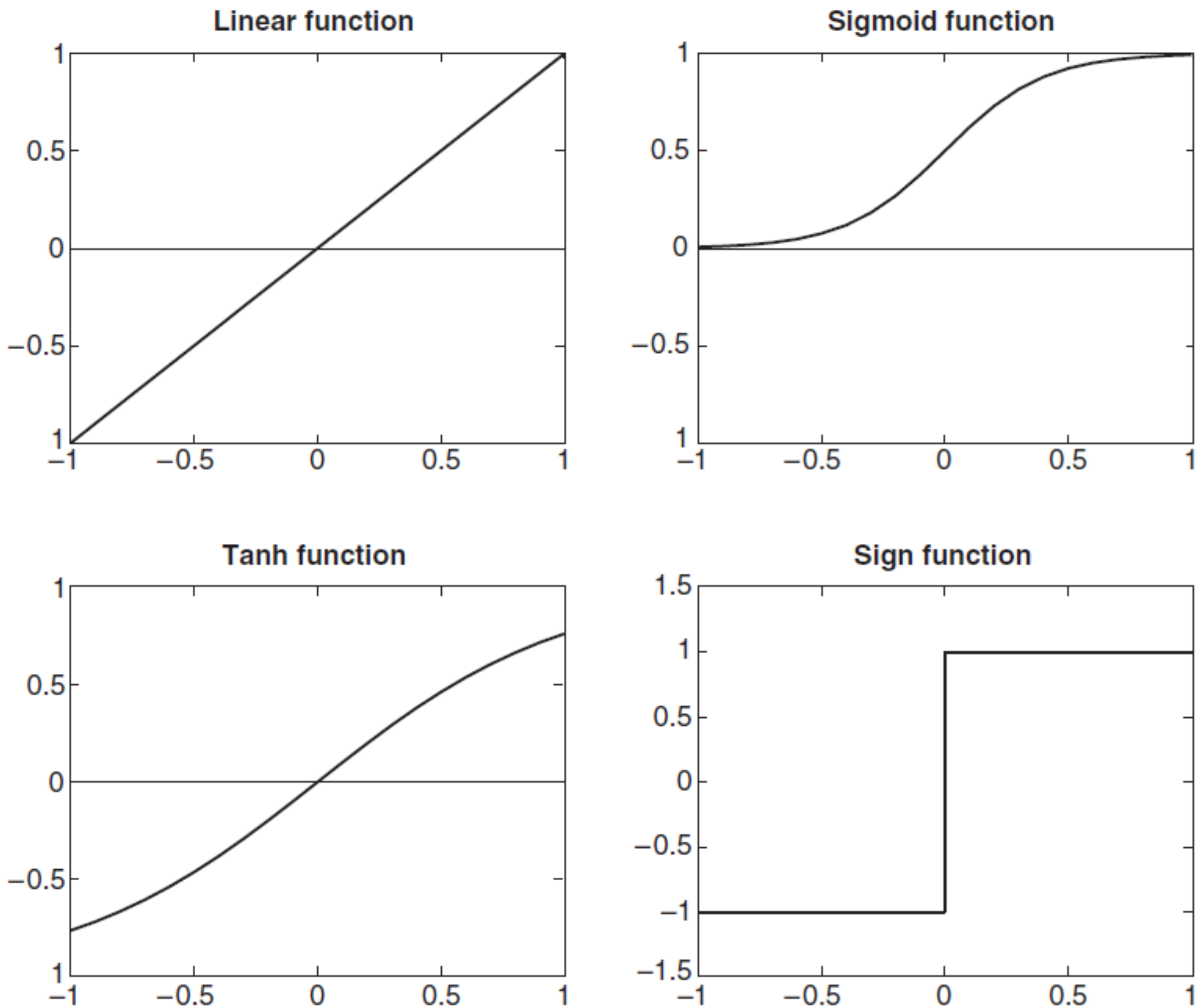
Our decision function was signum.

How about a more refined one:

