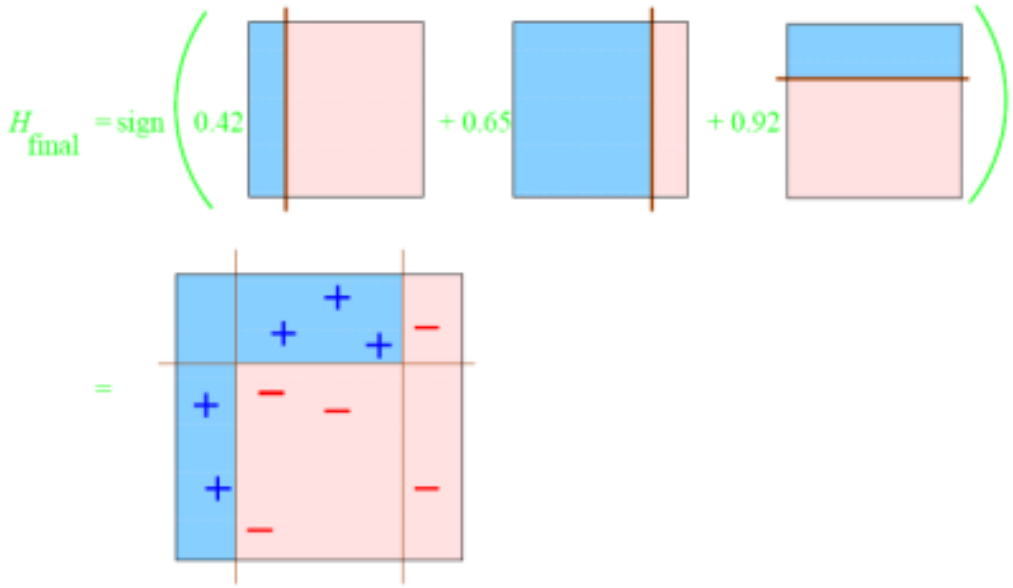
Introduction to Supervised Learning



Week 4: SVM wrap-up, Ensemble Methods, Adaboost

Iasonas Kokkinos

i.kokkinos@cs.ucl.ac.uk

University College London

Course Logistics

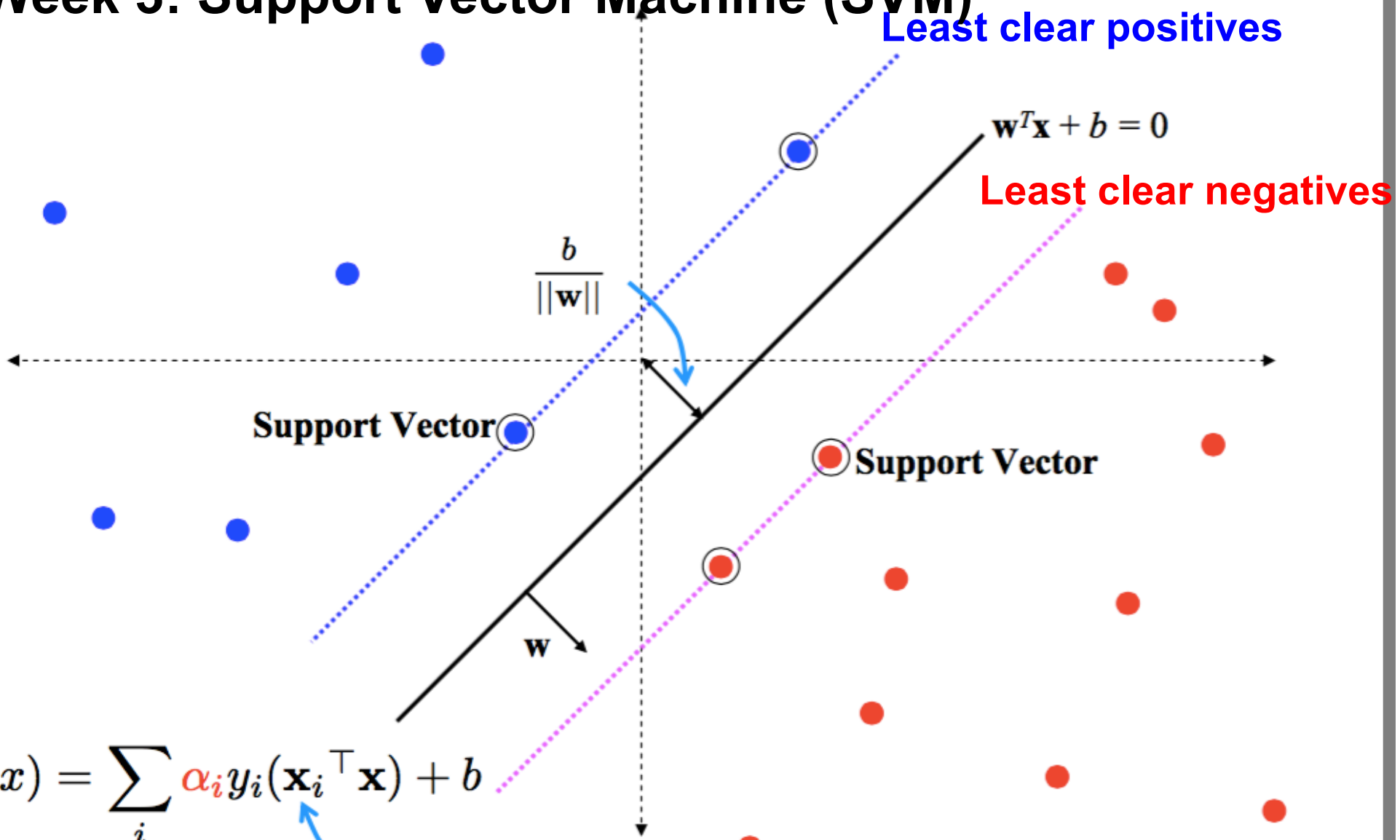
1st Deliverable: end of reading week (10 November), following informal poll during Monday's lab

4 first assignments in one deliverable

40% of total lab assignment grades

100% of lab assignment grades = 30% of your final grade

Week 3: Support Vector Machine (SVM)



$$f(x) = \sum_i \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

support vectors

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha^i \left(y^i \mathbf{x}^i\right)$$

Solution depends only on 'Support Vectors'

Primal and dual problems

Primal, in terms of **w:** $\min_{\mathbf{w}} \|\mathbf{w}\|^2$

$$\text{s.t.}: \quad y^i(\mathbf{w}^T\mathbf{x}^i + b) \geq 1, \quad \forall i$$

But: $\|\mathbf{w}^*\|^2 = \langle \mathbf{w}^*, \mathbf{w}^* \rangle$ where $\mathbf{w}^* = \sum_{i=1}^{N} \alpha^i \left(y^i \mathbf{x}^i\right)$

$$= \left\langle \sum_{i=1}^{N} \alpha^i y^i \mathbf{x}^i, \sum_{j=1}^{N} \alpha^j y^j \mathbf{x}^j \right\rangle = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha^i \alpha^j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j \rangle$$

Dual, in terms of $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$: $\min_{\boldsymbol{\alpha}} \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha^i \alpha^j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j \rangle$

$$\text{s.t.}: \quad y^i \left(\sum_{j=1}^{N} \alpha^j y^j \langle \mathbf{x}^j, \mathbf{x}^i \rangle + b \right) \geq 1, \quad i = 1, \ldots, N$$

Dual form of SVM & kernel trick

Optimization:

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha^i \alpha^j y^i y^j \langle \mathbf{x}^i, \mathbf{x}^j \rangle$$

$$\text{s.t.}: \quad y^i \left(\sum_{j=1}^{N} \alpha^j y^j \langle \mathbf{x}^j, \mathbf{x}^i \rangle + b \right) \geq 1, \quad \forall i$$

$$\boldsymbol{\alpha} \in \mathbb{R}^N \rightarrow O(N^3)$$

Primal and dual classifier forms:

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \sum_{i=1}^{N} \alpha^i y^i \langle \mathbf{x}^i, \mathbf{x} \rangle + b$$

What if we replace **x** with **φ(x)?**

Everything involves only inner products!

Rewrite everything in terms of Kernel

$$K(\mathbf{x}, \mathbf{y}) = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{y}) \rangle$$

Dual form of SVM & kernel trick

Optimization: $$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha^i \alpha^j y^i y^j \; K\left(\mathbf{x}^i, \mathbf{x}^j\right)$$

$$\text{s.t.}: \; y^i \left(\sum_{j=1}^{N} \alpha^j y^j K(\mathbf{x}^j, \mathbf{x}^i) + b \right) \geq 1, \quad i = 1, \ldots, N$$

Dual classifier form:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha^i y^i K(\mathbf{x}^i, \mathbf{x}) + b$$

$$= \sum_{\{i:\alpha^i \neq 0\}} w^i K(\mathbf{x}^i, \mathbf{x}) + b, \quad w^i = y^i \alpha^i$$

Compare with general nonlinear form: $$f(\mathbf{x}) = \sum_{k} w_k \phi_k(\mathbf{x})$$

N nonlinear functions – smart choice of sparse coefficients

Mercer Kernel Examples

Linear kernel

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y}$$

Polynomial kernel

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^p$$

Radial Basis Function (a.k.a. Gaussian) kernel

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{y}\|^2\right)$$

RBF kernel SVM

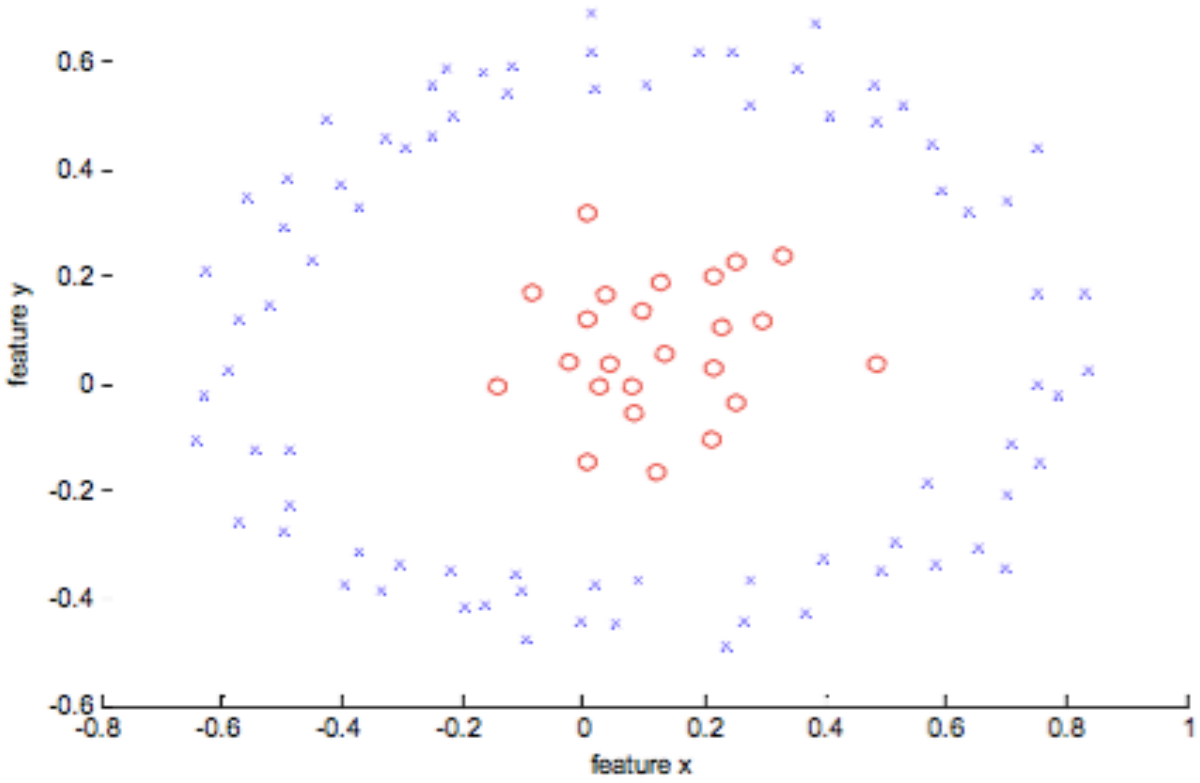
$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha^i y^i K(\mathbf{x}^i, \mathbf{x}) + b$$

$$= \sum_{\{i:\alpha^i \neq 0\}} \alpha^i y^i K(\mathbf{x}^i, \mathbf{x}) + b$$

$$= \sum_{\{i:\alpha^i \neq 0\}} w^i \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}^i - \mathbf{x}\|_2^2\right) + b$$

Discriminant form: sum of bumps centered on training points

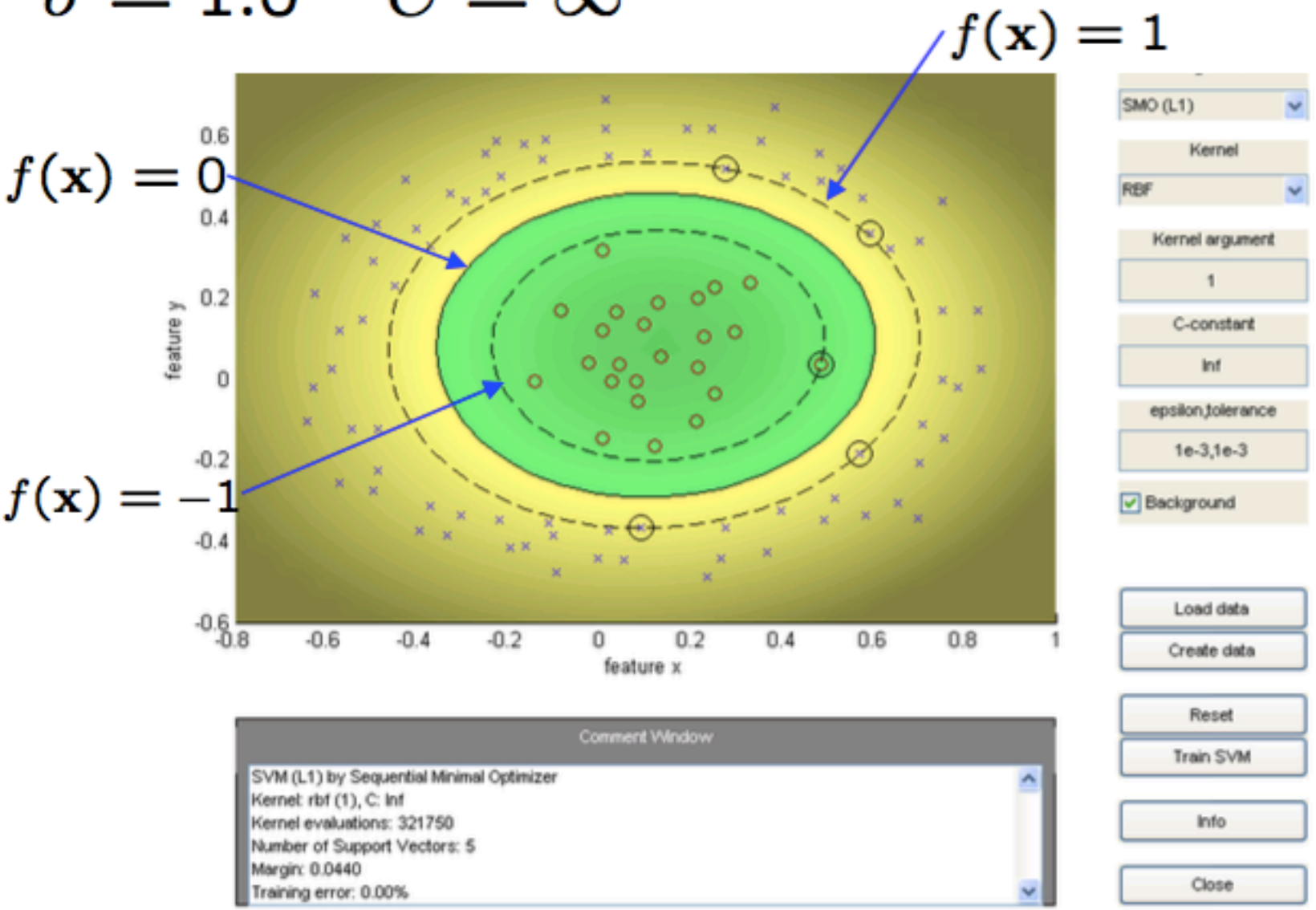
RBF-SVM example



- data is not linearly separable in original feature space

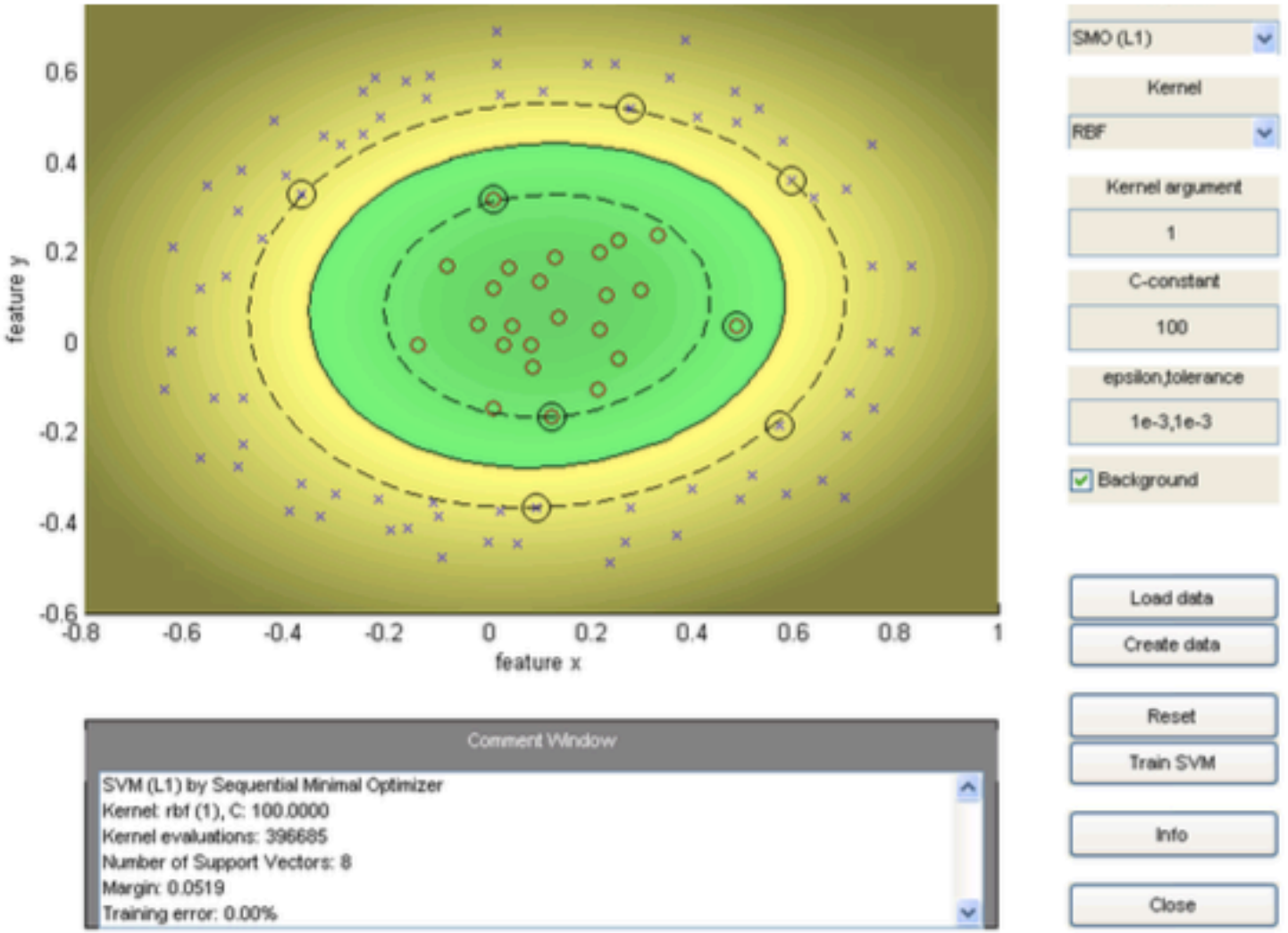
RBF-SVM example

$\sigma = 1.0 \quad C = \infty$



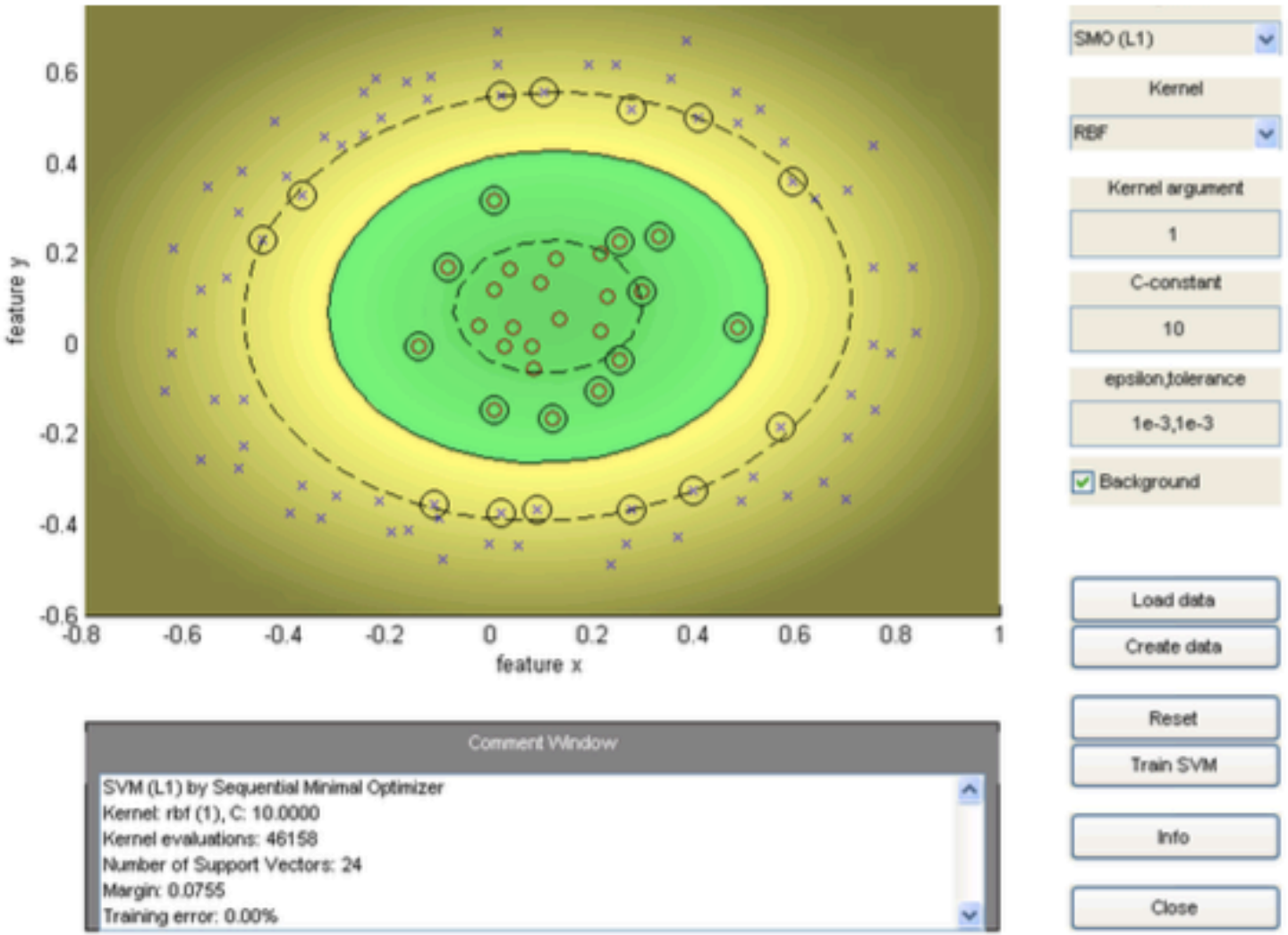
RBF-SVM example

$$\sigma = 1.0 \quad C = 100$$



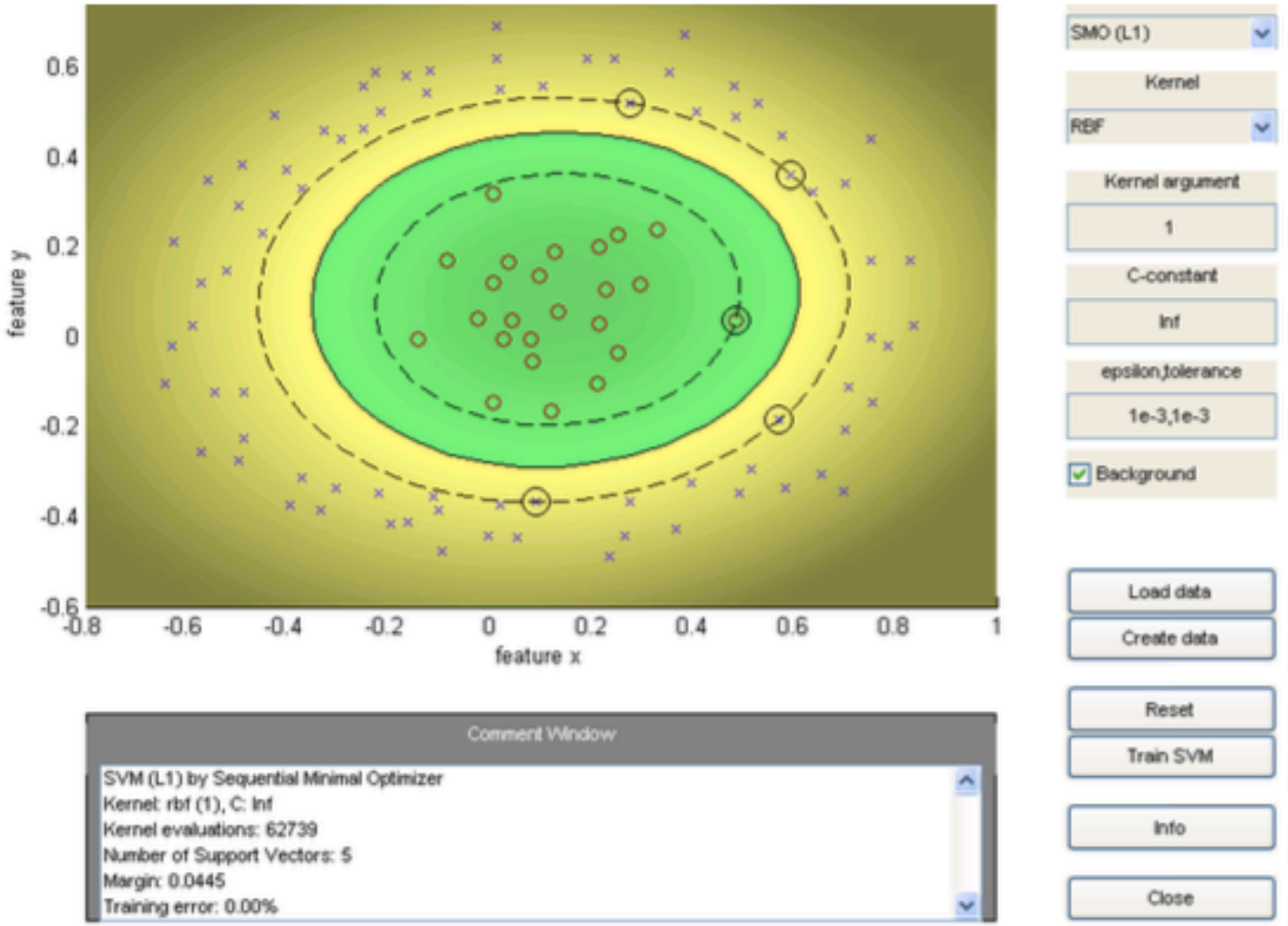
RBF-SVM example

$$\sigma = 1.0 \quad C = 10$$



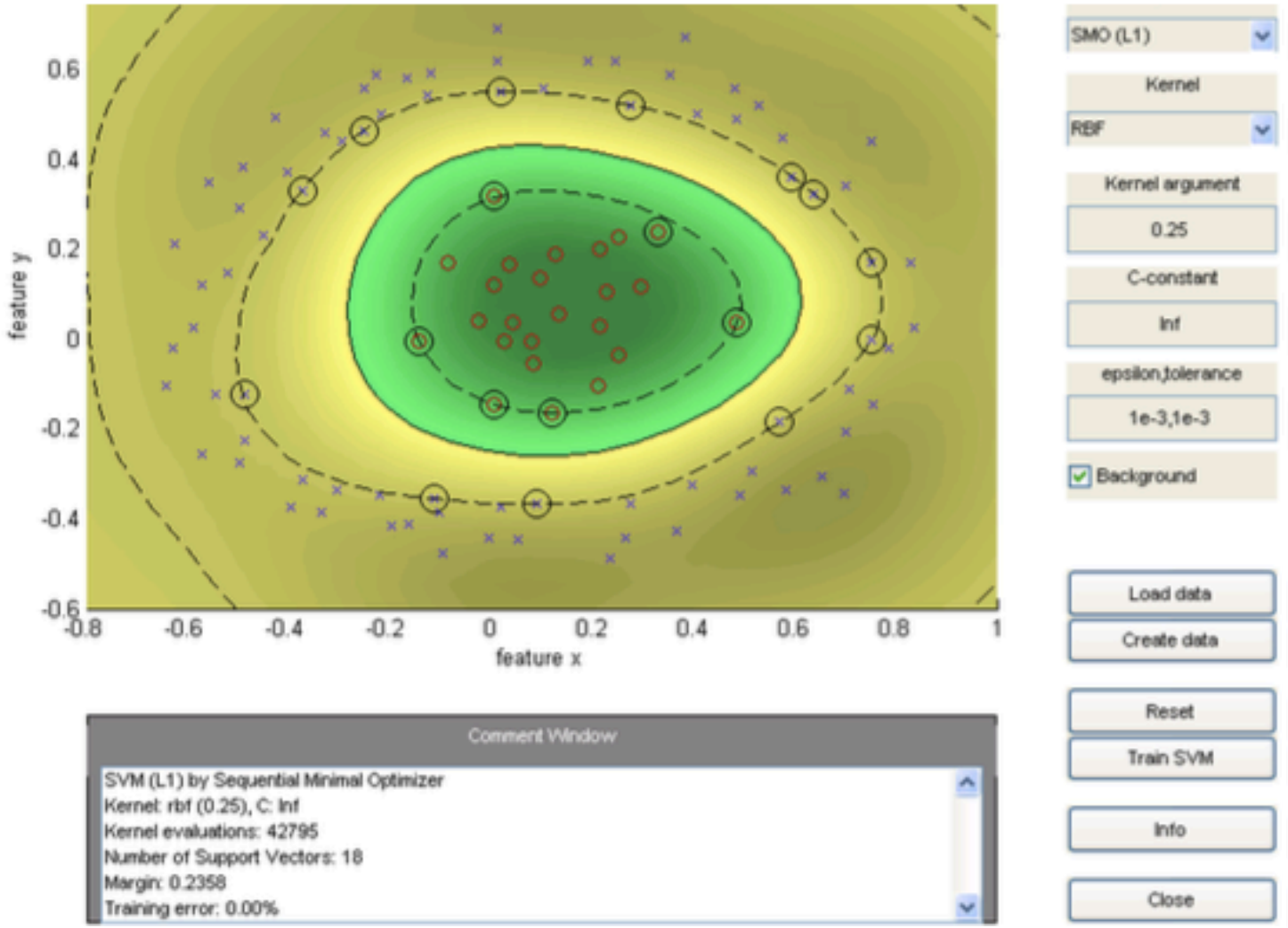
RBF-SVM example

$$\sigma = 1.0 \quad C = \infty$$



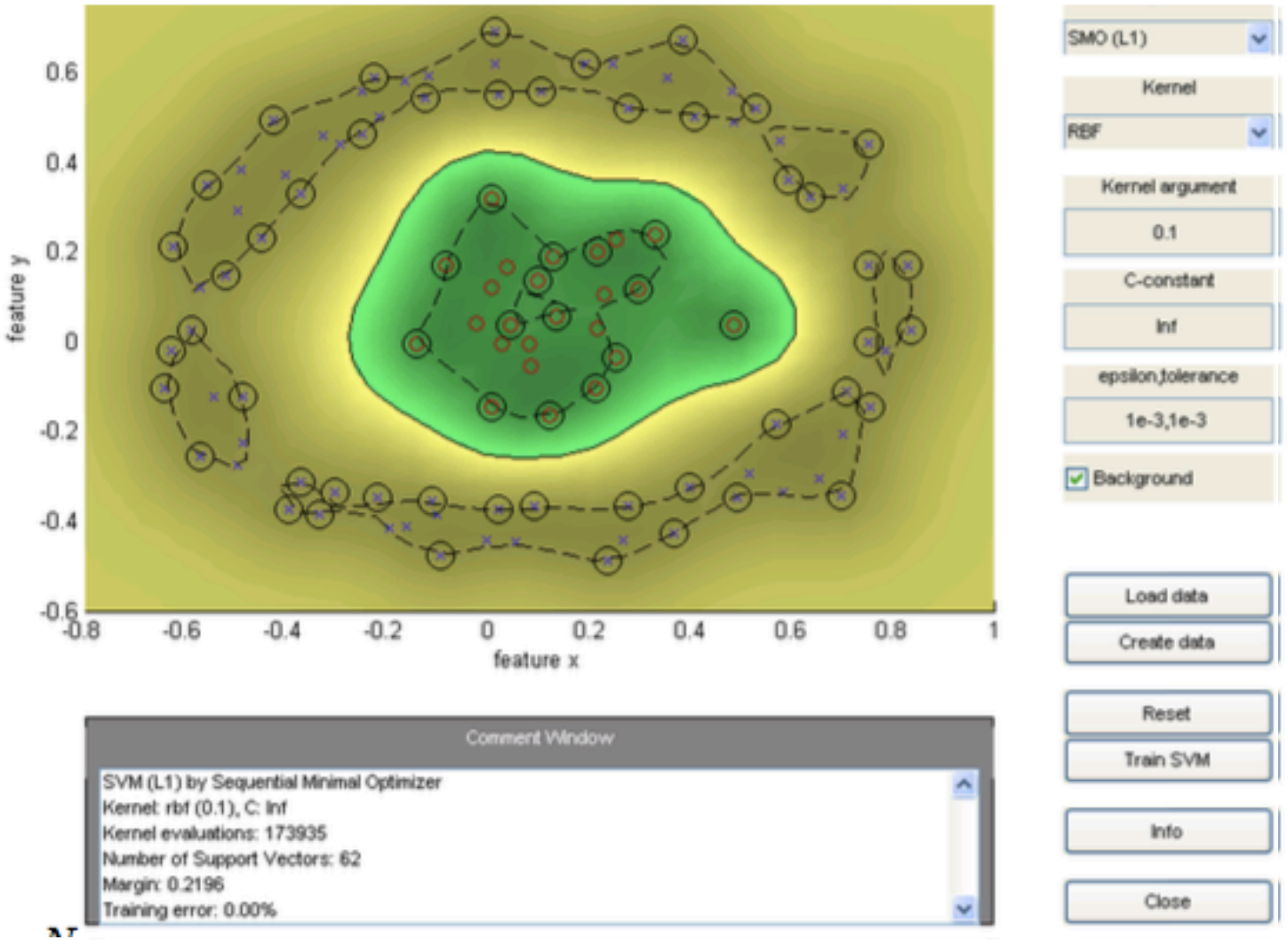
RBF-SVM example

$$\sigma = 0.25 \quad C = \infty$$

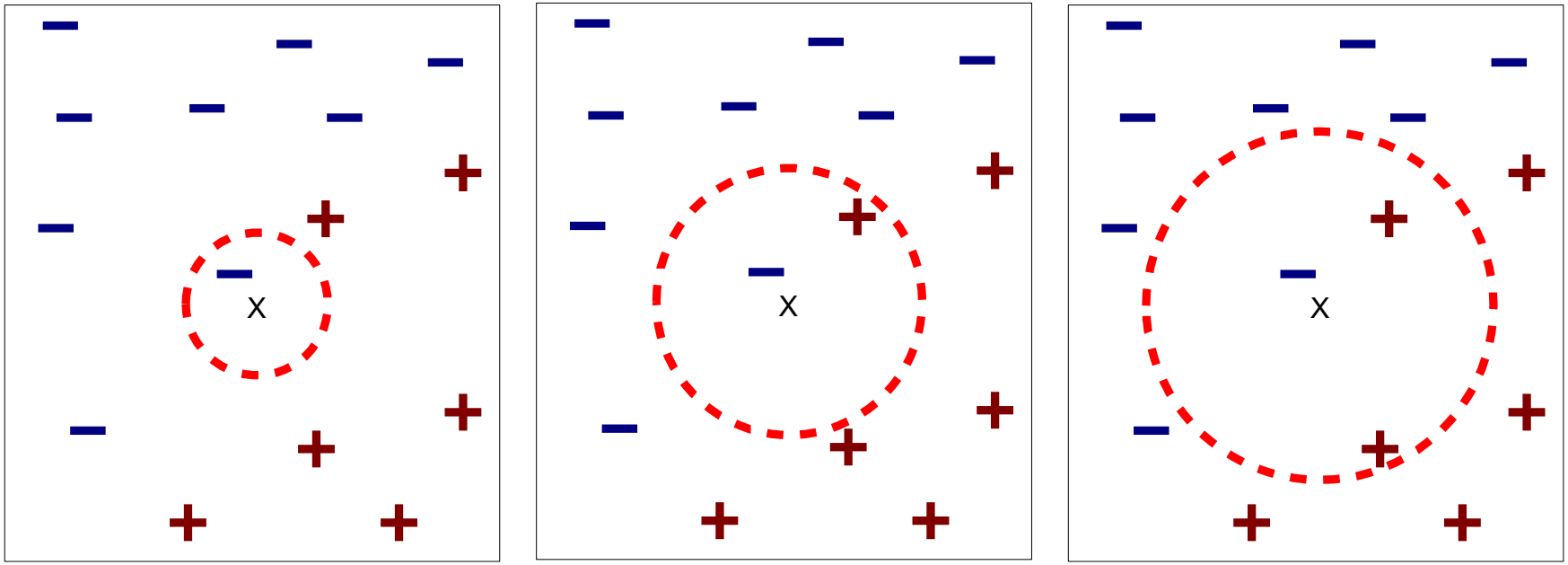


RBF-SVM example

$$\sigma = 0.1 \quad C = \infty$$



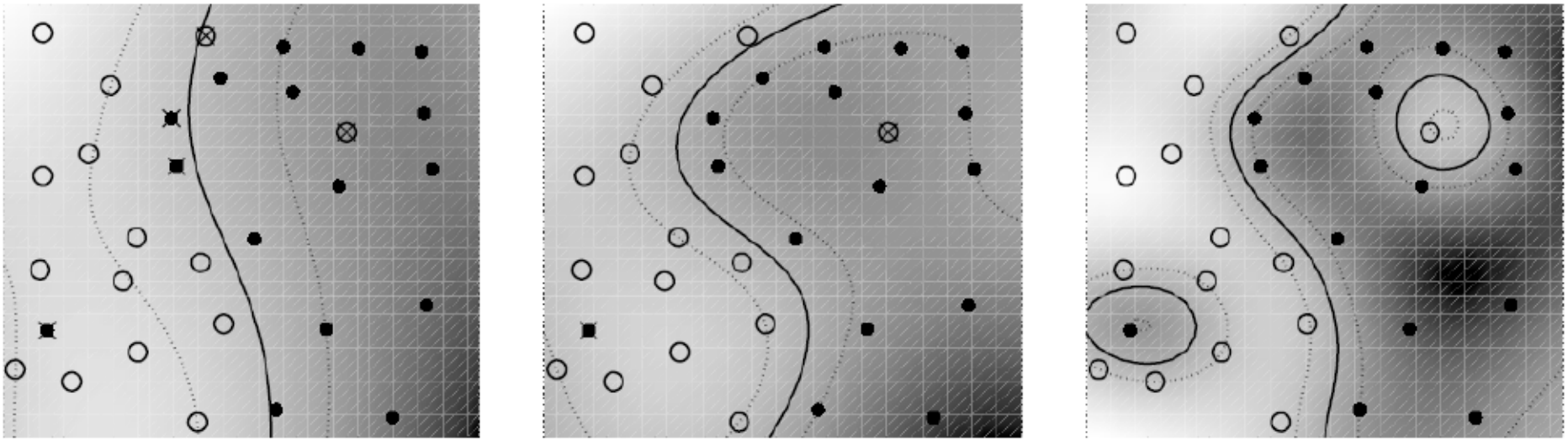
Reminder: K-nearest neighbor classifier



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

- **Compute distance to other training records**
- **Identify *K* nearest neighbors**
- **Take majority vote**

Large margins for nonlinear classifiers

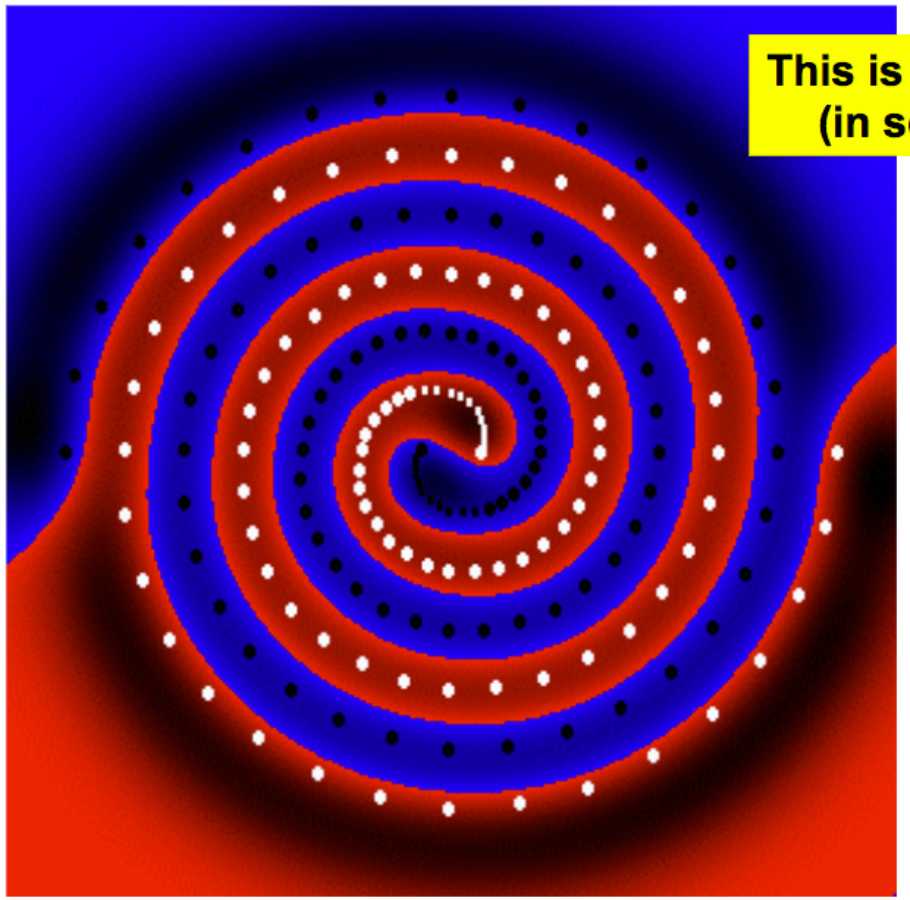


RBF Kernel width (σ)

Margin size: determined by both σ and regularizer

We can slide between a linear and a Nearest-Neighbor classifier!