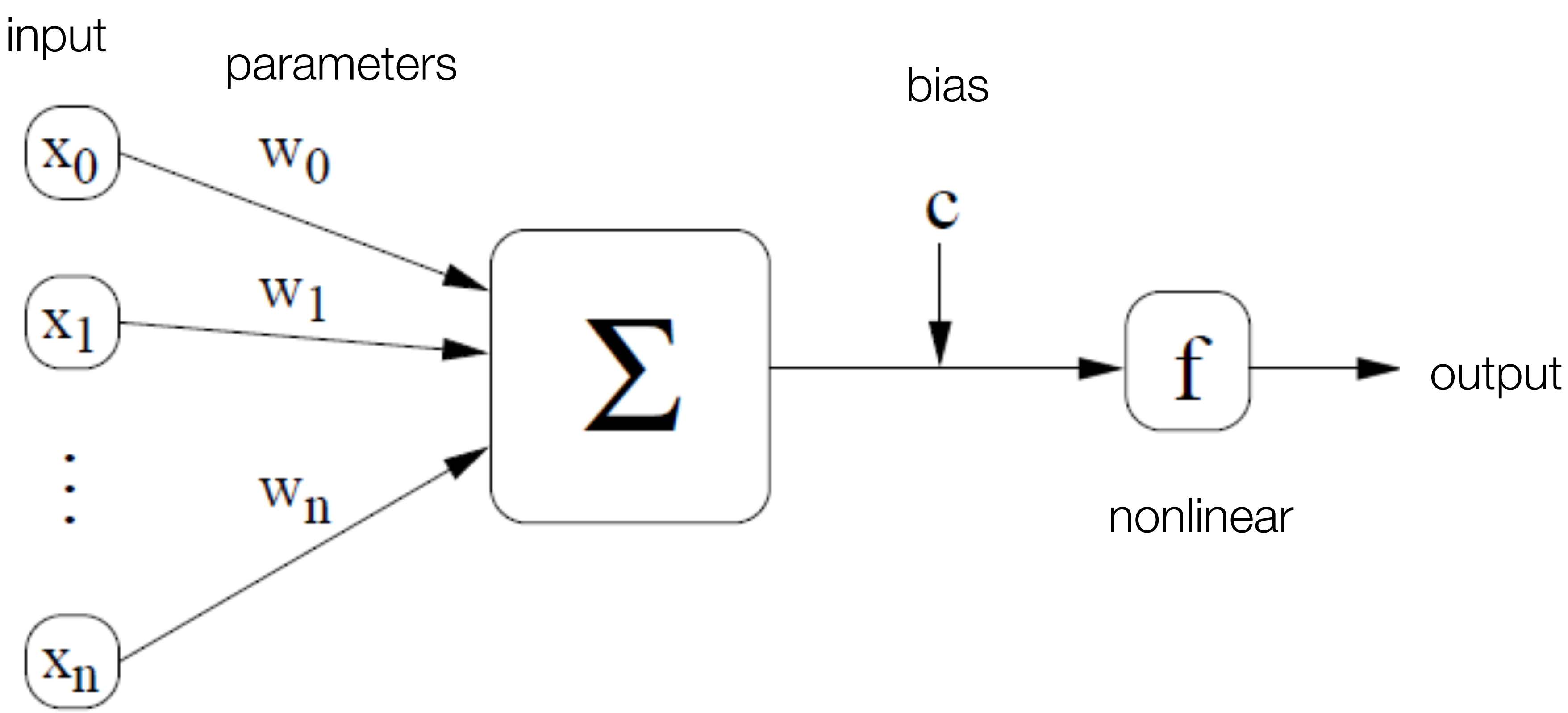
$$y = x^T w - 0.5$$

$$f(y) = y_{osztály} = \frac{1 + sgn(y)}{2}$$

Activation functions



Logistic regression



Logistic regression

Optimization:

$$w_{opt}=\text{argmax}_w \sum \ln(P(y_i|x_i,w))$$

In case of binary classification:

$$L(w)=\sum y_i \ln(P(y_i=1 \,|x_i,w)) + (1- y_i) \ln(P(y_i=0 \,|x_i,w))$$

What is the gradient? Some exercise ☺️

Logistic regression

Optimization:

$$w_{opt}=\text{argmax}_w \sum \ln(P(y_i|x_i,w))$$

In case of binary classification:

$$L(w)=\sum y_i \ln(P(y_i=1 \ |x_i,w)) + (1- y_i) \ln(P(y_i=0 \ |x_i,w))$$

What is the gradient? Some exercise ☺️

$$\sum x_{ij}(y_i - P(y_i|x_{ij},w_j))$$

Logistic regression

Or

$$\ln \frac{p(x)}{1-p(x)} \approx x^T\omega + \omega_0$$

hence
$$p(x \mid \omega) = sigm(x \mid \omega) = \frac{1}{1+\mathrm{e}^{-(x^T\omega+\omega_0)}}$$

The end is the same:

$$\begin{aligned}\frac{\partial \mathcal{L}(\omega \mid X)}{\partial \omega_i} &= \sum_{x_t \in X^{(+)}} \frac{\partial \ln p(x_t \mid \omega)}{\partial \omega_i} + \sum_{x_t \in X^{(-)}} \frac{\partial \ln(1-p(x_t \mid \omega))}{\partial \omega_i} \\ &= \sum_{x_t \in X^{(+)}} (1-p(x_t \mid \omega))x_{ti} - \sum_{x_t \in X^{(-)}} p(x_t \mid \omega)x_{ti} \\ &= \sum_{x_t \in \{X^{(-)}, X^{(+)}\}} (y_t - p(x_t \mid \omega))x_{ti}\end{aligned}$$

Linear separator (recap)

The problem of learning a half-space or a linear separator consists of n labeled examples $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ in d-dimensional space. The task is to find a d-dimensional vector **w**, if one exists, and a threshold b such that

$$\mathbf{w} \cdot \mathbf{a}_i > b \text{ for each } a_i \text{ labelled} +1$$

$$\mathbf{w} \cdot \mathbf{a}_i < b \text{ for each } a_i \text{ labelled } -1$$

A vector-threshold pair, (**w**, b), satisfying the inequalities is called a linear separator.