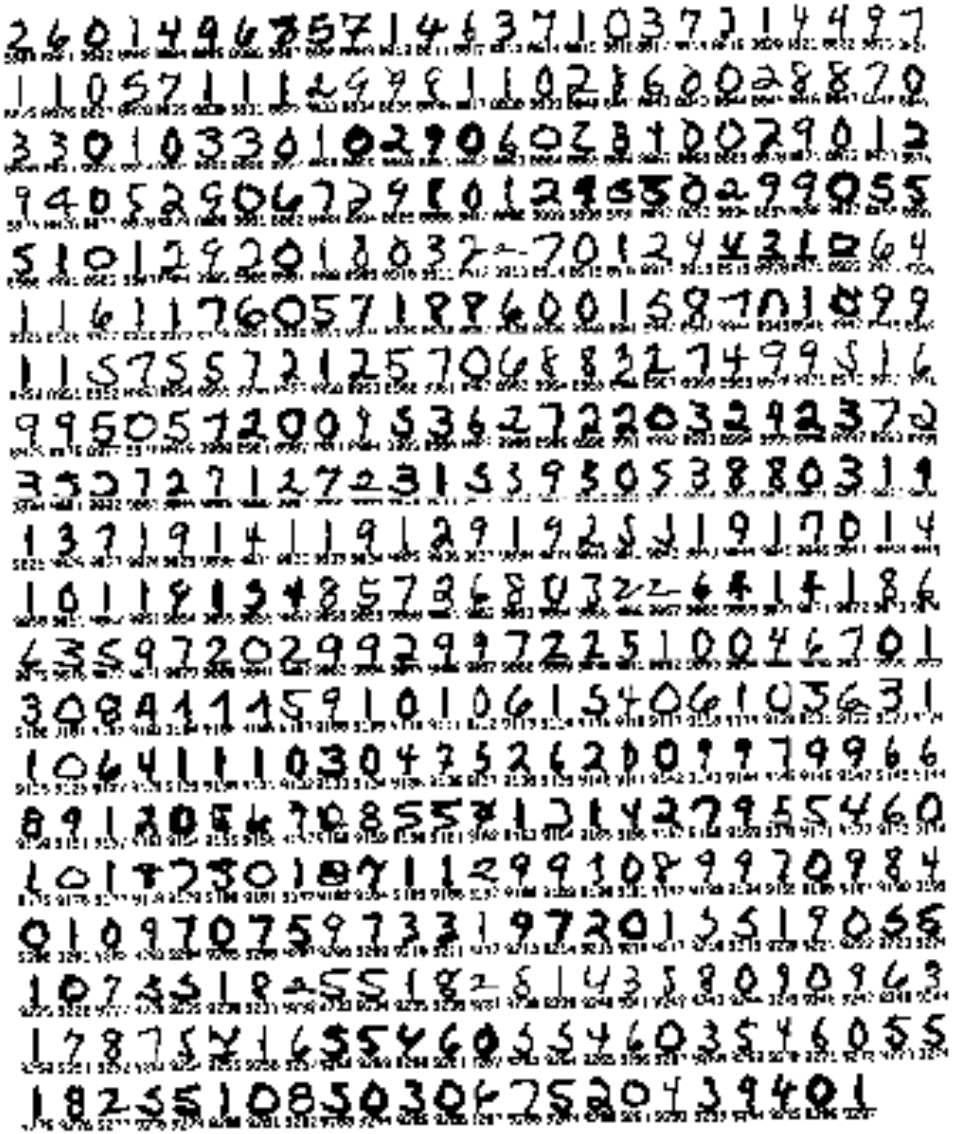
All of the flexibility you may need is there



www.kernel-methods.net

Guyon & Vapnik, 1995

- Handwritten digit recognition
  - US Postal Service Database
  - Standard benchmark task for many learning algorithms



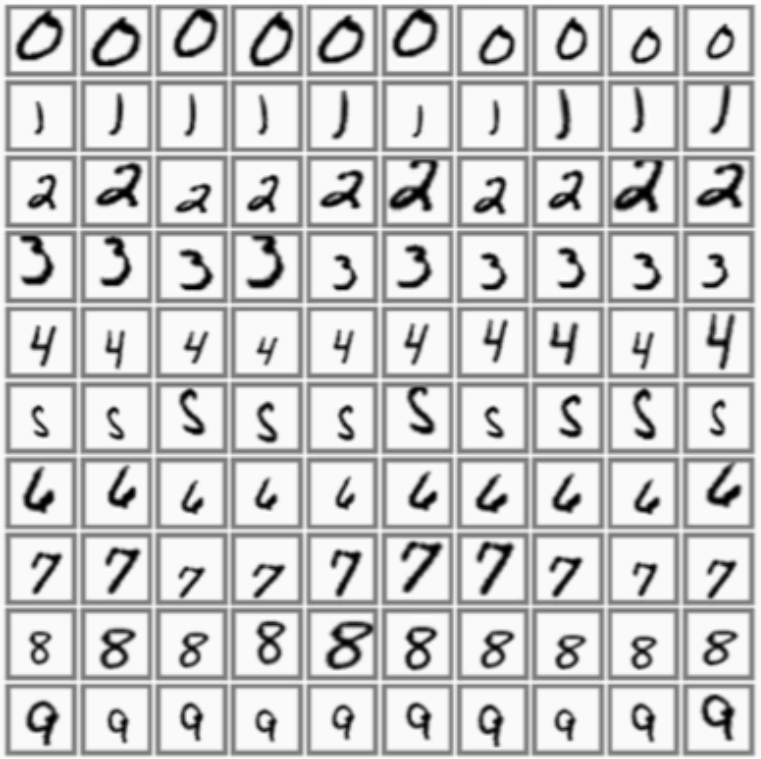
Application: Handwritten digit recognition

- Feature vectors: each image is 28 x 28 pixels. Rearrange as a 784-vector **x**

- Training: learn k=10 two-class 1 vs the rest SVM classifiers f_k (**x**)

- Classification: choose class with most positive score

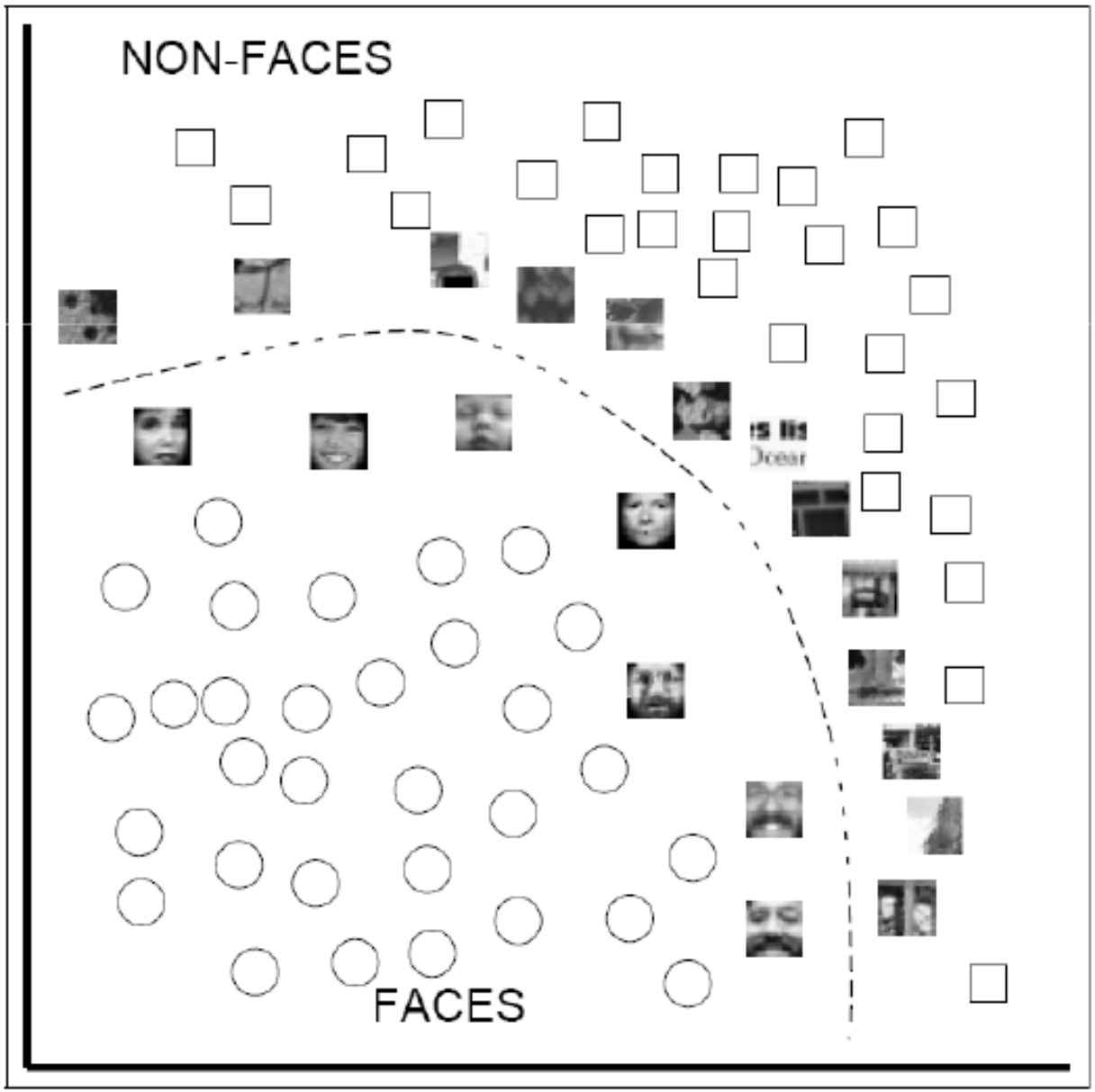
$$f(\mathbf{x}) = \max_k f_k(\mathbf{x})$$



Guyon & Vapnik 1995

- USPS benchmark
  - 2.5% error: human performance
- Different learning algorithms
  - 16.2% error: Decision tree (C4.5)
  - 5.9% error: (best) 2-layer Neural Network
  - 5.1% error: LeNet 1 - (massively hand-tuned) 5-layer network
- Different SVMs
  - 4.0% error: Polynomial kernel (p=3, 274 support vectors)
  - 4.1% error: Gaussian kernel (σ=0.3, 291 support vectors)

Support vectors for Faces (P&P 98)



Other kernels

- From http://www.kernel-methods.net/kernels.html

Kernel Functions Described in the Book:
Definition 9.1 Polynomial kernel 286
Computation 9.6 All-subsets kernel 289
Computation 9.8 Gaussian kernel 290
Computation 9.12 ANOVA kernel 293
Computation 9.18 Alternative recursion for ANOVA kernel 296
Computation 9.24 General graph kernels 301
Definition 9.33 Exponential difiusion kernel 307
Definition 9.34 von Neumann difiusion kernel 307
Computation 9.35 Evaluating difiusion kernels 308
Computation 9.46 Evaluating randomised kernels 315
Definition 9.37 Intersection kernel 309
Definition 9.38 Union-complement kernel 310
Remark 9.40 Agreement kernel 310
Section 9.6 Kernels on real numbers 311
Remark 9.42 Spline kernels 313
Definition 9.43 Derived subsets kernel 313
Definition 10.5 Vector space kernel 325
Computation 10.8 Latent semantic kernels 332
Definition 11.7 The p-spectrum kernel 342
Computation 11.10 The p-spectrum recursion 343
Remark 11.13 Blended spectrum kernel 344
Computation 11.17 All-subsequences kernel 347
Computation 11.24 Fixed length subsequences kernel 352
Computation 11.33 Naive recursion for gap-weighted subsequences kernel 358
Computation 11.36 Gap-weighted subsequences kernel 360
Computation 11.45 Trie-based string kernels 367
Algorithm 9.14 ANOVA kernel 294
Algorithm 9.25 Simple graph kernels 302
Algorithm 11.20 All-non-contiguous subsequences kernel 350
Algorithm 11.25 Fixed length subsequences kernel 352
Algorithm 11.38 Gap-weighted subsequences kernel 361
Algorithm 11.40 Character weighting string kernel 364
Algorithm 11.41 Soft matching string kernel 365
Algorithm 11.42 Gap number weighting string kernel 366
Algorithm 11.46 Trie-based p-spectrum kernel 368
Algorithm 11.51 Trie-based mismatch kernel 371
Algorithm 11.54 Trie-based restricted gap-weighted kernel 374
Algorithm 11.62 Co-rooted subtree kernel 380
Algorithm 11.65 All-subtree kernel 383
Algorithm 12.8 Fixed length HMM kernel 401
Algorithm 12.14 Pair HMM kernel 407
Algorithm 12.17 Hidden tree model kernel 411
Algorithm 12.34 Fixed length Markov model Fisher kernel 427

Text Classification: Examples

- Classify news stories as *World, US, Business, SciTech, Sports, Entertainment, Health, Other*
- Add MeSH terms to Medline abstracts
  - e.g. “Conscious Sedation” [E03.250]
- Classify business names by industry.
- Classify student essays as *A,B,C,D,* or *F.*
- Classify email as *Spam, Other.*
- Classify email to tech staff as *Mac, Windows, ..., Other.*
- Classify pdf files as *ResearchPaper, Other*
- Classify documents as *WrittenByReagan, GhostWritten*
- Classify movie reviews as *Favorable,Unfavorable,Neutral.*
- Classify technical papers as *Interesting, Uninteresting.*
- Classify jokes as *Funny, NotFunny.*
- Classify web sites of companies by Standard Industrial Classification (SIC) code.

Text Classification: Examples

- Best-studied benchmark: *Reuters-21578* newswire stories
  - 9603 train, 3299 test documents, 80-100 words each, 93 classes

ARGENTINE 1986/87 GRAIN/OILSEED REGISTRATIONS

BUENOS AIRES, Feb 26

Argentine grain board figures show crop registrations of grains, oilseeds and their products to February 11, in thousands of tonnes, showing those for future shipments month, 1986/87 total and 1985/86 total to February 12, 1986, in brackets:

- Bread wheat prev 1,655.8, Feb 872.0, March 164.6, total 2,692.4 (4,161.0).
- Maize Mar 48.0, total 48.0 (nil).
- Sorghum nil (nil)
- Oilseed export registrations were:
- Sunflowerseed total 15.0 (7.9)
- Soybean May 20.0, total 20.0 (nil)

The board also detailed export registrations for subproducts, as follows....

→ Categories: **grain**, **wheat** (of 93 binary choices)

Representing text for classification

f(ARGENTINE 1986/87 GRAIN/OILSEED REGISTRATIONS
BUENOS AIRES, Feb 26
Argentine grain board figures show crop registrations of grains, oilseeds and their products to February 11, in thousands of tonnes, showing those for future shipments month, 1986/87 total and 1985/86 total to February 12, 1986, in brackets:

- Bread wheat prev 1,655.8, Feb 872.0, March 164.6, total 2,692.4 (4,161.0).
- Maize Mar 48.0, total 48.0 (nil).
- Sorghum nil (nil)
- Oilseed export registrations were:
- Sunflowerseed total 15.0 (7.9)
- Soybean May 20.0, total 20.0 (nil)

The board also detailed export registrations for subproducts, as follows....)=y

?

simplest useful

What is the ~~**best**~~ representation for the document *x* being classified?

Bag of words representation

ARGENTINE 1986/87 **GRAIN/OILSEED** REGISTRATIONS
BUENOS AIRES, Feb 26
Argentine **grain** board figures show crop registrations of **grains**, **oilseeds** and their products to February 11, in thousands of **tonnes**, showing those for future **shipments** month, 1986/87 **total** and 1985/86 **total** to February 12, 1986, in brackets:

- Bread **wheat** prev 1,655.8, Feb 872.0, March 164.6, **total** 2,692.4 (4,161.0).
- **Maize** Mar 48.0, total 48.0 (nil).
- **Sorghum** nil (nil)
- **Oilseed** export registrations were:
- **Sunflowerseed** total 15.0 (7.9)
- **Soybean** May 20.0, total 20.0 (nil)

The board also detailed export registrations for subproducts, as follows....

→ Categories: **grain**, **wheat**

Bag of words representation

xxxxxxxxxxxxxxxxxxx **GRAIN/OILSEED** xxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxx **grain** xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx **grains**, **oilseeds**
xxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxx **tonnes**, xxxxxxxxxxxxxxxxx
shipments xxxxxxxxxxxx **total** xxxxxxxxx **total** xxxxxxxx
xxxxxxxxxxxxxxxxxxx:

- Xxxxx **wheat** xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, **total** xxxxxxxxxxxxxxxx
- **Maize** xxxxxxxxxxxxxxxx
- **Sorghum** xxxxxxxxxx
- **Oilseed** xxxxxxxxxxxxxxxxxxxx
- **Sunflowerseed** xxxxxxxxxxxxxx
- **Soybean** xxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx....

→ Categories: **grain**, **wheat**

Bag of words representation

xxxxxxxxxxxxxxxxxxx **GRAIN/OILSEED** xxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxx **grain** xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx **grains**, **oilseeds**
xxxxxxxxxx xxxxxxxxxxxxxxxxxxxxxxxxxx **tonnes**,
xxxxxxxxxxxxxxxxx **shipments** xxxxxxxxxxxx **total** xxxxxxxxx **total**
xxxxxxxx xxxxxxxxxxxxxxxxxxxx:

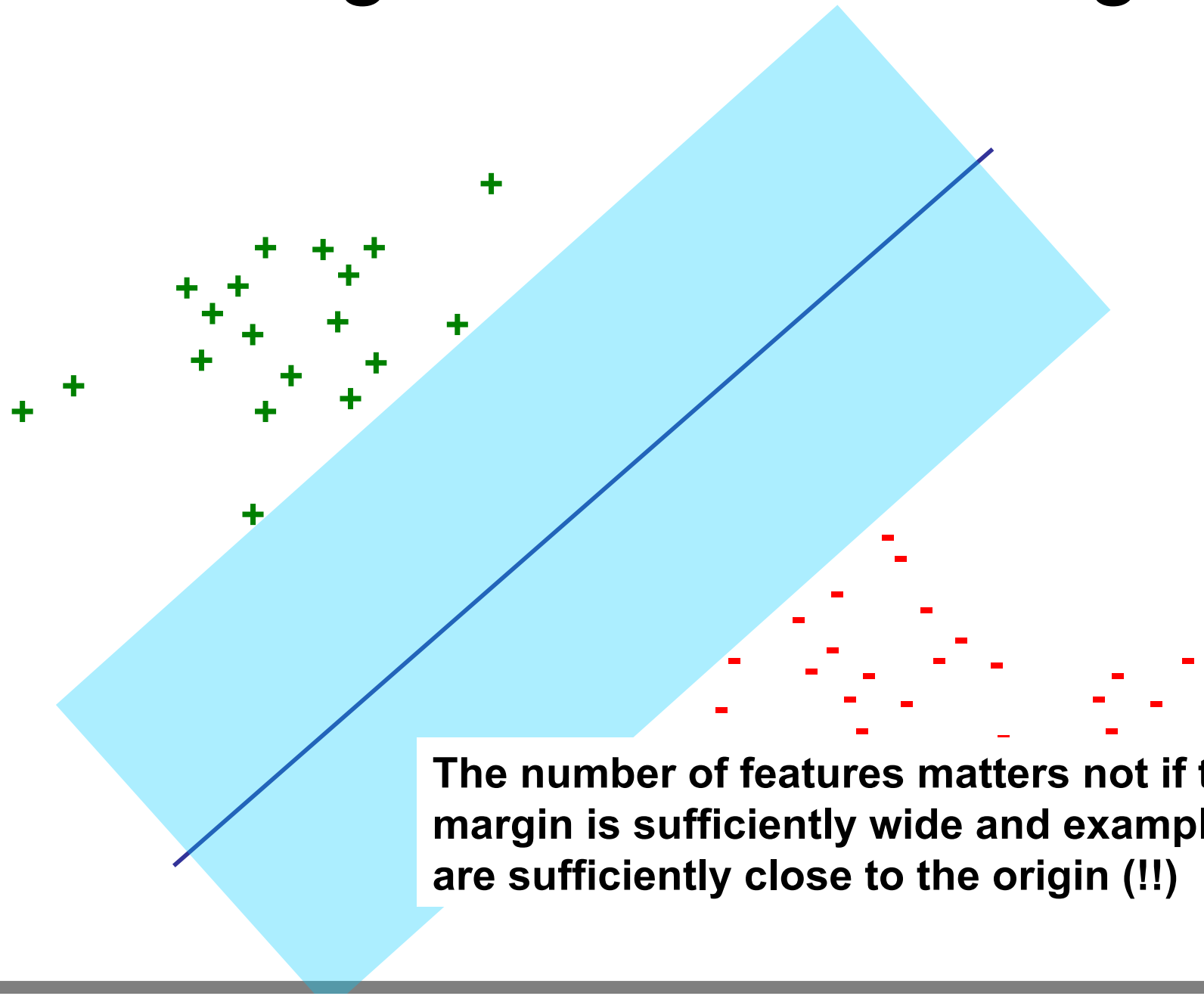
- Xxxxx **wheat** xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx, **total** xxxxxxxxxxxxxxxx
- **Maize** xxxxxxxxxxxxxxxxxx
- **Sorghum** xxxxxxxxxx
- **Oilseed** xxxxxxxxxxxxxxxxxxxxx
- **Sunflowerseed** xxxxxxxxxxxxxx
- **Soybean** xxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx....

word	*freq*
grain(s)	3
oilseed(s)	2
total	3
wheat	1
maize	1
soybean	1
tonnes	1
...	...

Categories: **grain**, **wheat**

Margin-based Learning



The number of features matters not if the margin is sufficiently wide and examples are sufficiently close to the origin (!!)

Support Vector Machine Results

					SVM (poly) degree d =					SVM (rbf) width γ =			
	Bayes	Rocchio	C4.5	k-NN	1	2	3	4	5	0.6	0.8	1.0	1.2
earn	95.9	96.1	96.1	97.3	98.2	98.4	**98.5**	98.4	98.3	**98.5**	98.5	98.4	98.3
acq	91.5	92.1	85.3	92.0	92.6	94.6	**95.2**	95.2	95.3	95.0	95.3	95.3	**95.4**
money-fx	62.9	67.6	69.4	78.2	66.9	72.5	75.4	74.9	**76.2**	74.0	75.4	**76.3**	75.9
grain	72.5	79.5	89.1	82.2	91.3	93.1	**92.4**	91.3	89.9	**93.1**	91.9	91.9	90.6
crude	81.0	81.5	75.5	85.7	86.0	87.3	88.6	**88.9**	87.8	**88.9**	89.0	88.9	88.2
trade	50.0	77.4	59.2	77.4	69.2	75.5	76.6	77.3	**77.1**	76.9	78.0	**77.8**	76.8
interest	58.0	72.5	49.1	74.0	69.8	63.3	67.9	73.1	**76.2**	74.4	75.0	**76.2**	76.1
ship	78.7	83.1	80.9	79.2	82.0	85.4	86.0	**86.5**	86.0	**85.4**	86.5	87.6	87.1
wheat	60.6	79.4	85.5	76.6	83.1	84.5	85.2	**85.9**	83.8	**85.2**	85.9	85.9	85.9
corn	47.3	62.2	87.7	77.9	86.0	86.5	85.3	**85.7**	83.9	**85.1**	85.7	85.7	84.5
microavg.	**72.0**	**79.9**	**79.4**	**82.3**	84.2	85.1	85.9	86.2	85.9	86.4	86.5	86.3	86.2
					combined: **86.0**					combined: **86.4**			

Sequence Data versus Structure and Function

Sequences for four chains of human hemoglobin

```
>1A3N:A HEMOGLOBIN
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHGK
KVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPA
VHASLDKFLASVSTVLTSKYR
>1A3N:B HEMOGLOBIN
VHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKV
KAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFGK
EFTPPVQAAYQKVVAGVANALAHKYH
>1A3N:C HEMOGLOBIN
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHGK
KVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPA
VHASLDKFLASVSTVLTSKYR
>1A3N:D HEMOGLOBIN
VHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKV
KAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFGK
EFTPPVQAAYQKVVAGVANALAHKYH
```

Tertiary Structure

Function:
oxygen transport

Learning Problem

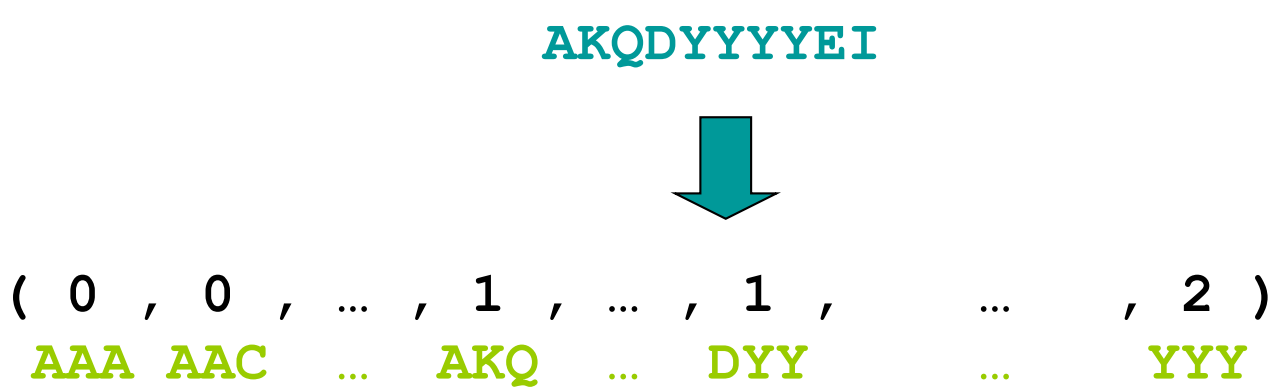
- Reduce to binary classification problem: positive (+) if example belongs to a family (e.g. G proteins) or superfamily (e.g. nucleoside triphosphate hydrolases), negative (-) otherwise
- Use *supervised learning* approach to *train* a classifier

SVMs for Protein Classification

- Want to define feature map from space of protein sequences to vector space
- Goals:
  - Computational efficiency
  - Competitive performance with known methods
  - No reliance on generative model – general method for sequence-based classification problems

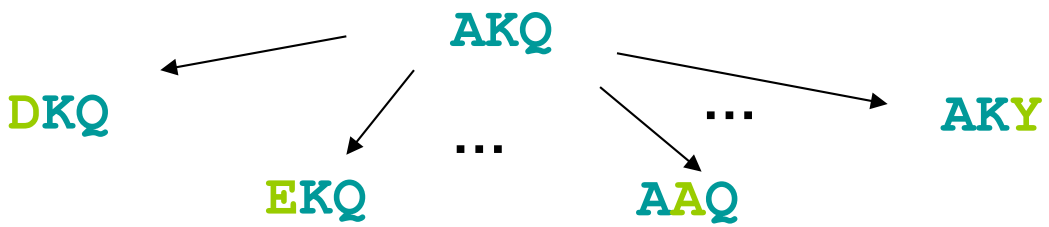
k-Spectrum Feature Map

- Feature map for k-spectrum with no mismatches:
  - For sequence x, $F_{(k)}(x) = (F_t(x))_{\{k\text{-mers } t\}}$, where $F_t(x)$ = #occurrences of t in x



(k,m)-Mismatch Feature Map

- Feature map for k-spectrum, allowing m mismatches:
  - if s is a k-mer, $F_{(k,m)}(s) = (F_t(s))_{\{\text{k-mers } t\}}$, where $F_t(s) = 1$ if s is within m mismatches from t, 0 otherwise
  - extend additively to longer sequences x by summing over all k-mers s in x



The Kernel Trick

- To train an SVM, can use *kernel* rather than explicit feature map
  - For sequences x, y, feature map F, kernel value is inner product in feature space:
    $K(x, y) = \langle F(x), F(y) \rangle$
  - Gives sequence similarity score
  - Example of a string kernel
  - Can be efficiently computed via traversal of *trie data structure*
- *Further reading:*

C. Leslie, E. Eskin, and W. Noble, *The Spectrum Kernel: A String Kernel for SVM Protein Classification.* Pacific Symposium on Biocomputing, 2002.