Linear separator

If we add an extra dimension to each sample and our norm vector we can rewritten the above formula as

$$(\mathbf{w'} \cdot \mathbf{a'}_i)\, l_i > 0$$

where $1 \le i \le n$ and $\mathbf{a'}_I = (\mathbf{a}_i, 1)$, $\mathbf{w'} = (\mathbf{w}, b)$.

Perceptron learning

Let $\mathbf{w} = l_1\mathbf{a}_1$ and $|\mathbf{a}_i|=1$ for each $\mathbf{a}_i$

while exists any $\mathbf{a}_i$ with $(w \cdot a_i)l_i \leq 0$

do

$\mathbf{w}^{t+1} = \mathbf{w}^t + l_i\mathbf{a}_i$

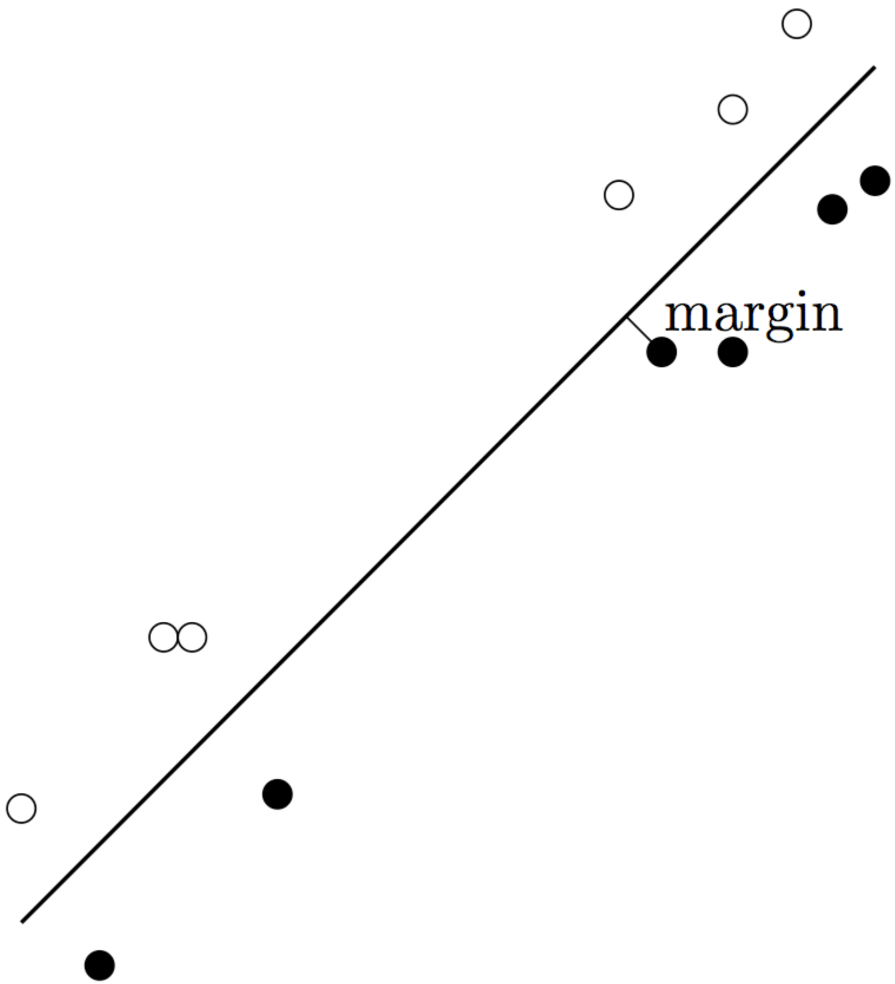
If our problem linearly separable, $(\mathbf{w} \cdot \mathbf{a}_i)l_i > 0$ for all i.

Margin

Definition: For a solution w, where $|\mathbf{a}_i| = 1$ for all examples, the margin is defined to be the minimum distance of the hyperplane $\{\mathbf{x} | \mathbf{w} \cdot \mathbf{x} = 0\}$ to any $\mathbf{a}_i$, namely,

$$\min_i \frac{(\mathbf{w} \cdot \mathbf{a_i}) l_i}{|\mathbf{w}|}$$

Theorem: Suppose there is a solution $\mathbf{w}^*$ with margin $\delta > 0$. Then, the perceptron learning algorithm finds some solution **w** with $(\mathbf{w} \cdot \mathbf{a}_i)\, l_i > 0$ for all i in at most $\frac{1}{\delta^2} - 1$ iterations.


margin

Maximizing the Margin

The margin of a solution $\mathbf{w}$ to $(\mathbf{w}^T\mathbf{a}_i)l_i > 0$, $1 \le l \le n$, where $|\mathbf{a}_i| = 1$ is. By modifying the weight vector, we can convert the optimization problem to one with a concave objective function:

$$\delta = \min_i \frac{l_i(\mathbf{w}^T\mathbf{a_i})}{|\mathbf{w}|}$$

$$l_i\left(\frac{\mathbf{w}^T\mathbf{a_i}}{|\mathbf{w}|\delta}\right) > 1$$

for all $\mathbf{a}_i$. Our modified model is

$$\mathbf{v} = \frac{\mathbf{w}}{\delta|\mathbf{w}|}$$

Maximizing δ is equivalent to minimizing $|\mathbf{v}|$!