C. Leslie, E. Eskin, J. Weston and W. Noble, *Mismatch String Kernels for SVM Protein Classification.* NIPS 2002.

D. Haussler, Convolution kernels on discrete structures, 1999

Lecture outline

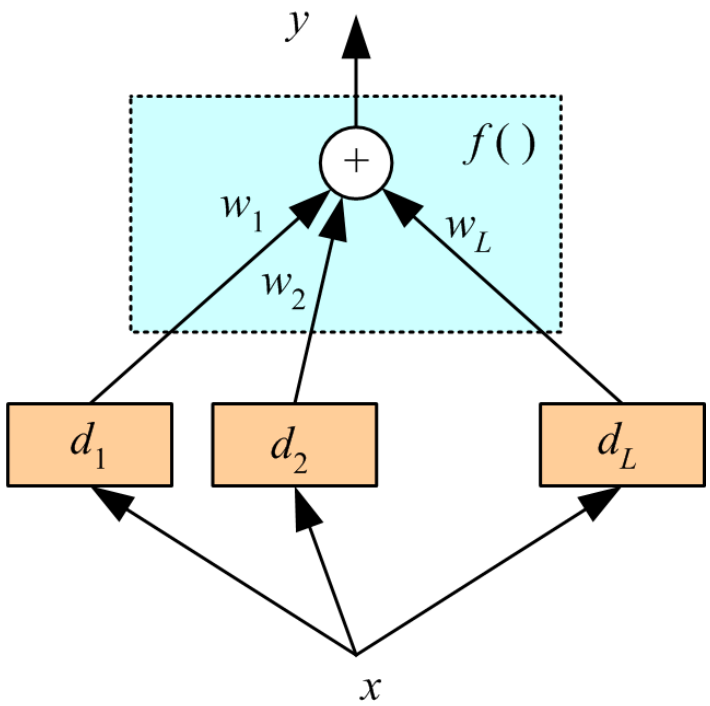
Adaboost

Decision Trees

Random Forests

Voting Methods

- Give up idea of building `the' classifier
- Generate a group of base-learners which has higher accuracy when combined
- Main tasks
  - Generating the learners
  - Combining them



Why should this work?

- Committee of M predictors for target output $y_{COM}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x})$
- Output: true value + error $y(\mathbf{x}) = h(\mathbf{x}) + \epsilon(\mathbf{x})$
- Expected sum of squares error for m-th expert:

$$\mathbb{E}_{\mathbf{x}}[(y_m(\mathbf{x}) - h(\mathbf{x}))^2] = \mathbb{E}_{\mathbf{x}}[e_m(\mathbf{x})^2]$$

- Average error of individual members:

$$\mathbb{E}_{AV} = \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})^2\right]$$

- Average error of committee:

$$\mathbb{E}_{COM} = \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x}) - h(\mathbf{x})\right\}^2\right] = \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\mathbf{x})\right\}^2\right]$$

Why should this work? (continued)

- Average error of committee:

$$\mathbb{E}_{COM} = \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x}) - h(\mathbf{x})\right\}^2\right] = \mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\mathbf{x})\right\}^2\right]$$

- **If** committee members have uncorrelated errors: $\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\epsilon_j(\mathbf{x})\right] = 0$ **then:**

$$\mathbb{E}_{\mathbf{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\mathbf{x})\right\}^2\right] = \mathbb{E}_{\mathbf{x}}\left[\frac{1}{M^2}\sum_{m=1}^{M}\left\{\epsilon_m^2(\mathbf{x}) + \sum_{k\neq m}\epsilon_m(\mathbf{x})\epsilon_k(\mathbf{x})\right\}\right]$$

$$= \frac{1}{M}\left[\frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_{\mathbf{x}}\left[\epsilon_m^2(\mathbf{x})\right]\right]$$

In conclusion: $\mathbb{E}_{COM} = \frac{1}{M}\mathbb{E}_{AV}$

Consider having a jury in an exam: each examiner has own criteria, but they can't all be wrong by accident.

Ensemble methods

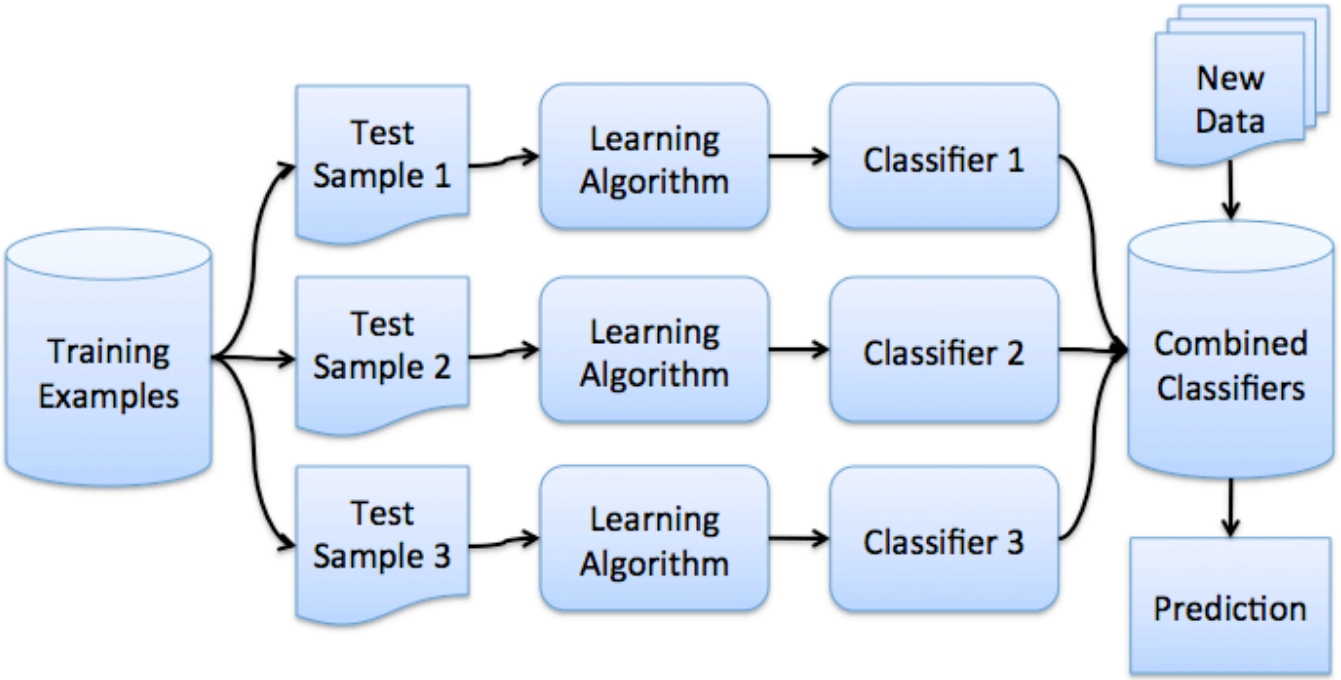
1) Train a set of simple classifiers (‘weak learners’) so that they are complementary to each other

2) Obtain an aggregate decision by combining their results

Two main methods: bagging & boosting

Bagging Idea

- General algorithm
  - Iterate
    - Pick subset of training data
    - Obtain weak learner
      - Easy to train and evaluate
  - Output final classifier by majority voting of the weak learners



We have done this before already…

- Cross-Validation
  - Split the available data into N disjunct subsets.
  - In each run, train on N-1 subsets for training a classifier.
  - Estimate the generalization error on the held-out validation set

- E.g. 5-fold cross-validation

train	train	train	train	**test**
train	train	train	**test**	train
train	train	**test**	train	train
train	**test**	train	train	train
test	train	train	train	train

Boosting Idea

- General algorithm
  - Iterate
    - Pick subset of training data using a sampling distribution
    - Obtain weak learner
      - Use weak learner to update sampling distribution

Adaboost

- Adaptive Boosting
  - `A decision theoretic generalization of on-line learning and an application to boosting', Freund and Schapire '95
  - Versatile (Discrete data, arbitrary distributions, multi-class, regression)
  - Very easy to program
  - Excellent results
- Defines a classifier using an additive model (weighted voting):

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

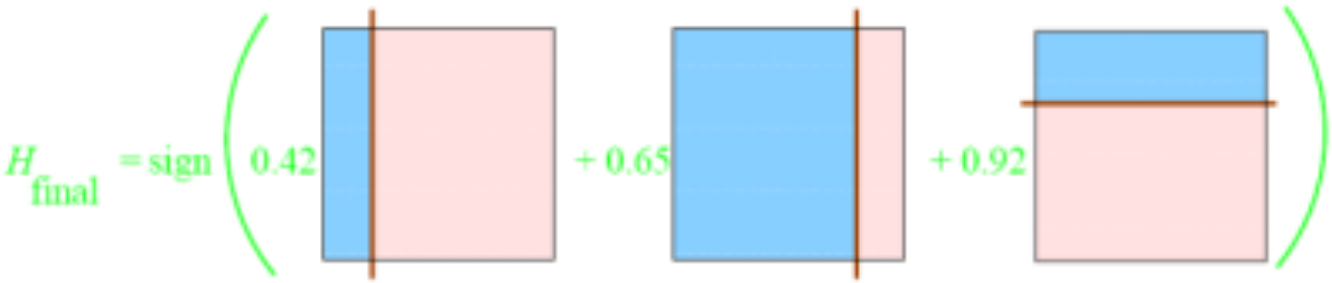
Strong classifier

Weak classifier

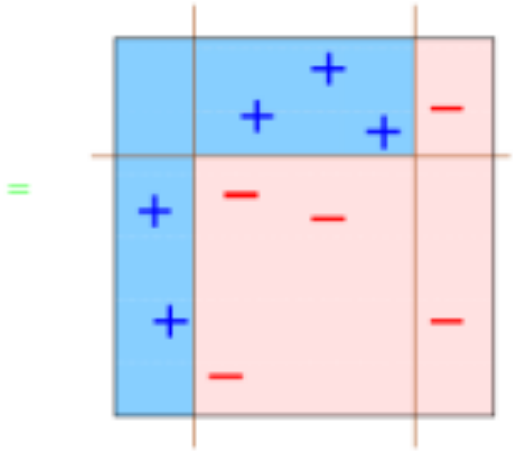
Weight

Feature vector

Adaboost, in pictures

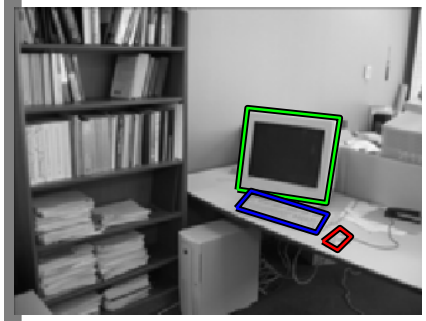


elementary functions

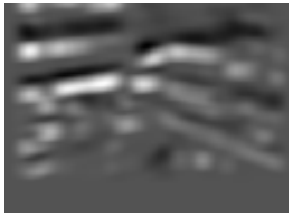


nonlinear decision boundary!

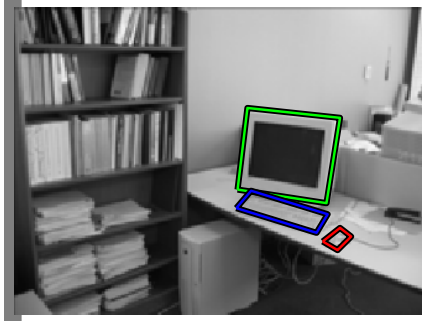
Example: screen detection



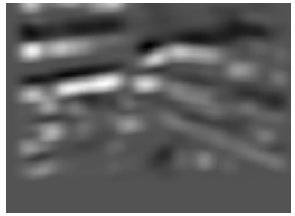
Feature output



Example: screen detection

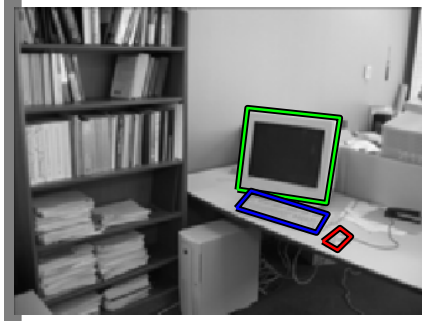


Feature output

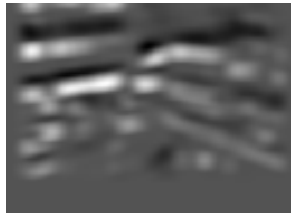


Thresholded output

Example: screen detection



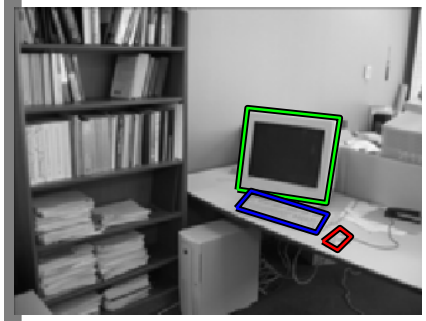
Feature output



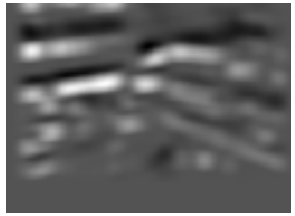
Thresholded output

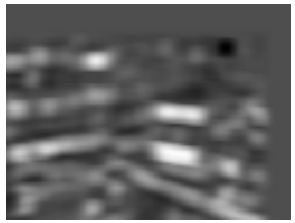
Strong classifier at iteration 1

Example: screen detection



Feature output

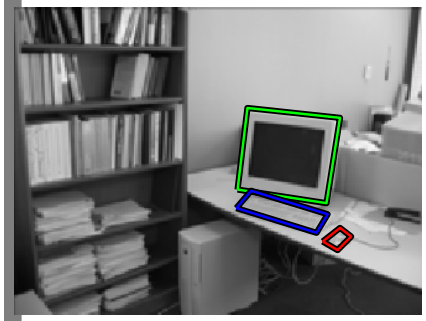


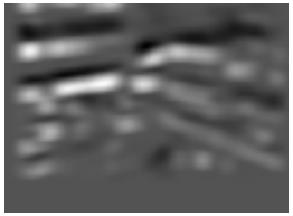


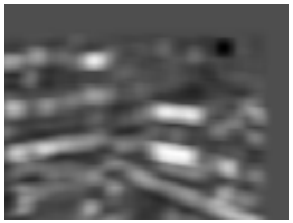
Thresholded output

Strong classifier

Example: screen detection

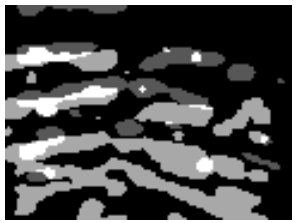






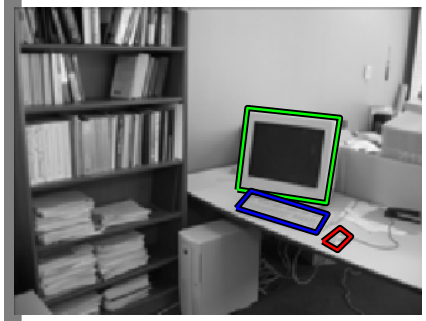
Feature output

Thresholded output

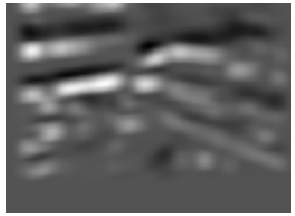


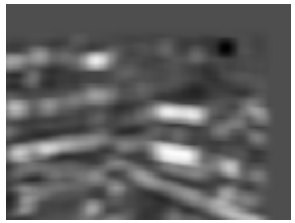
Strong classifier

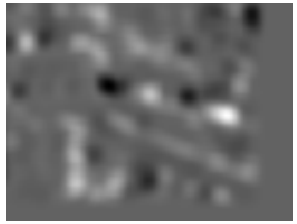
Example: screen detection



Feature output

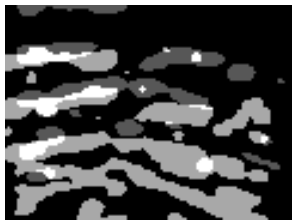


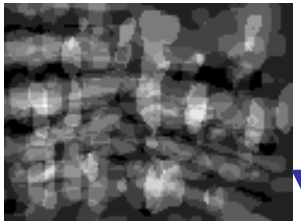




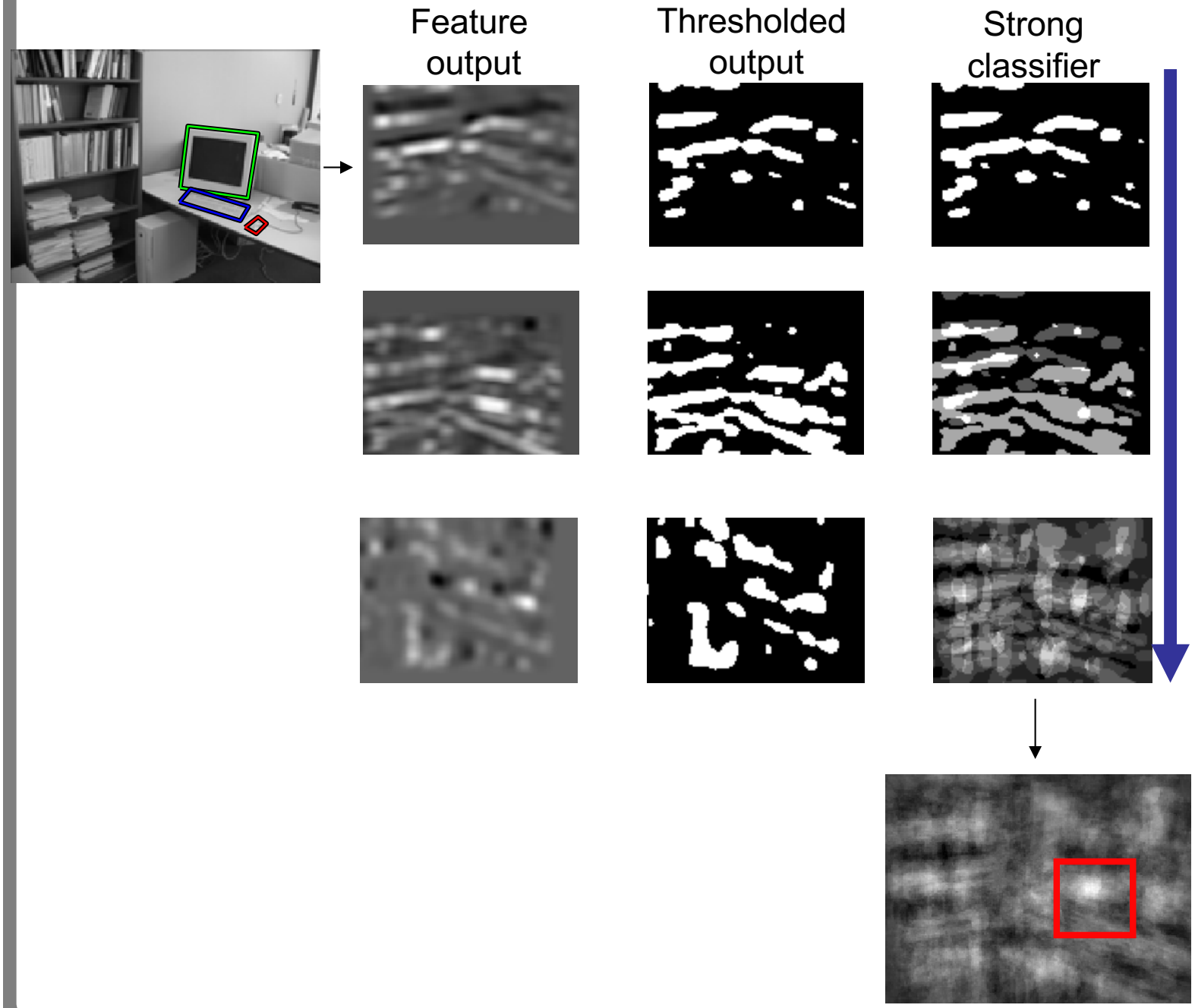
Thresholded output

Strong classifier



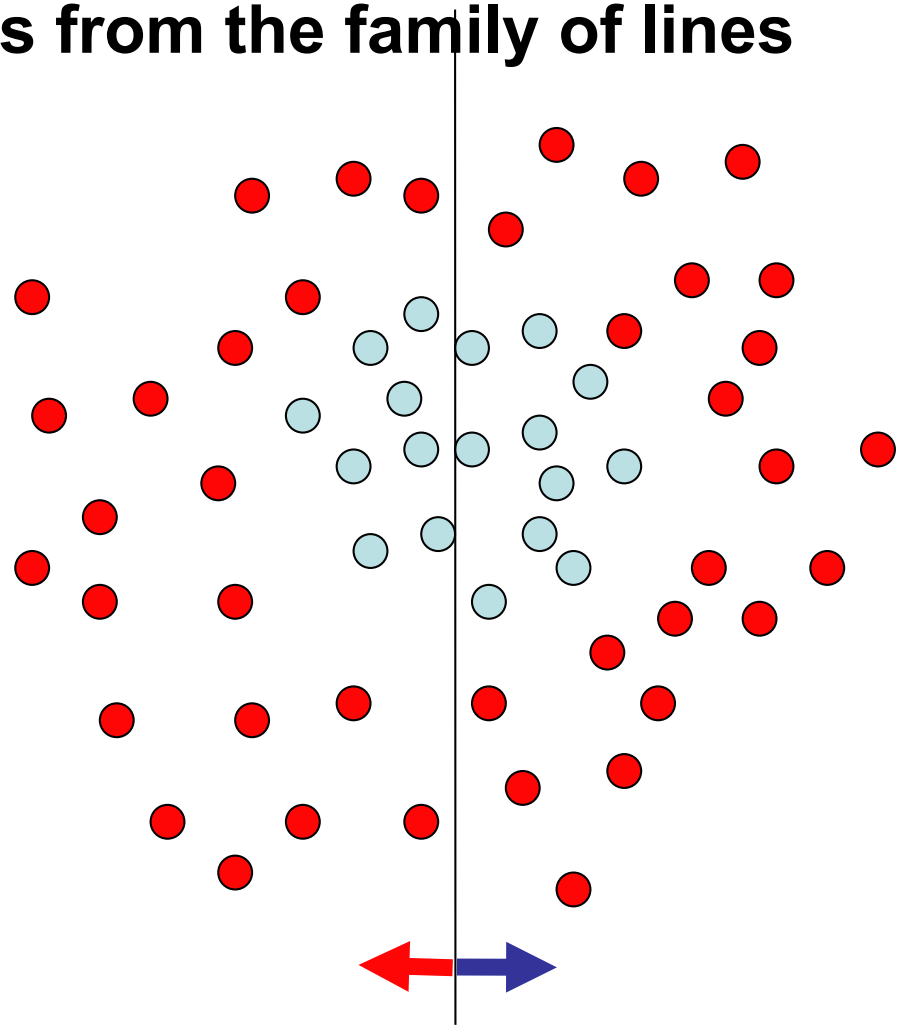


Example: screen detection



Toy example

Weak learners from the family of lines



Each data point has a class label:

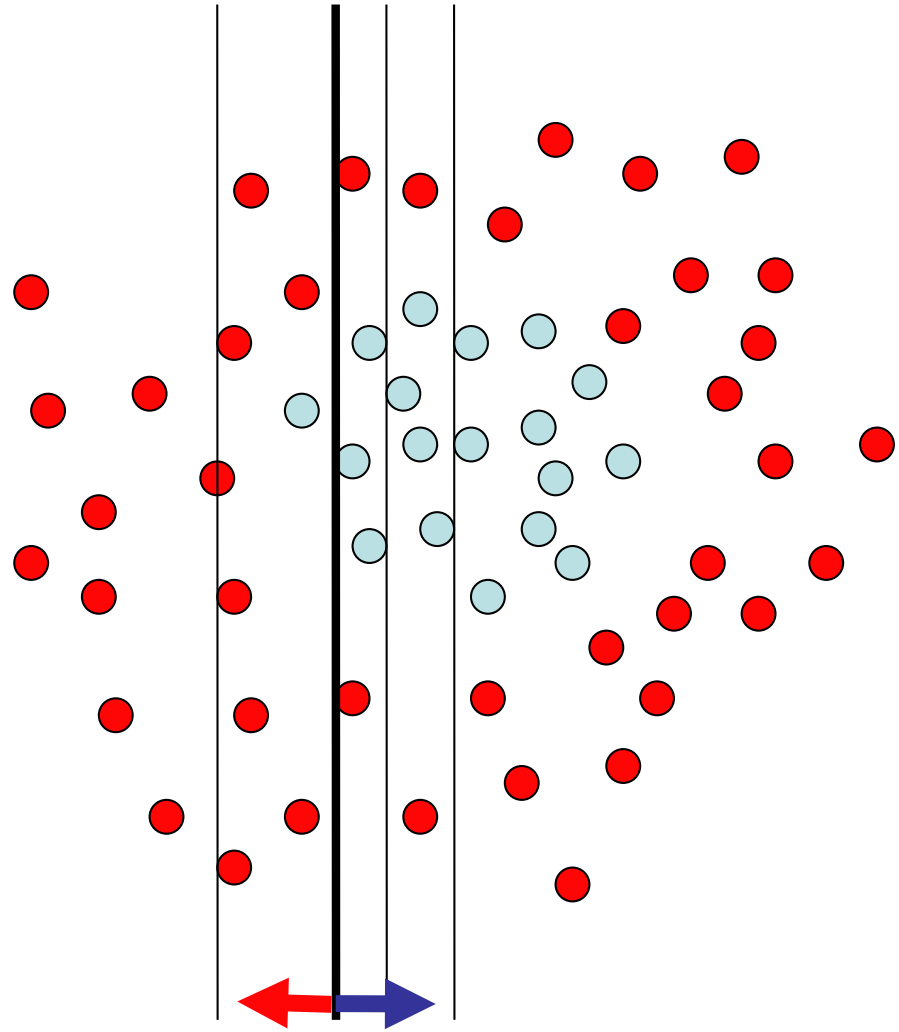
$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

and a weight:

$$w_t = 1$$

h => p(error) = 0.5 it is at chance

Toy example



Each data point has a class label:

$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

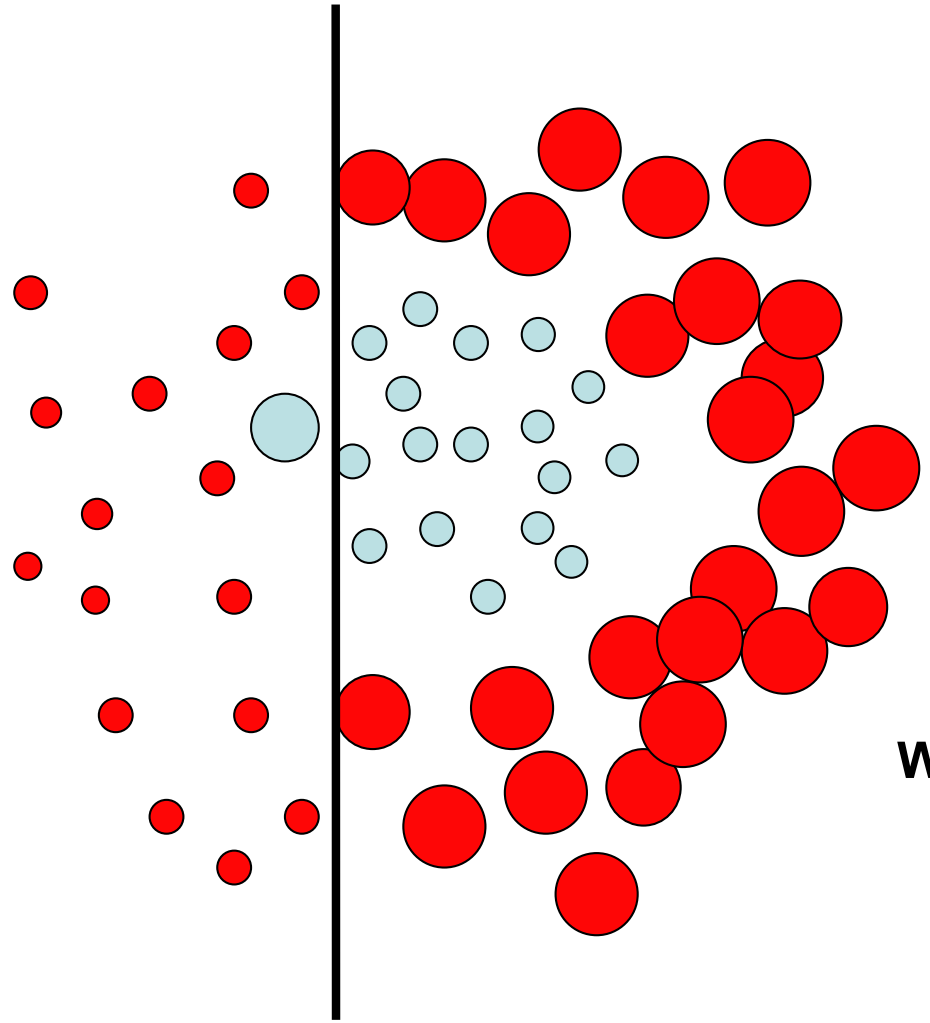
and a weight:

$$w_t = 1$$

This one seems to be the best

This is a ‘weak classifier’: It performs slightly better than chance.

Toy example



Each data point has a class label:

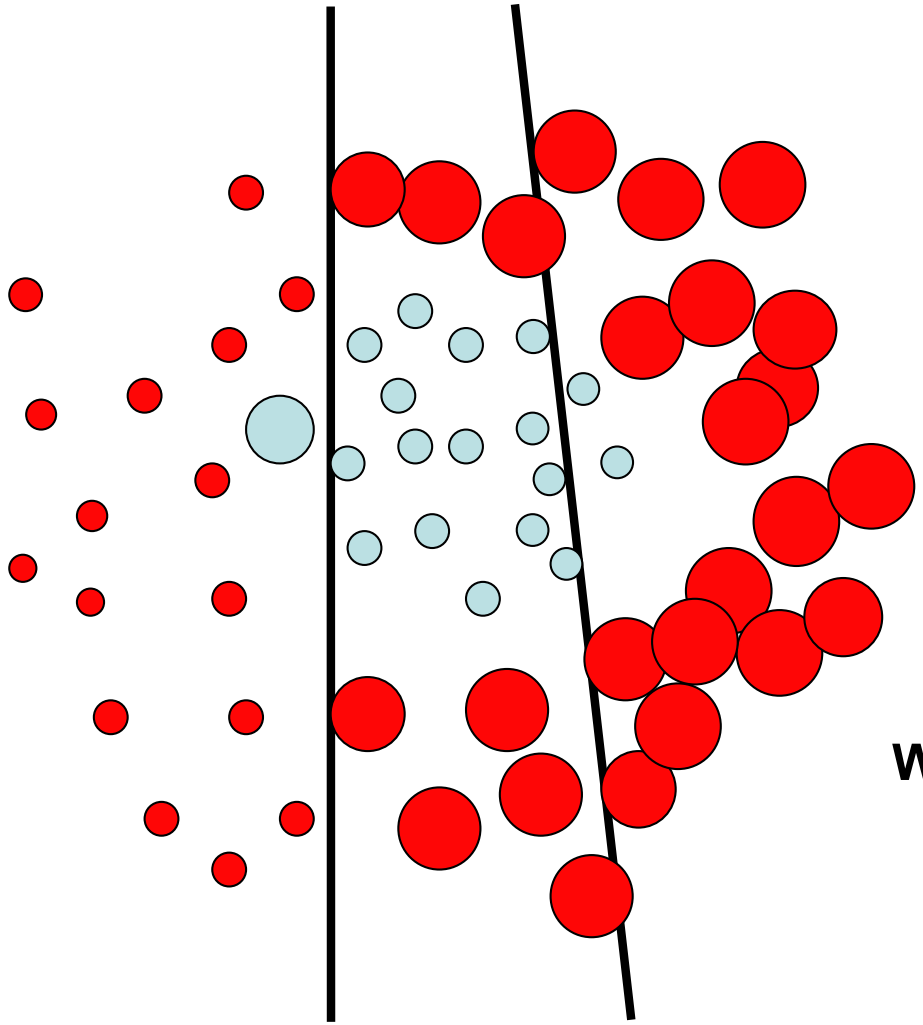
$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



Each data point has a class label:

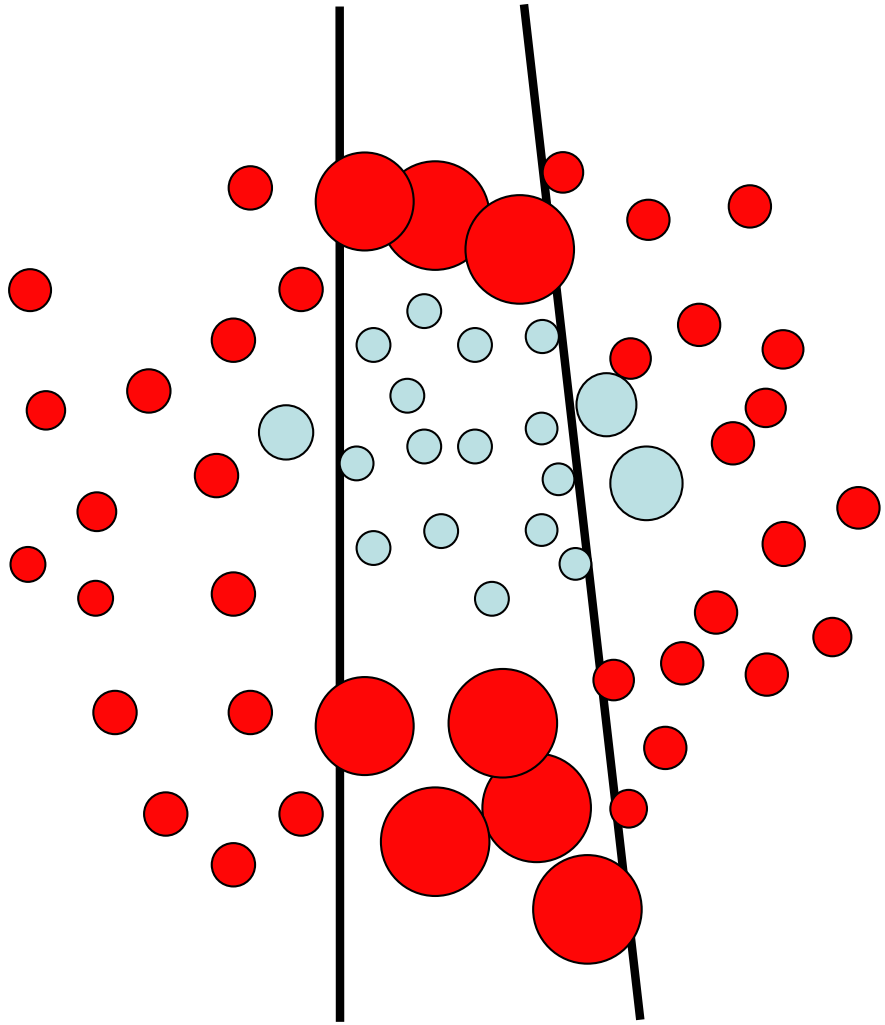
$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



Each data point has a class label:

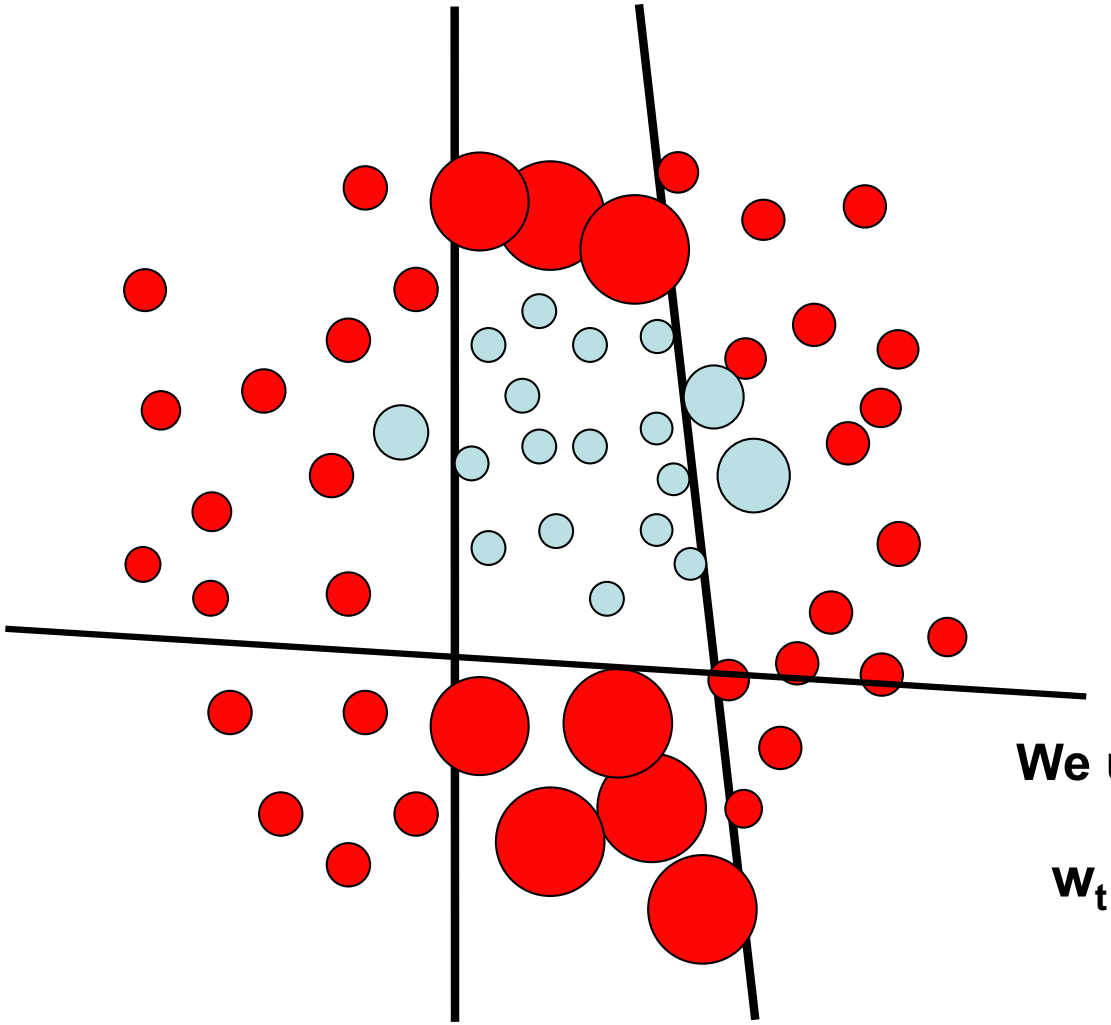
$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example



Each data point has a class label:

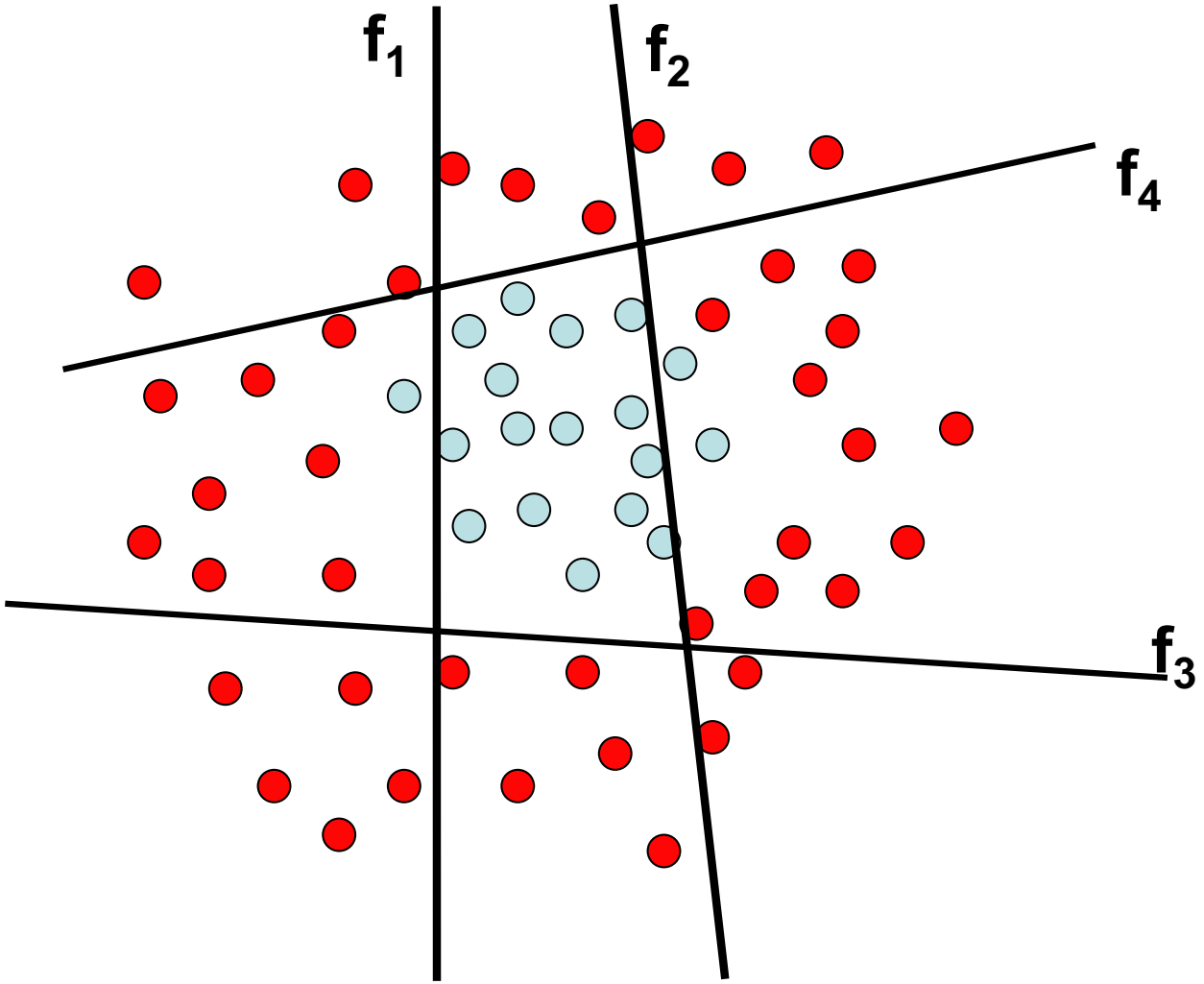
$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

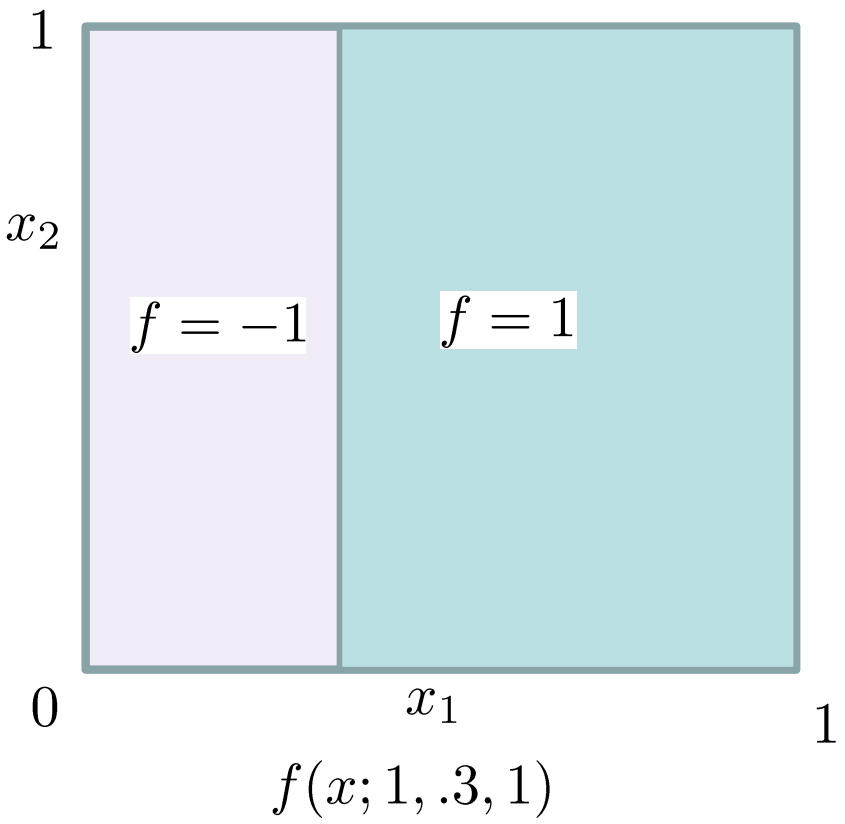
Toy example



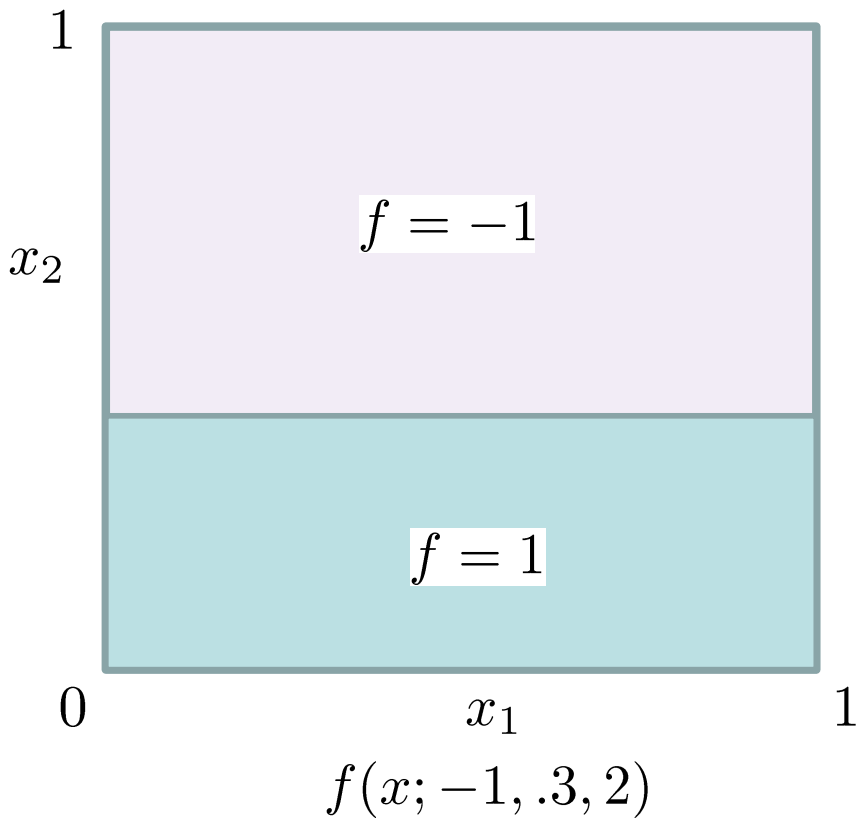
The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

Weak Learner Example

- Decision stump $f(x; s, \theta, i) = 2s[x_i > \theta] - s, \quad s \in \{-1, 1\}$



$f(x; 1, .3, 1)$



$f(x; -1, .3, 2)$

  - Extremely easy to train and evaluate

Adaboost Algorithm

- Given: $(x^i, y^i), x^i \in \mathcal{X}, y^i \in \{-1, 1\}, i = 1, \ldots, N$
- Initialize: $D_1(i) = \frac{1}{N}$ **D: distribution on samples**
- For $t = 1 \ldots T$
  - Find classifier $h_t : \mathcal{X} \to \{-1, 1\}$ with smallest weighted error

$$\epsilon_t = \frac{\sum_{i=1}^{N} D_t^i [y^i \neq h_t(x^i)]}{\sum_i D_t^i}$$

  - Update distribution

$$D_{t+1}^i = \frac{D_t^i}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if} y^i = h_t(x^i) \\ \exp(\alpha_t), & \text{if} y^i \neq h_t(x^i) \end{cases}$$

$$a_t = \frac{1}{2} \log \frac{(1-\epsilon_t)}{\epsilon_t} \qquad = \frac{D_t^i}{Z_t} \exp(-\alpha_t y^i h_t(x^i))$$

$$Z_t = \sum_i D_t^i \exp(-\alpha_t y^i h_t(x^i))$$

- Final classifier $H(x) = \text{sign}\left(\sum_t a_t h_t(x)\right)$

Optimization perspective of Adaboost

Claim: each round minimizes: $\sum_{i=1}^{N} \exp(-y^i f(x^i))$

by adding a new term to current f: $f'(x) = f(x) + \alpha h(x)$

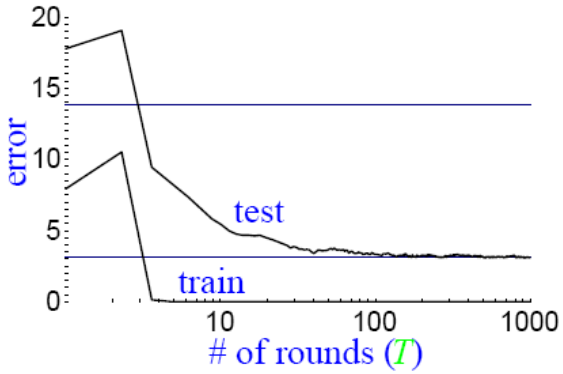
Proof: $\sum_{i=1}^{N} \exp(-y^i f'(x^i)) \quad = \quad \sum_{i=1}^{M} \exp(-y^i [f(x^i) + \alpha h(x^i)])$

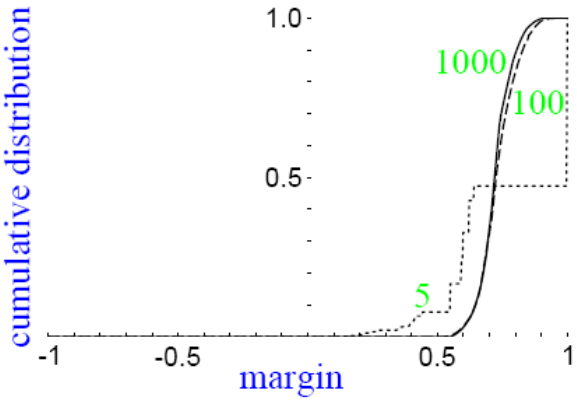
Choice of h mimimizes: $\epsilon = \sum_{i} D^i y^i h(x^i)$

Choice of α mimimizes: $\sum_{i=1}^{N} D^i \exp(-\alpha y^i h(x^i))$ $\qquad a_j = \frac{1}{2} \log \frac{(1-\epsilon_j)}{\epsilon_j}$

Generalization Error for Adaboost

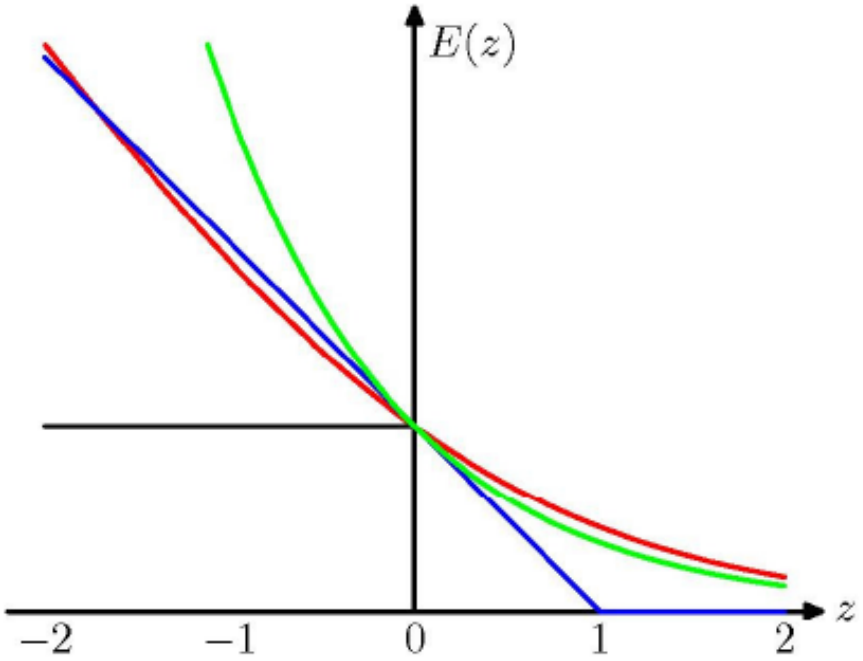
- **Empirical Evidence**





- **Theoretical Justification:**
  - **Margin of example i:** $y_i f(x_i)$
  - **Adaboost is increasing the margins of the training set**
  - **Large margin classifiers lead to good generalization**

Loss functions



- z: y*f(x)
- Ideal misclassification cost H(-z) (# training errors)
- Exponential Error exp(-z) (Adaboost)
- Cross Entropy error ln(1 + exp(-z)) (Logistic regression)
- Hinge loss max(0,1-z) (SVMs)

Using Adaboost to compute posteriors

- Consider `Empirical distribution' of data

$$E_{X,Y} \exp(-yf(x)) = \int_x \sum_y \exp(-yf(x))P(x,y)dxdy$$

- Optimize w.r.t $f(x)$: $$f(x) = \log \frac{P(y=1|x)}{P(y=-1|x)}$$

- We can therefore use $f(x)$ for a probabilistic classifier

Application to Vision

- Viola & Jones, `Rapid Object Detection using a Boosted Cascade of Simple Features', CVPR 01
  - First reliable, real-time face detection system
  - Used in commercial products, e.g. digital cameras
- Sample detections (back in 2001)

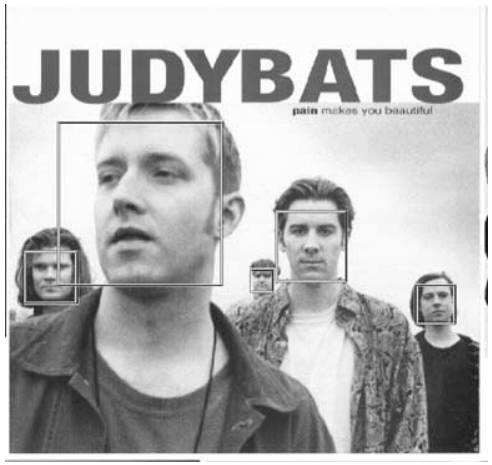
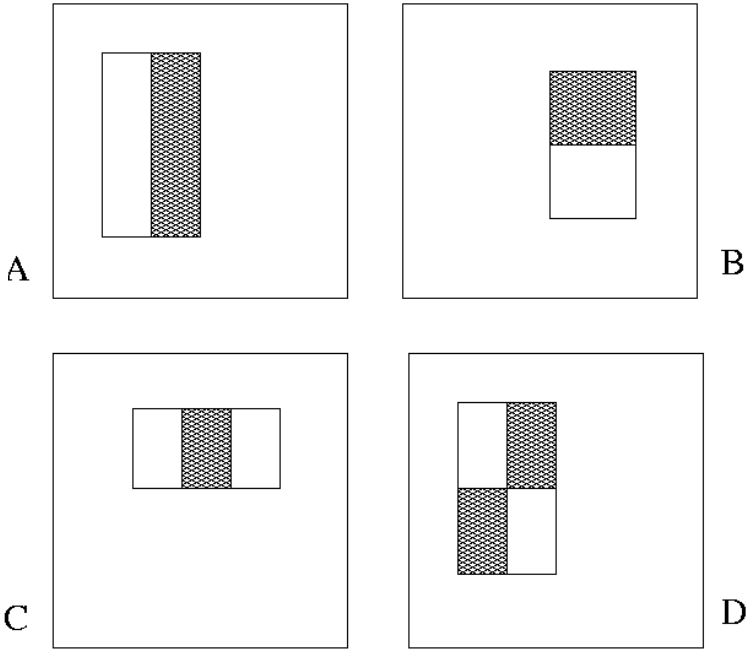


Image Features (Weak learners)

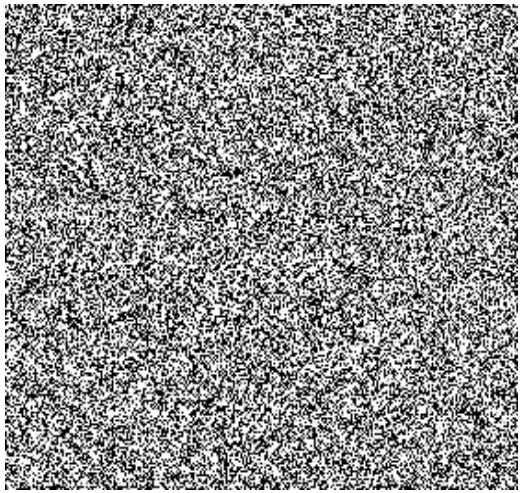
“Rectangle filters”



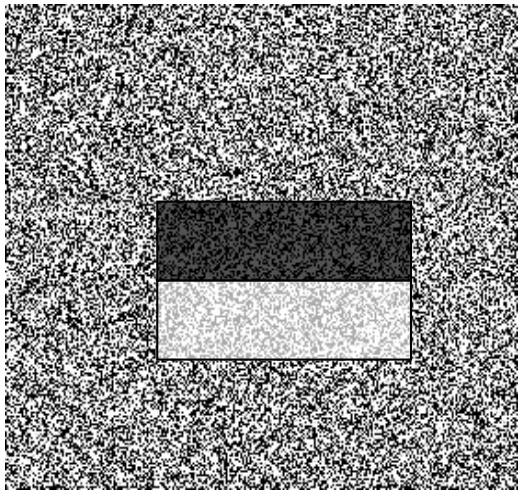
Value* = ∑ *(pixels in white area)* – ∑ *(pixels in black area)

Decision stump: Threshold difference
Why these features?
Extremely fast to compute (4 pixel operations per box)

Example



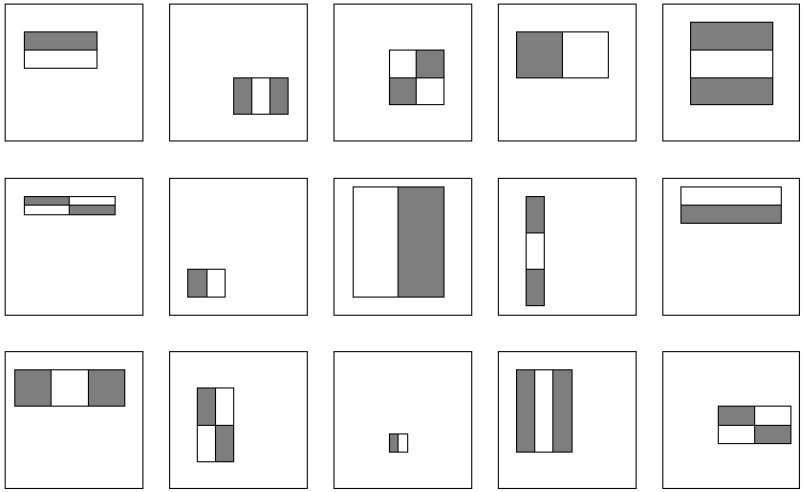
Source



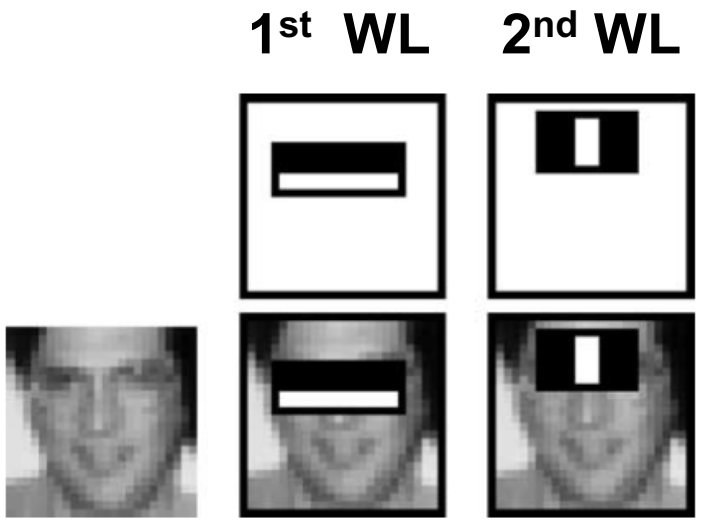
Result

Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~180,000!

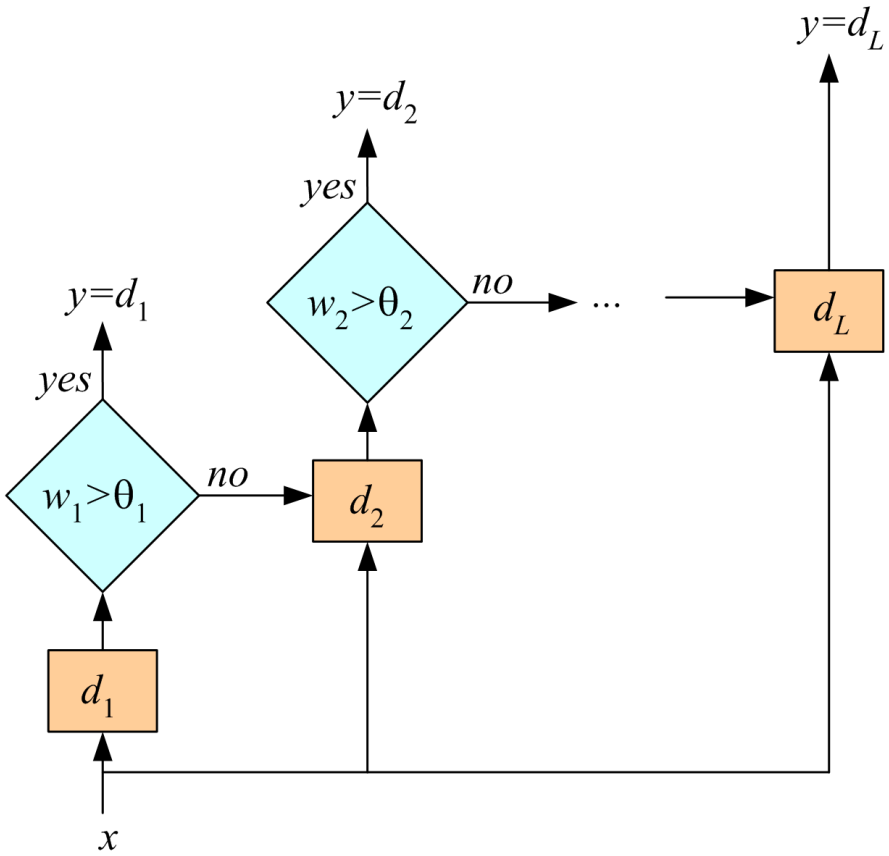


- Chosen automatically by Adaboost



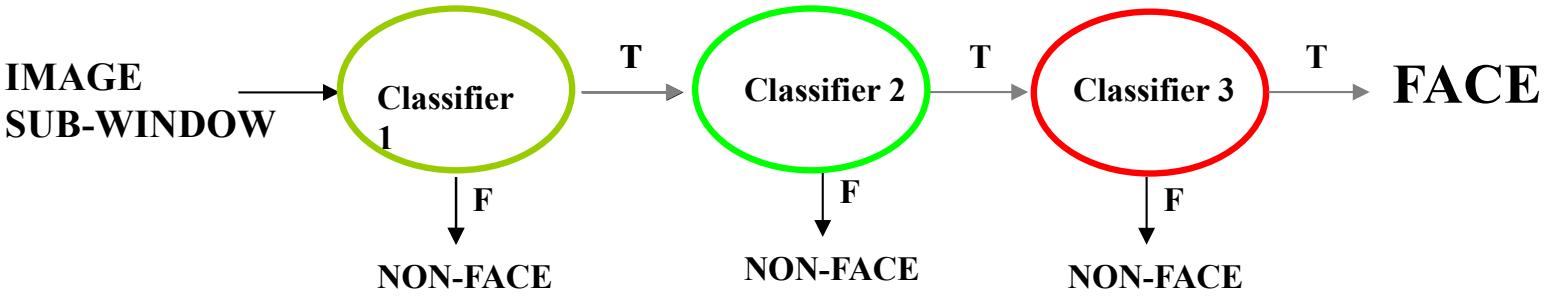
Viola & Jones: Cascade

- Reject large portions of the image based on thresholding first few weak learners



Speedup tricks

- Attentional cascade



- Integral images (4 pixel operations per filter)
- C++ implementation available in OpenCV [Lienhart, 2002]
  - http://sourceforge.net/projects/opencvlibrary/
- Matlab wrappers for OpenCV code available, e.g. here
  - http://www.mathworks.com/matlabcentral/fileexchange/19912

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

The implemented system

- Training Data
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - 9500 non-face images
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose

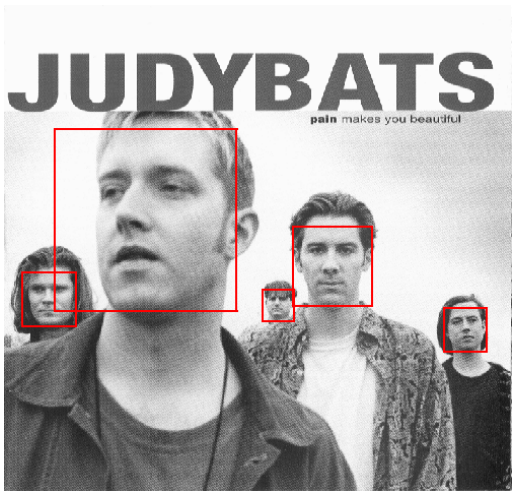
P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