Maximizing the Margin

Our (almost) final optimization problem is

minimize $|\mathbf{v}|$ subject to $l_i(\mathbf{v}^T \mathbf{a}_i) > 1, \forall i.$

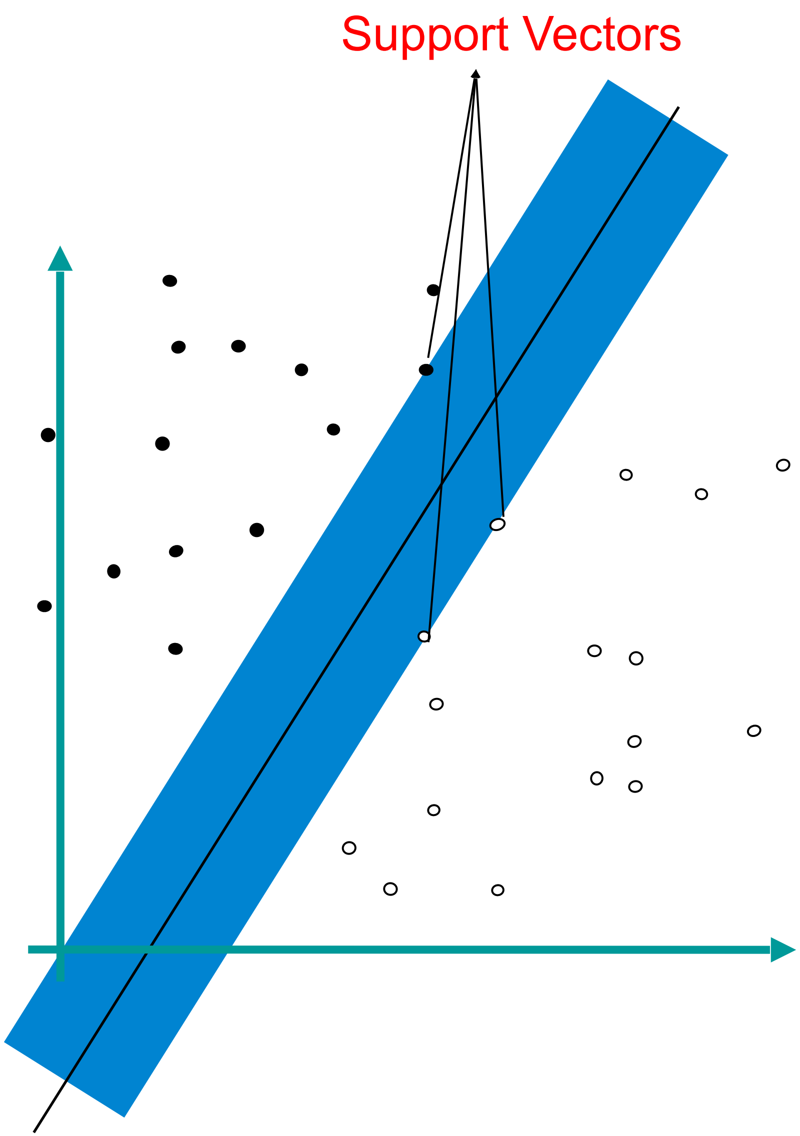
Because of nice properties of $|\mathbf{v}|^2$ we will optimize on that:

minimize $|\mathbf{v}|^2$ subject to $l_i(\mathbf{v}^T \mathbf{a}_i) \geq 1, \forall i.$

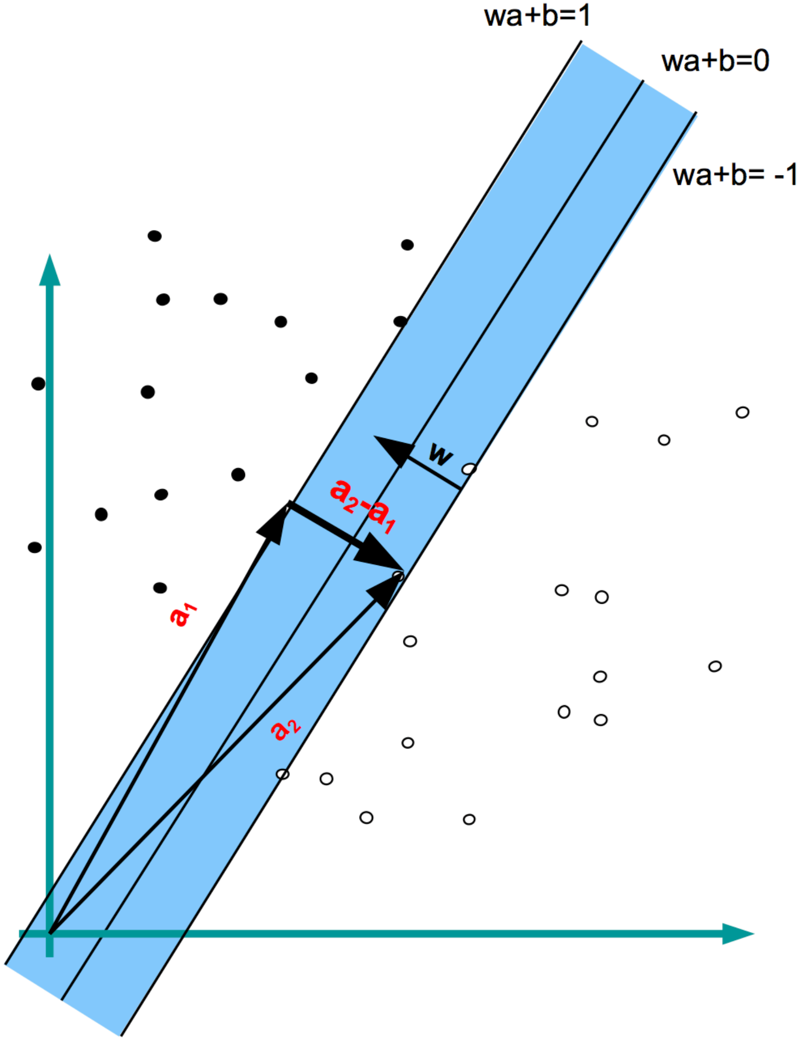
Let V be the space spanned by the examples $\mathbf{a}_i$ for which there is equality, namely for which $l_i(\mathbf{v}^T \mathbf{a}_i) = 1$.

We claim that **v** lies in V . If not, **v** has a component orthogonal to V. Reducing this component infinitesimally does not violate any inequality, but contradicting our optimization.

Maximizing the Margin



Maximizing the Margin



Be a_2 and a_1 two vectors so that a_2-a_1 is parallel to w and

$$wa_1+b=1$$

$$wa_2+b=-1$$

We know:

$$a_2-a_1=-n\frac{w}{\|w\|} \longrightarrow a_2=a_1-n\frac{w}{\|w\|}$$

and

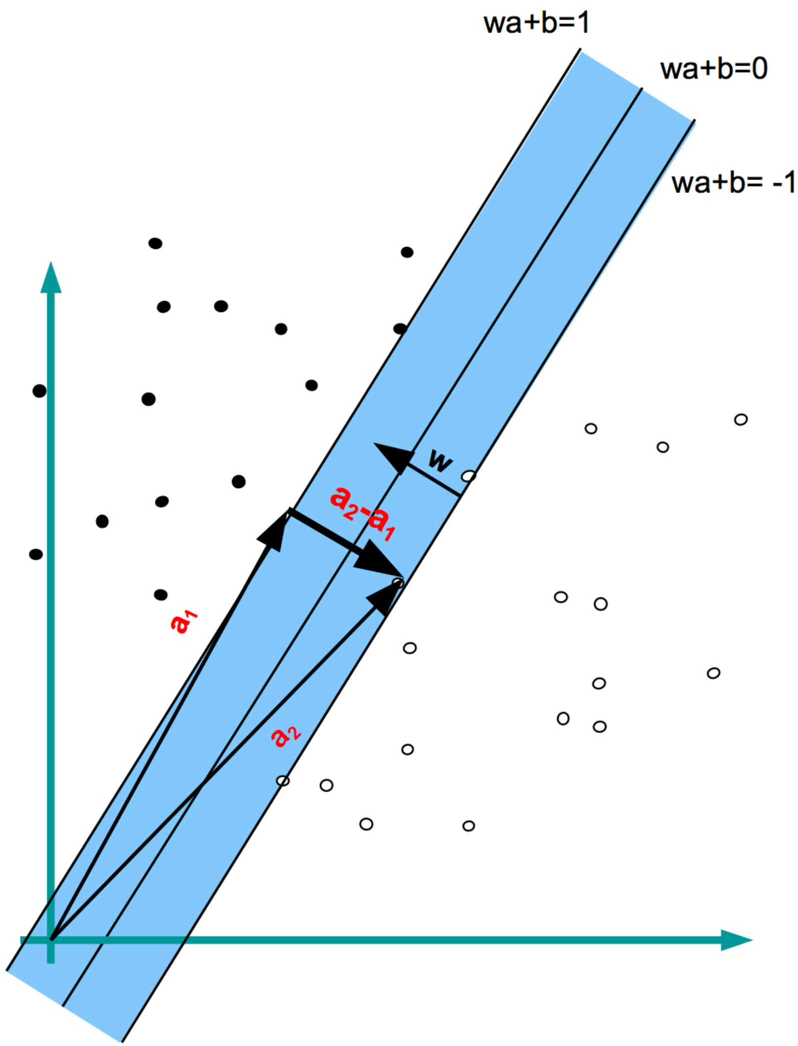
$$wa_2+b=\left(a_1-n\frac{w}{\|w\|}\right)w+b=-1$$