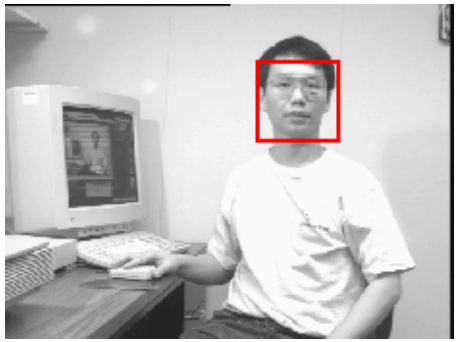
System performance

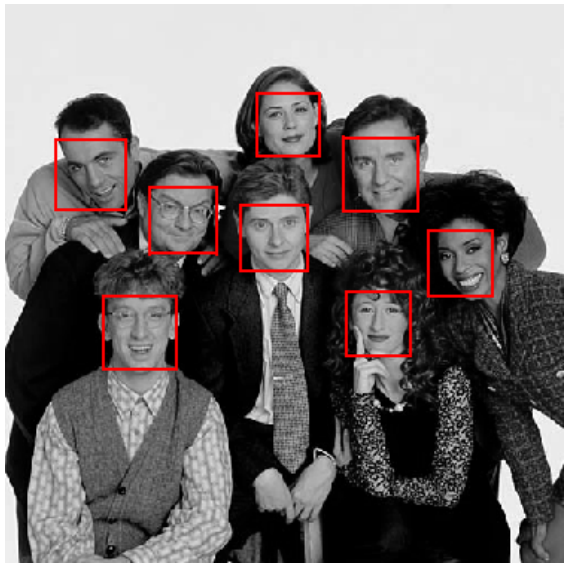
- Training time: “weeks” on 466 MHz Sun workstation
- 38 layers, total of 6061 features
- Average of 10 features evaluated per window on test set
- “On a 700 Mhz Pentium III processor, the face detector can process a 384 by 288 pixel image in about .067 seconds”
  - 15 Hz
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

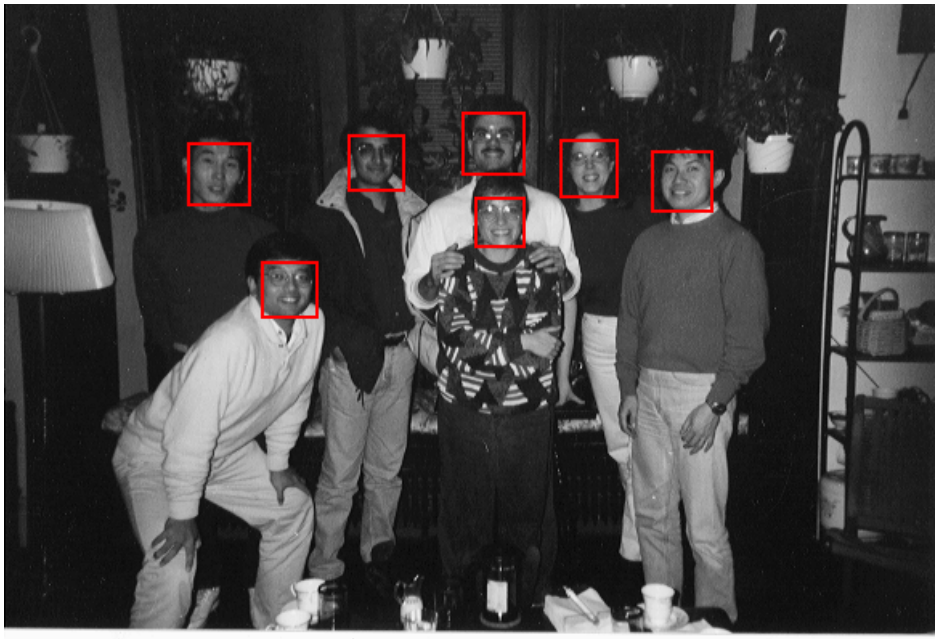
P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Detection results (2001)










MPEGMOVIE VX
W
T
MENU
DISP
HOME
SONY

Lecture outline

Recap

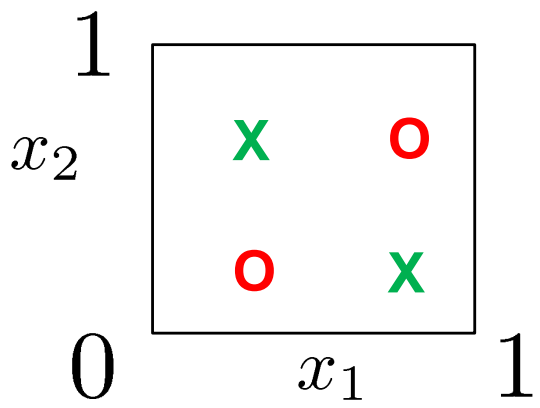
Adaboost

Decision Trees

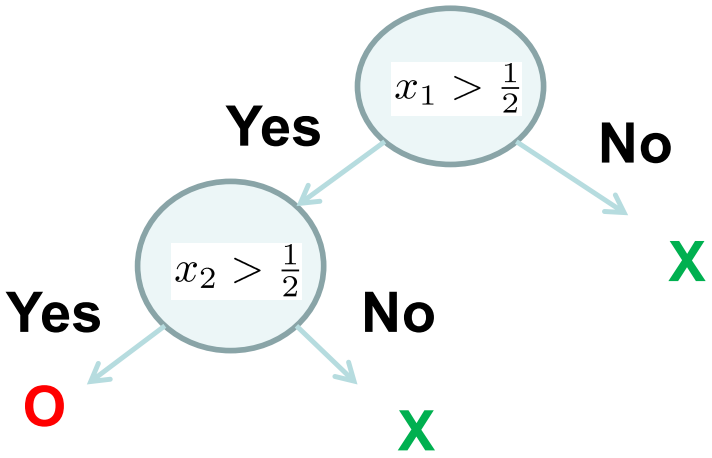
Random Forests & Ferns

Will AdaBoost always work?

- No decision stump can separate these data with error less than 1/2

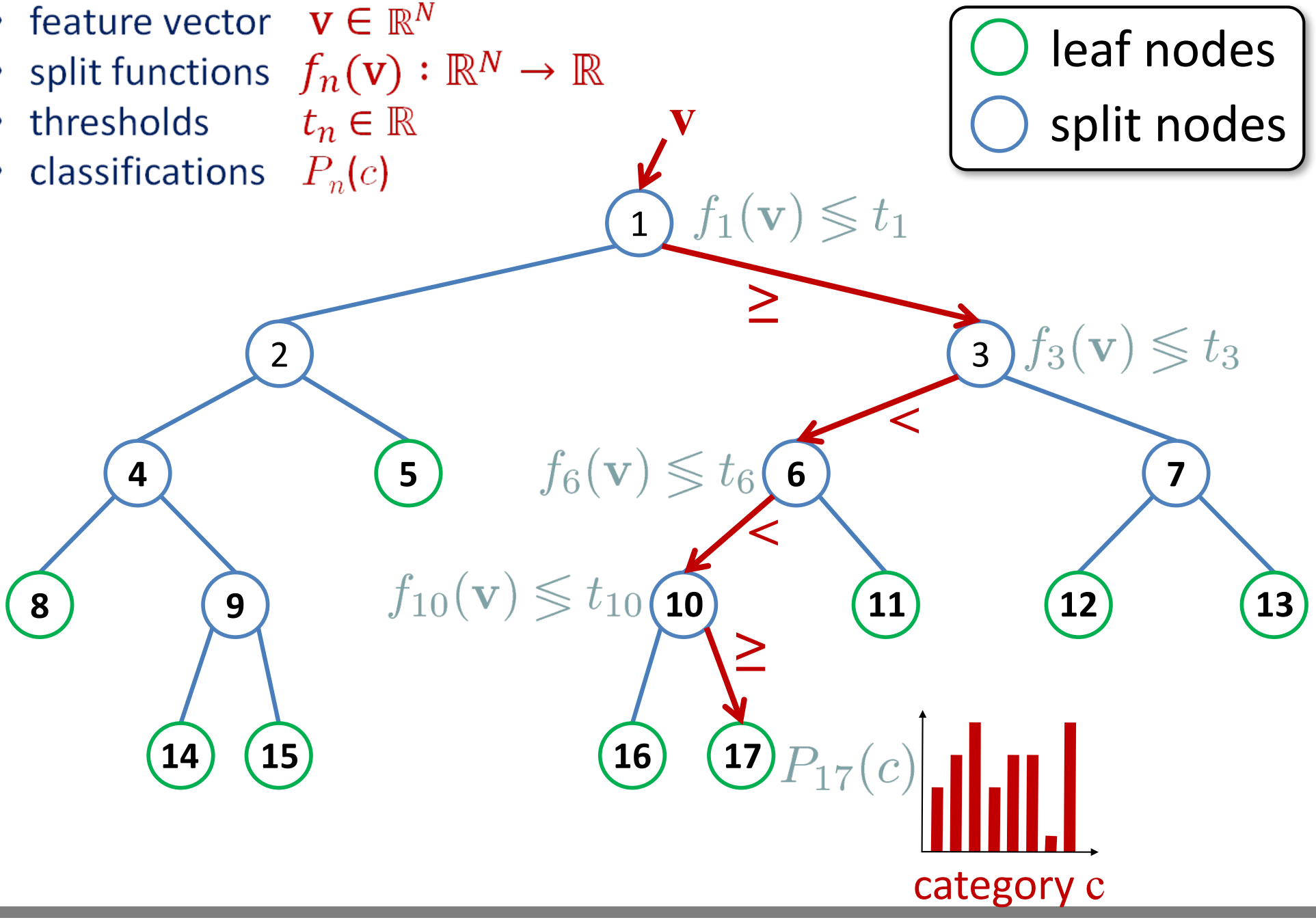


- In two steps: error 1/4

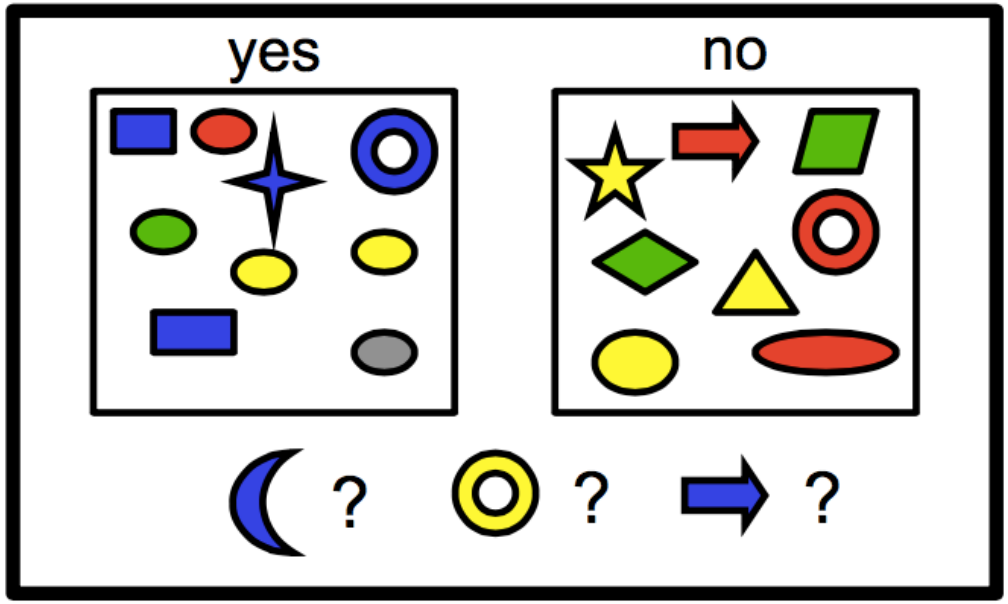


Binary Decision Trees

- feature vector $\mathbf{v} \in \mathbb{R}^N$
- split functions $f_n(\mathbf{v}) : \mathbb{R}^N \rightarrow \mathbb{R}$
- thresholds $t_n \in \mathbb{R}$
- classifications $P_n(c)$

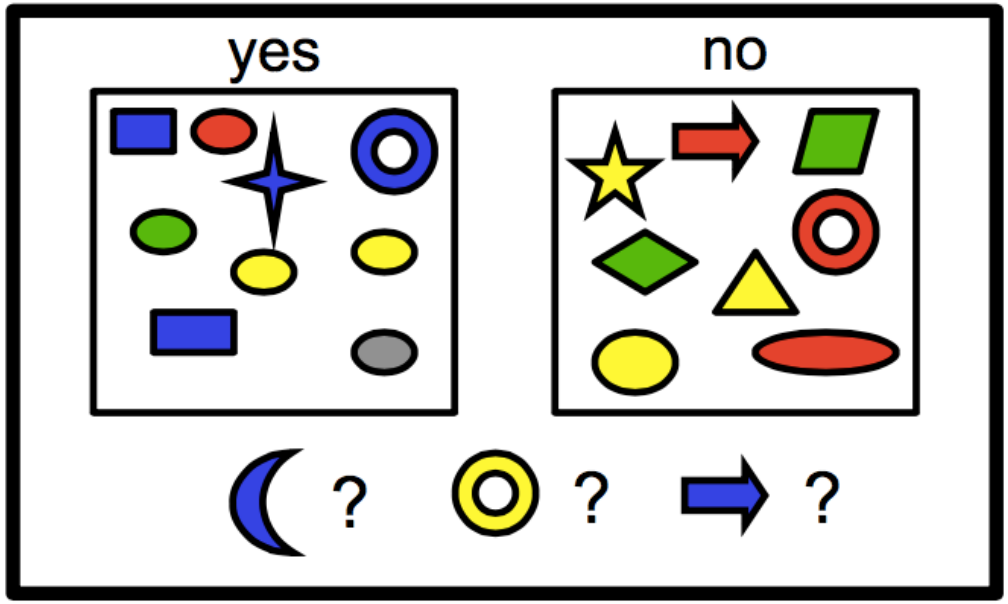


Example: Decision Trees for Classification

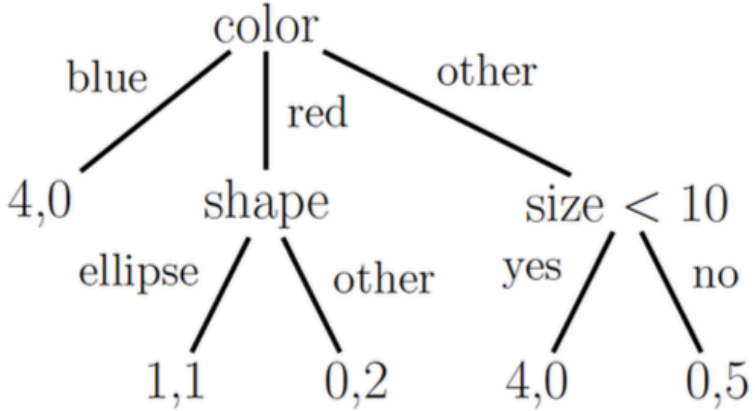


Features: color, shape, size

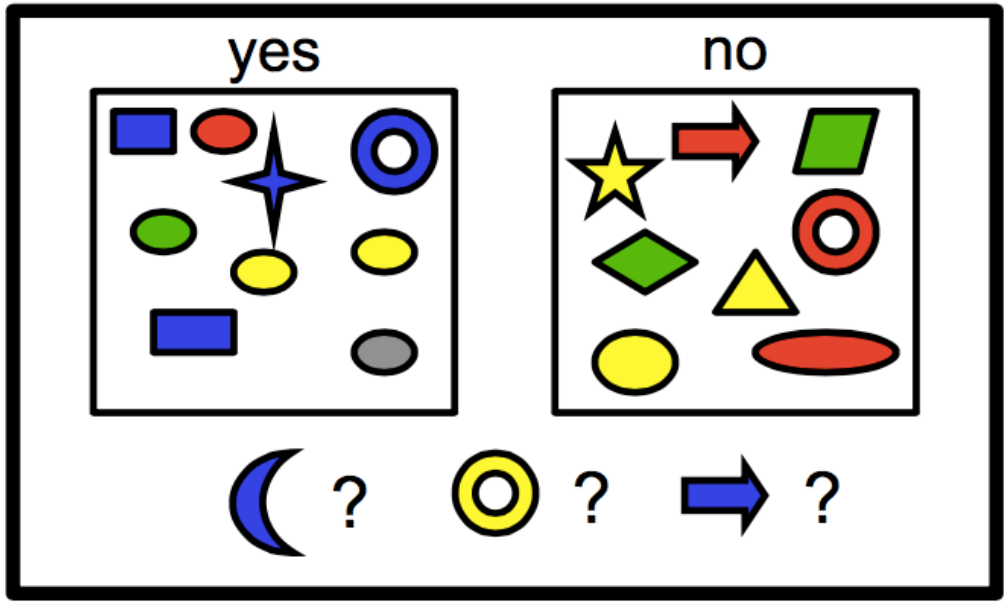
Example: Decision Trees for Classification

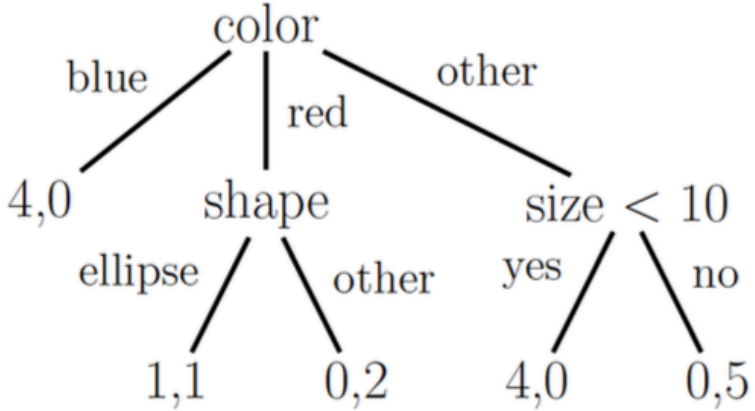


Features: color, shape, size



Example: Decision Trees for Classification

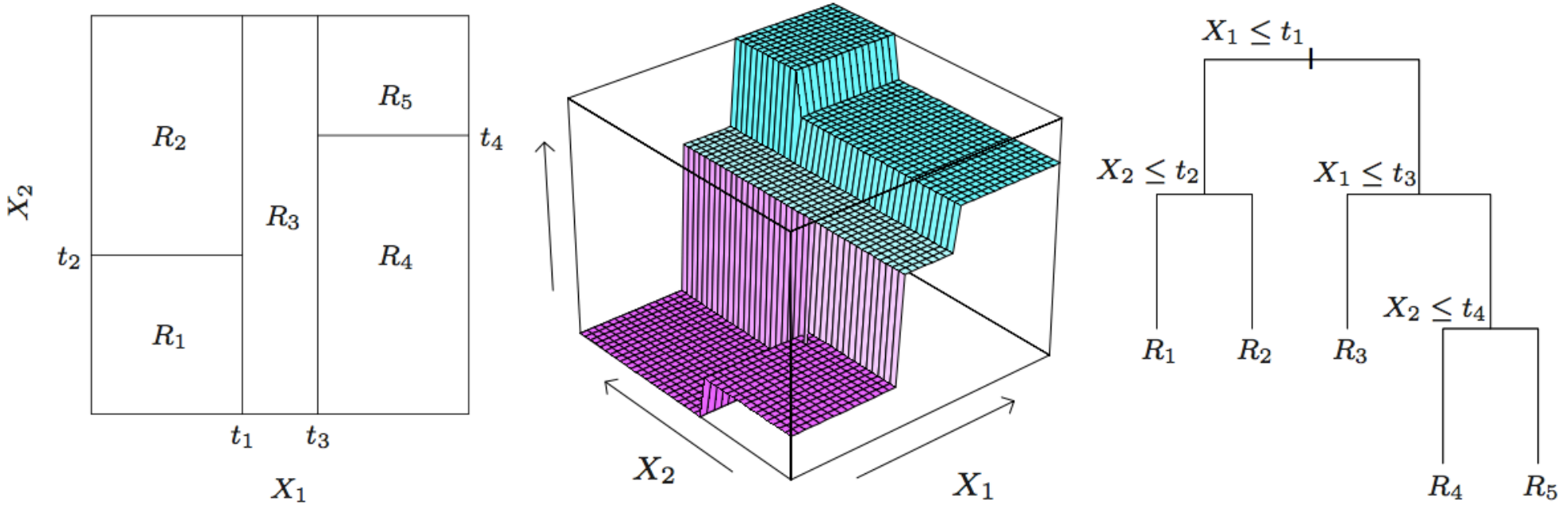




Features: color, shape, size

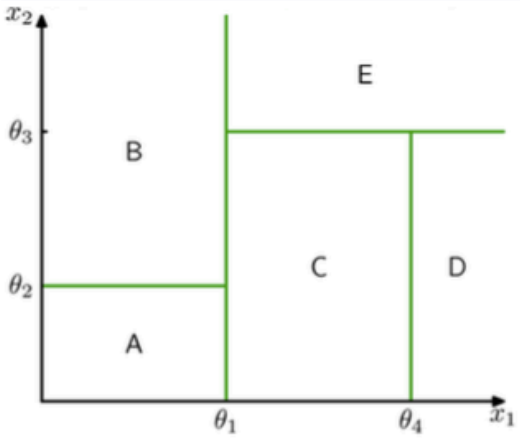
- Each node in tree splits examples according to a single feature
- Leaves have categorical distribution learned from labels of all training data whose path through tree ends there

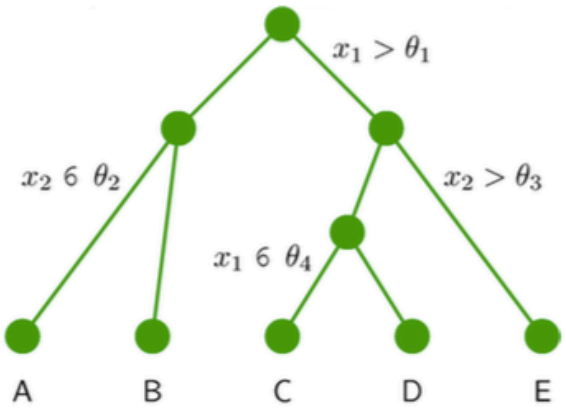
Example: Decision Trees for Regression



- Each node in tree splits examples according to a single feature
- Leaves have Gaussian distribution (mean and variance) learned from values of all training data whose path through tree ends there

Decision Tree Learning





Leaves of tree: $R_1, R_2, \ldots, R_J$.