where $$wa_1+b=1$$

$$wa_1+b-n\frac{w}{\|w\|}w=-1$$

$$1-n\frac{w}{\|w\|}w=-1 \longrightarrow n=\frac{2}{\|w\|}$$

Maximizing the Margin



Be a_2 and a_1 two vectors so that a_2-a_1 is parallel to w and

$$wa_1+b=1$$

$$wa_2+b=-1$$

We know:

$$a_2-a_1=-n\frac{w}{\|w\|} \longrightarrow a_2=a_1-n\frac{w}{\|w\|}$$

and

where

$$1-n\frac{w}{\|w\|}w=-1 \longrightarrow n=\frac{2}{\|w\|}$$

Maximize $n=\frac{2}{\|w\|}$

↓

Minimize $\|w\|$

Soft Margin

It may happen that there are linear separators for which almost all but a small fraction of examples are on the correct side.

Our goal is to find a solution **w** for which at least $(1 - \varepsilon)n$ of the n inequalities are satisfied.

Unfortunately, such problems are NP-hard and there are no good algorithms to solve them.

Soft Margin

First idea: Count the number of misclassified points (“loss”). Our goal is to minimize the “loss”.

With a nicer loss function it is possible to solve the problem.

Let us introduce so called slack variables

$$y_i,\ i = 1, 2, \ldots, n$$

where y_i measures how badly the example $\mathbf{a}_i$ is classified.

Soft Margin

Now we can include slack variables in the original objective function:

$$\text{minimize } |\mathbf{v}|^2 + c\sum_{i=1}^{n} y_i$$

$$\left.\begin{array}{ll}\text{subject to} & (\mathbf{v}\cdot\mathbf{a_i})l_i \geq 1 - y_i \\ & y_i \geq 0.\end{array}\right\} \; i = 1, 2, \ldots, n$$

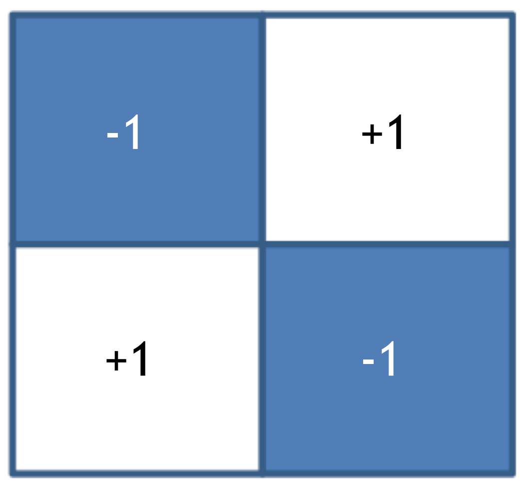
Let y_i be zero, if $\mathbf{a}_i$ classified correctly and 1 - l_i ($\mathbf{v}^T\mathbf{a}_i$) if not ->

$$|\mathbf{v}|^2 + c\sum_{i}(1 - l_i(\mathbf{v}\cdot\mathbf{a_i}))^+$$

where $x^+ = \begin{cases} 0 & x \leq 0 \\ x & \text{otherwise} \end{cases}$

Nonlinear separators

There are problems where no linear separator exists but where there are nonlinear separators. For example, there may be a polynomial p(·) such that $p(\mathbf{a}_i) > 1$ for all +1 labeled examples and $p(\mathbf{a}_i) < 1$ for all -1 labeled examples.



Solution: $p(\cdot) = x_1 \, x_2$

Polynomial separator

Assume:

There exist a polynomial p of degree at most D such that an example **a** has label +1 if and only if p(a) > 0

Each d-tuple of integers $(i_1, i_2, \ldots, i_d)$

$$i_1 + i_2 + \cdots + i_d \leq D$$

leads to a distinct monomial:

$$x_1^{i_1} x_2^{i_2} \cdots x_d^{i_d}.$$

Polynomial separator

By letting the coefficients of the monomials be unknowns, we can formulate a linear program in m variables whose solution gives the required polynomial

$$p(x_1, x_2, \ldots, x_d) = \sum_{\substack{i_1, i_2, \ldots, i_d \\ i_1 + i_2 + \cdots + i_d \leq D}} w_{i_1, i_2, \ldots, i_d} x_1^{i_1} x_2^{i_2} \cdots x_d^{i_d}$$

For even small values of D the number of coefficients can be very large!

Polynomial separator

An example: suppose both d and D equal to 2.
The number of possible monomials is 6,

I_1, i_2 form a set {(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)}

The (0,0) term is the bias (b), our polynomial has a form,

$$p(x_1, x_2) = b + w_{10}x_1 + w_{01}x_2 + w_{11}x_1x_2 + w_{20}x_1^2 + w_{02}x_2^2$$

For each example $\mathbf{a}_i$

$$b + w_{10}a_{i1} + w_{01}a_{i2} + w_{11}a_{i1}a_{i2} + w_{20}a_{i1}^2 + w_{02}a_{i2}^2 > 0 \text{ if label of } i = +1$$

$$b + w_{10}a_{i1} + w_{01}a_{i2} + w_{11}a_{i1}a_{i2} + w_{20}a_{i1}^2 + w_{02}a_{i2}^2 < 0 \text{ if label of } i = -1$$

Polynomial separator

The approach above can be thought of as embedding the examples $\mathbf{a}_i$ that are in d-space into a m-dimensional space:

each $i_1, i_2, \ldots, i_d$ summing to at most D, and if

$$\varphi(\mathbf{x}): R^d \text{ -> } R^m$$

$$\mathbf{a}_i = (x_1, x_2, \ldots, x_d) \quad \text{->} \quad x_1^{i_1} x_2^{i_2} \cdots x_d^{i_d}$$

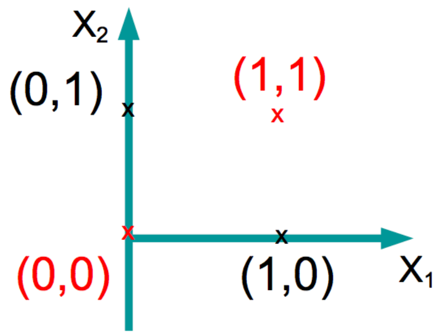
If d=2 and D=2: $\varphi(\mathbf{x}) = (x_1, x_2, x_1^2, x_1 x_2, x_2^2)$

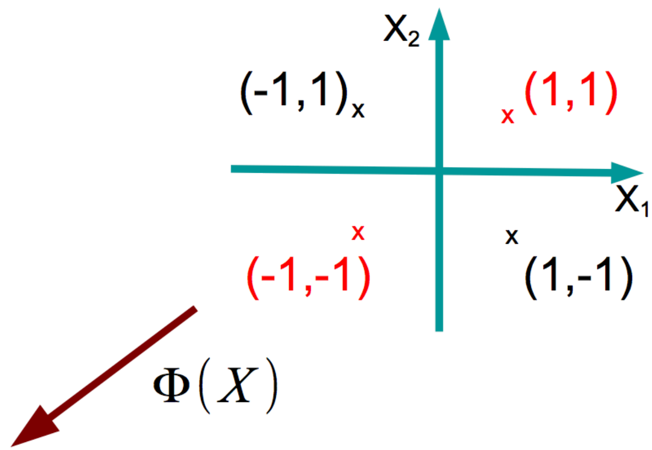
If d=3 and D=2:

$$\varphi(\mathbf{x}) = (x_1, x_2, x_3, x_1 x_2, x_1 x_3, x_2 x_3, x_1^2, x_2^2, x_3^2)$$

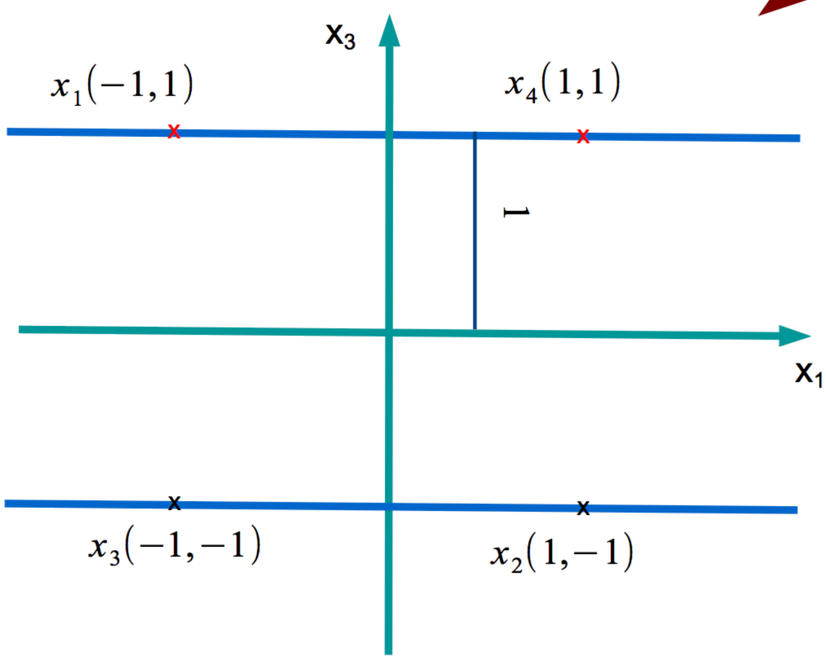
Polynomial separator

Original feature space





After the transformation



$x_{11}=-1\,,\,x_{13}=1$ $\qquad y_1=-1$

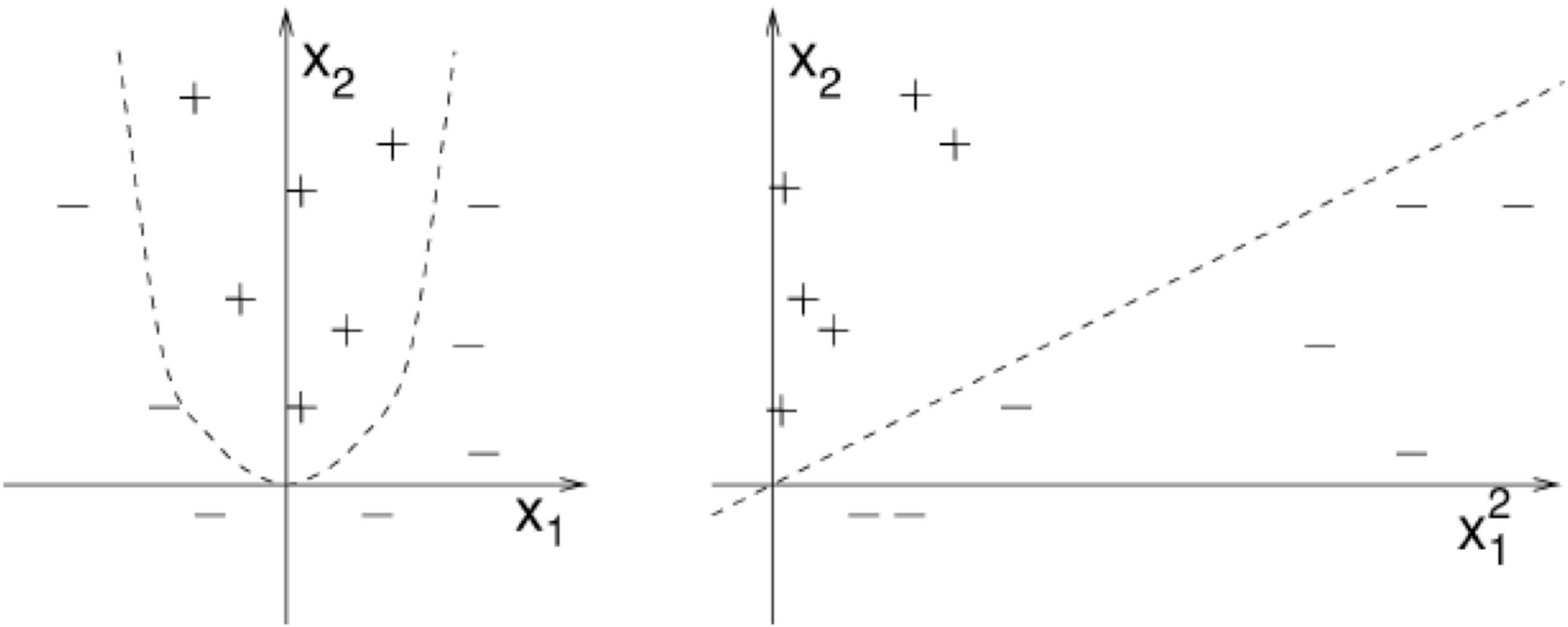
$x_{21}=1\,,\,x_{23}=-1$ $\qquad y_2=+1$

$x_{31}=-1\,,\,x_{33}=-1$ $\qquad y_3=+1$

$x_{41}=1\,,\,x_{43}=1$ $\qquad y_4=-1$

Polynomial separator

$$\Phi(x)=(x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{(2)}x_1 x_2, 1)$$



Polynomial separator

We can use the previously defined objective function to find the coefficients

$$\min |\mathbf{w}|^2 \text{ subject to } (\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{a_i}))l_i \geq 1 \text{ for all } i$$

But how to avoid computing the transformed vectors?

Lemma: Any optimal solution w to the convex program above is a linear combination of the $\varphi(\mathbf{a_i})$

So $\mathbf{w} = \sum_i y_i \boldsymbol{\varphi}(\mathbf{a_i})$ and $\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{a_j})$ can be computed without actually knowing the transformed vectors.

Polynomial separator

Say, $\mathbf{w} = \sum_i y_i \boldsymbol{\varphi}(\mathbf{a_i})$

where the y_i are real variables.

Then

$$|\mathbf{w}|^2 = \left(\sum_i y_i \boldsymbol{\varphi}(\mathbf{a_i})\right)^T \left(\sum_j y_j \boldsymbol{\varphi}(\mathbf{a_j})\right) = \sum_{i,j} y_i y_j \boldsymbol{\varphi}(\mathbf{a_i})^T \boldsymbol{\varphi}(\mathbf{a_j})$$

And our optimization has a form

$$\text{minimize } \sum_{i,j} y_i y_j \boldsymbol{\varphi}(\mathbf{a_i})^T \boldsymbol{\varphi}(\mathbf{a_j})$$

$$\text{subject to } l_i \left(\sum_j y_j \boldsymbol{\varphi}(\mathbf{a_j})^T \boldsymbol{\varphi}(\mathbf{a_i})\right) \geq 1 \quad \forall i.$$

Kernel matrix

In the above formulation we do not need the transformed vectors, only the dot product for all i,j pairs.

Let us define the kernel matrix as

$$k_{ij} = \boldsymbol{\varphi}(\mathbf{a_i})^T \boldsymbol{\varphi}(\mathbf{a_j})$$

So we can rewrite once again our optimization as

$$\text{minimize} \sum_{ij} y_i y_j k_{ij} \qquad \text{subject to} \qquad l_i \sum_j k_{ij} y_j \geq 1$$

This formulation is called as Support Vector Machine (SVM) Instead of m parameters we have n^2 entries.