Regression:

$$p(t_n \mid x_n \in R_j) = \mathcal{N}(t_n \mid \mu_j, \sigma_j^2)$$

Binary Classification:

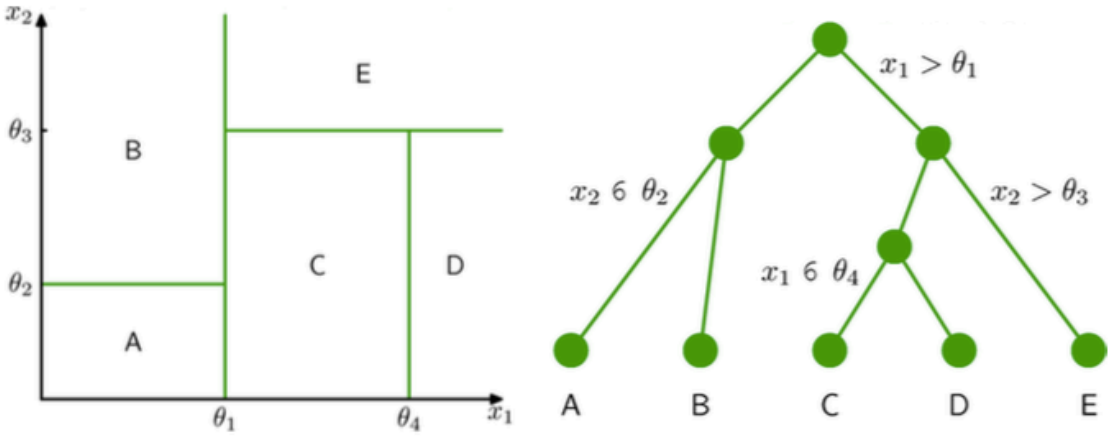
$$p(t_n \mid x_n \in R_j) = \text{Ber}(t_n \mid \mu_j)$$

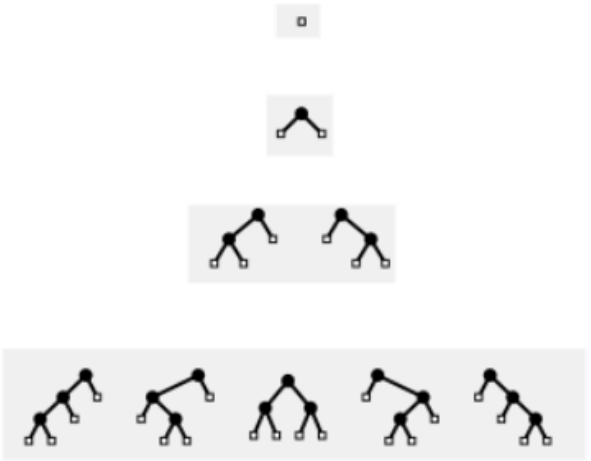
- Let *w* denote decision tree parameters: structure, splits, leaf distributions
- Conceptually, we would like to minimize the following objective:

$$f(w) = \lambda L(w) - \sum_{n=1}^{N} \log p(t_n \mid x_n, w)$$

- Here, L(w) is some measure of tree complexity:
  count of number of nodes, negative log of prior on trees, etc.

Decision Tree Learning

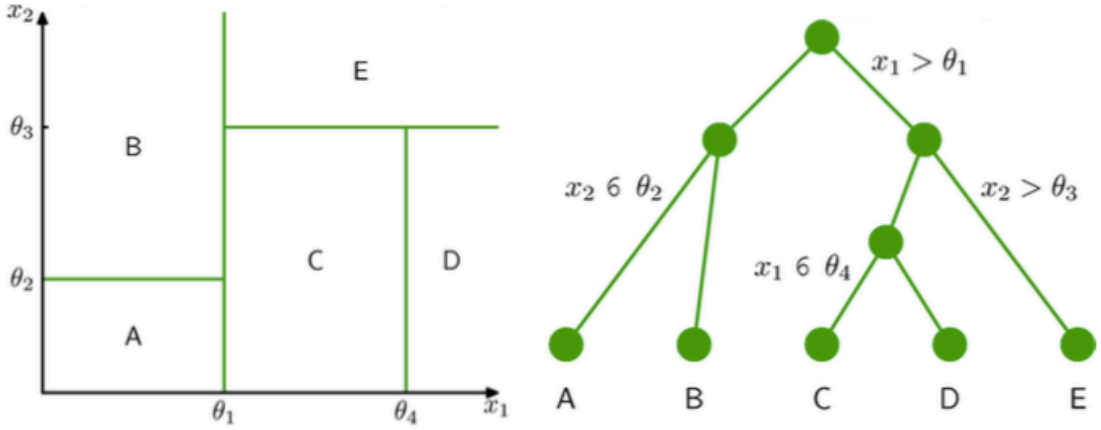




- **PROBLEM:** There are an enormous number of tree structures & split thresholds
- Optimizing leaf distributions given structure is easy, but searching structures is hard
- Classic approach: Build tree greedily, using various heuristics to control complexity, and check with validation data

CART: Classification & Regression Trees, C4.5, ID3, …

Greedy Decision Tree Learning



Leaves of tree: $R_1, R_2, \ldots, R_J$.

Regression:

$$p(t_n \mid x_n \in R_j) = \mathcal{N}(t_n \mid \mu_j, \sigma_j^2)$$

Binary Classification:

$$p(t_n \mid x_n \in R_j) = \text{Ber}(t_n \mid \mu_j)$$

N_j = number of training data for leaf j

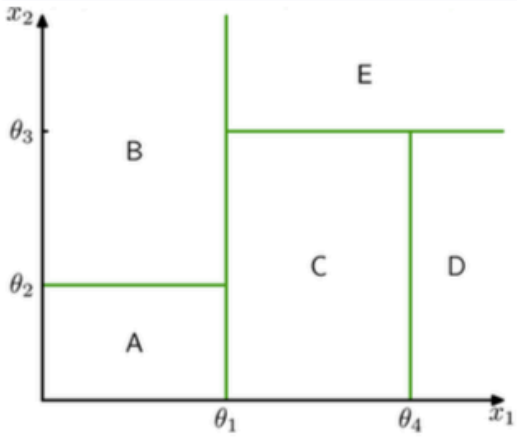
- Given a candidate tree, estimate leaf parameters via maximum likelihood:

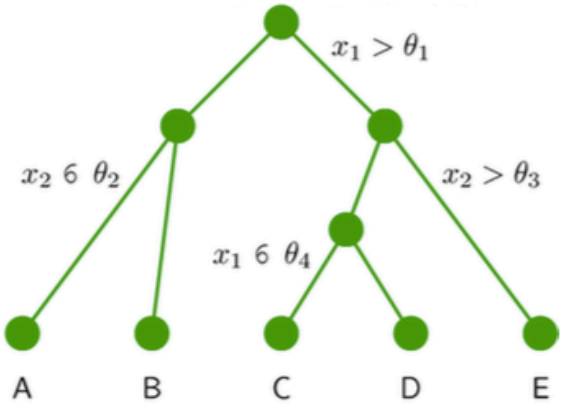
Regression: $\mu_j = \frac{1}{N_j} \sum_{n \mid x_n \in R_j} t_n \qquad \sigma_j^2 = \frac{1}{N_j} \sum_{n \mid x_n \in R_j} (t_n - \mu_j)^2$

Classification: $\mu_j = \frac{1}{N_j} \sum_{n \mid x_n \in R_j} t_n \qquad \text{Var}(t_n \mid x_n \in R_j) = \mu_j(1 - \mu_j)$

- Greedily, pick the leaf with largest variance, choose new split that minimizes sum of child variances (solvable via exhaustive search)
- Recurse to refine tree. Initialize with all data in single root node.

Controlling Decision Tree Complexity





Leaves of tree: $R_1, R_2, \ldots, R_J$.

Regression:

$$p(t_n \mid x_n \in R_j) = \mathcal{N}(t_n \mid \mu_j, \sigma_j^2)$$

Binary Classification:

$$p(t_n \mid x_n \in R_j) = \text{Ber}(t_n \mid \mu_j)$$

- **PROBLEM:** Once number of leaves in tree equals N, can set variance in each leaf to zero. This perfectly predicts each training data, but overfits!
- **Pruning:** Recursively prune leaves from tree, use validation data to check whether increases or decreases prediction accuracy
- **Random forests:** Randomly subsample data and features. Greedily fit a decision tree to each, then average their predictions. This is a form of the bootstrap that can lead to enormous gains in prediction accuracy.

Binary Decision Trees Summary

- Fast greedy training algorithms
  - Work with heterogeneous pool of features (e.g. shape, size, color,…)
- Fast testing algorithm
- Needs careful choice of hyper-parameters
  - maximum depth
  - number of features and thresholds to try
- Prone to over-fitting

Lecture outline

Recap

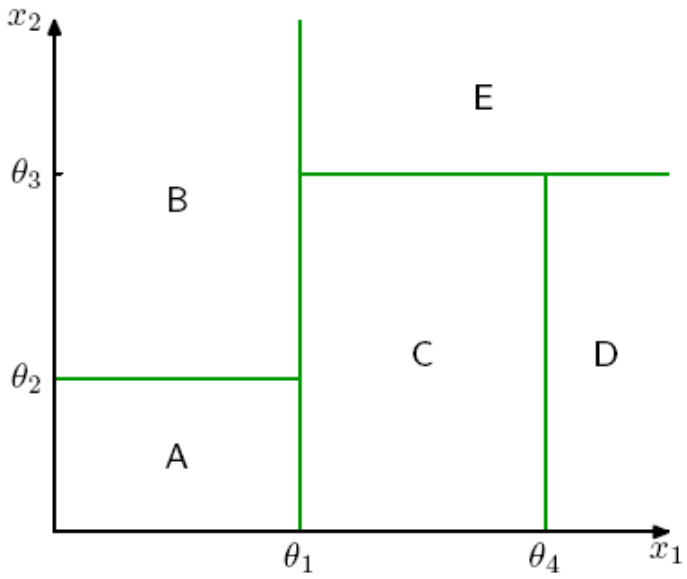
Adaboost

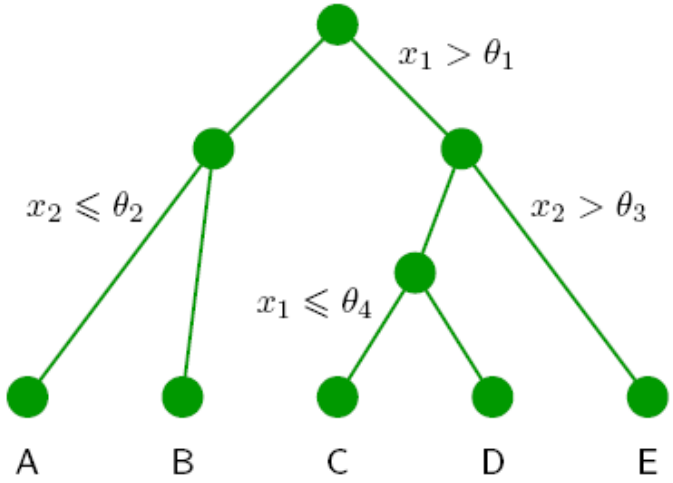
Decision Trees

Random Forests & Ferns

Which level of classification tree?

- Deeper trees result in more complex classifiers



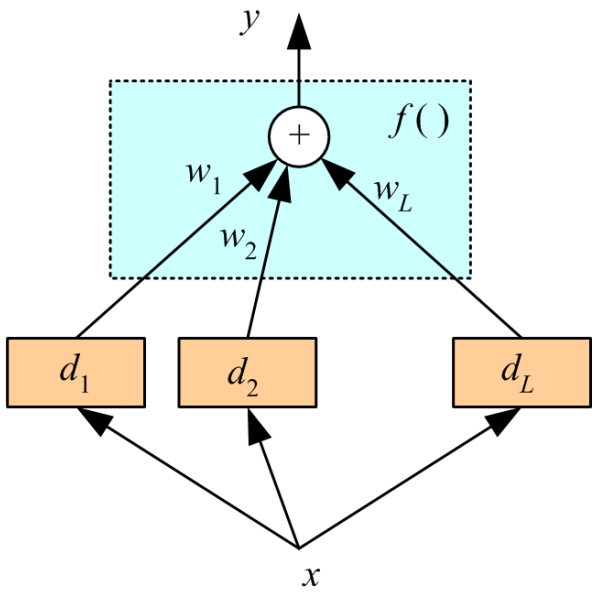


- After some point overfitting occurs

Voting Methods

- Give up idea of building `the' classifier
- Generate a group of base-learners which has higher accuracy when combined
- Main tasks
  - Generating the learners
  - Combining them

$$y_{COM}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x})$$



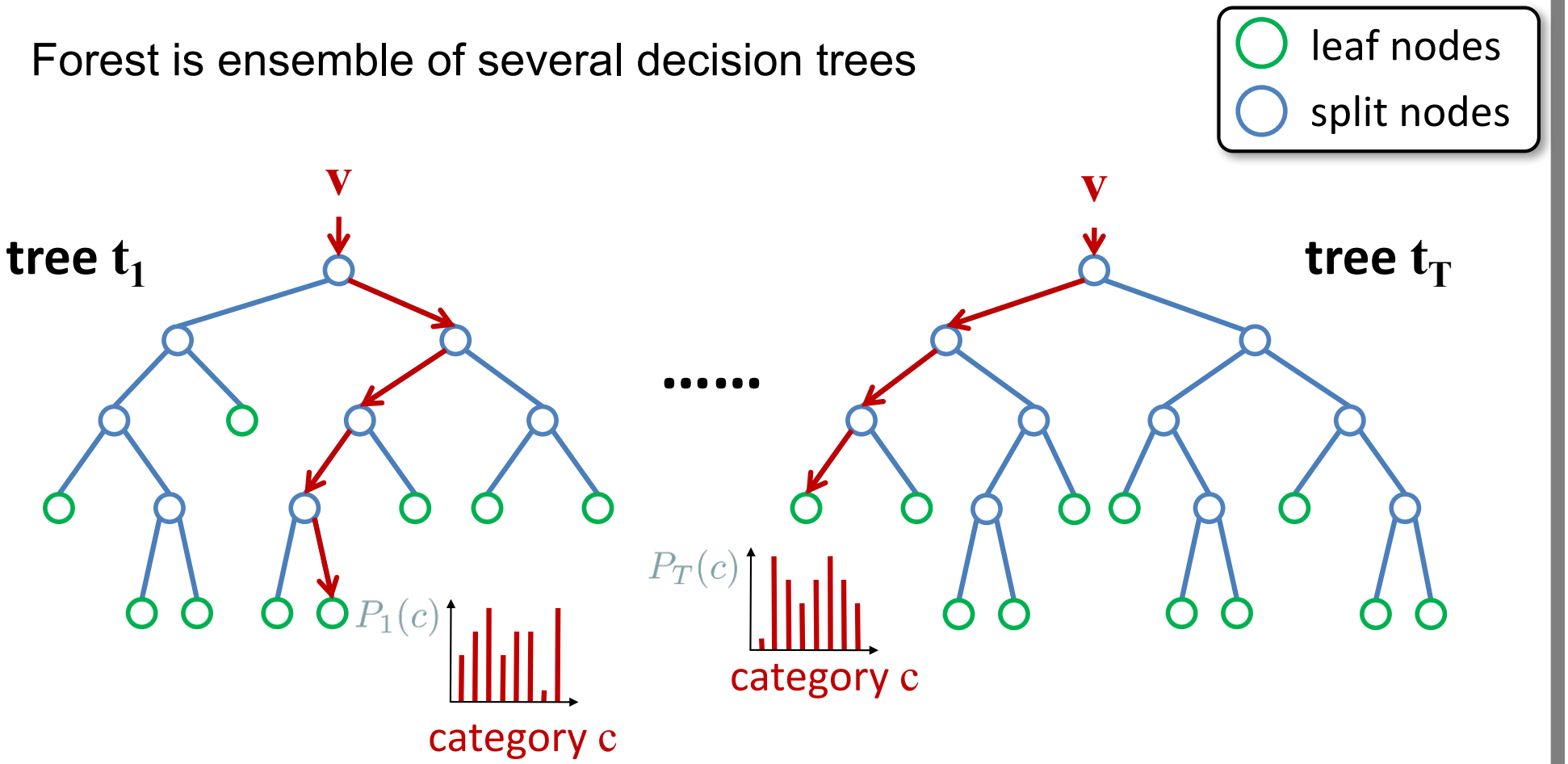
If errors have zero mean and are uncorrelated: $\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\right] = 0$

$$\mathbb{E}_{\mathbf{x}}\left[\epsilon_m(\mathbf{x})\epsilon_j(\mathbf{x})\right] = 0$$

then $\mathbb{E}_{COM} = \frac{1}{M}\mathbb{E}_{AV}$

A Forest of Trees

- Forest is ensemble of several decision trees



– classification is $$P(c|\mathbf{v}) = \frac{1}{T} \sum_{t=1}^{T} P_t(c|\mathbf{v})$$

[Amit & Geman 97]
[Breiman 01]
[Lepetit *et al.* 06]

Learning a Forest

- Divide training examples into T subsets I_t
  - **improves generalization**
  - reduces **memory requirements** & **training time**
- Train each decision tree t on subset I_t
  - same decision tree learning as before

Randomized Learning

- Recursively split examples at node n
  - set I_n indexes labeled training examples $(\mathbf{v}_i, l_i)$:

left split

$$I_l = \{i \in I_n \mid f(\mathbf{v}_i) < t\}$$

right split

$$I_r = I_n \setminus I_l$$

threshold

function of example i's feature vector

- At node n, $P_n(c)$ is histogram of example labels l_i

More Randomized Learning

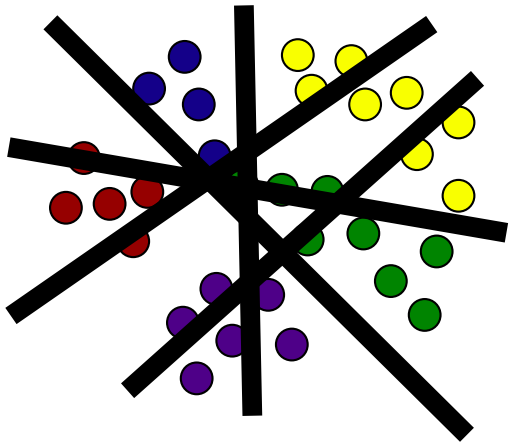
$$\begin{aligned} \text{left split } I_{\mathrm{l}} &= \{i \in I_n \mid f(\mathbf{v}_i) < t\} \\ \text{right split } I_{\mathrm{r}} &= I_n \setminus I_{\mathrm{l}} \end{aligned}$$

- Features $\mathrm{f}(\mathrm{v})$ chosen at random from feature pool $\mathrm{f} \; 2 \; \mathrm{F}$
- Thresholds t chosen in range $t \in (\min_i f(\mathbf{v}_i), \max_i f(\mathbf{v}_i))$
- Choose f and t to maximize purity criterion
- Design choices: how many thresholds? Which features? When should we stop growing?

Why do random tests work?

For a small number of classes

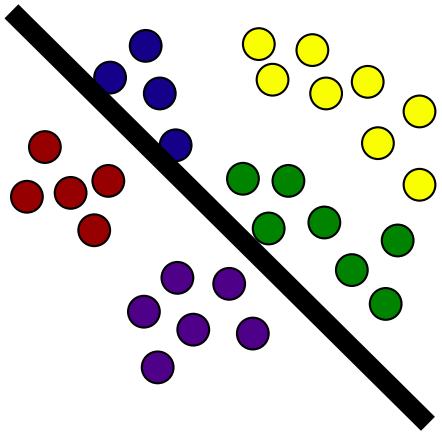
- we can try several tests, and
- retain the best one according to some criterion.



Why do random tests work?

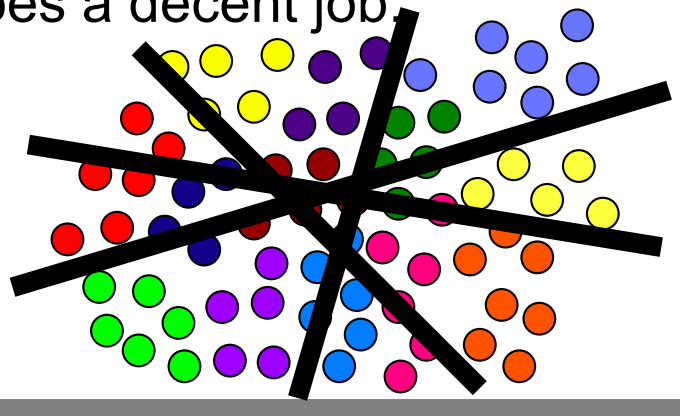
For a small number of classes

- we can try several tests, and
- retain the best one according to some criterion.



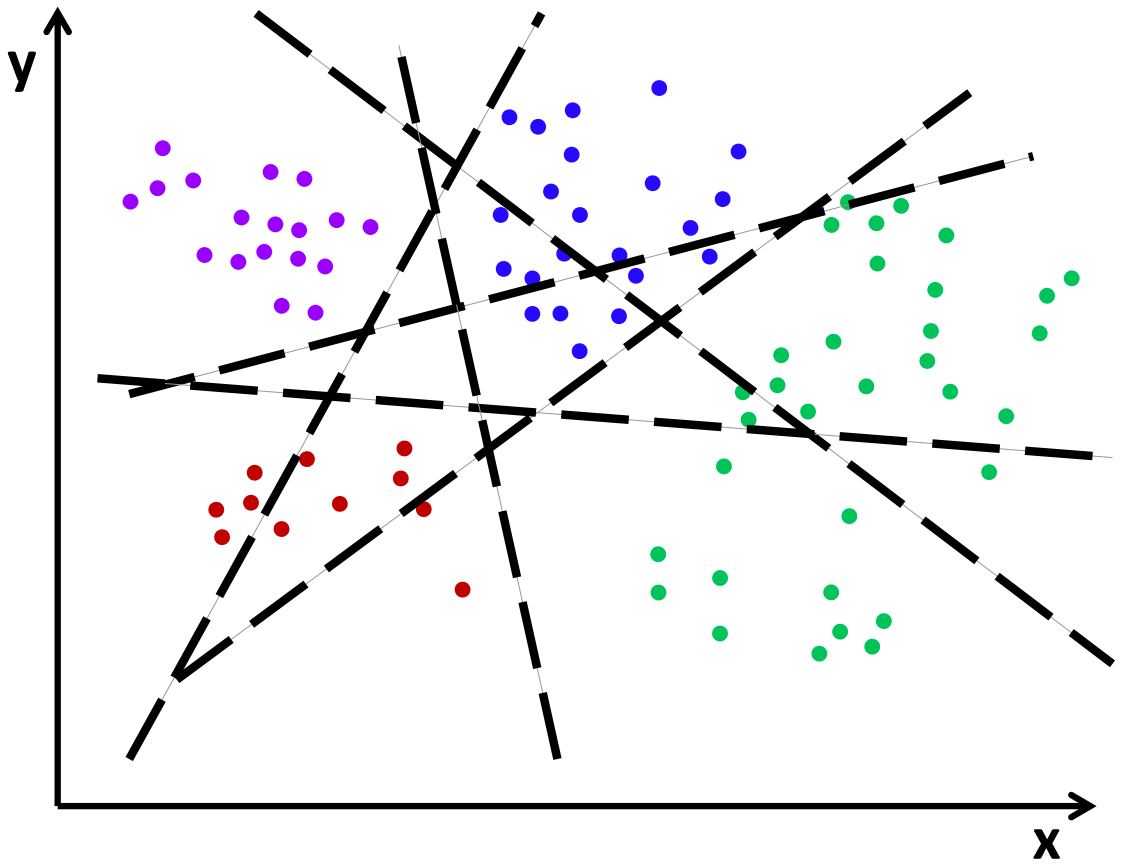
When the number of classes is large

- any test does a decent job:



Toy Learning Example

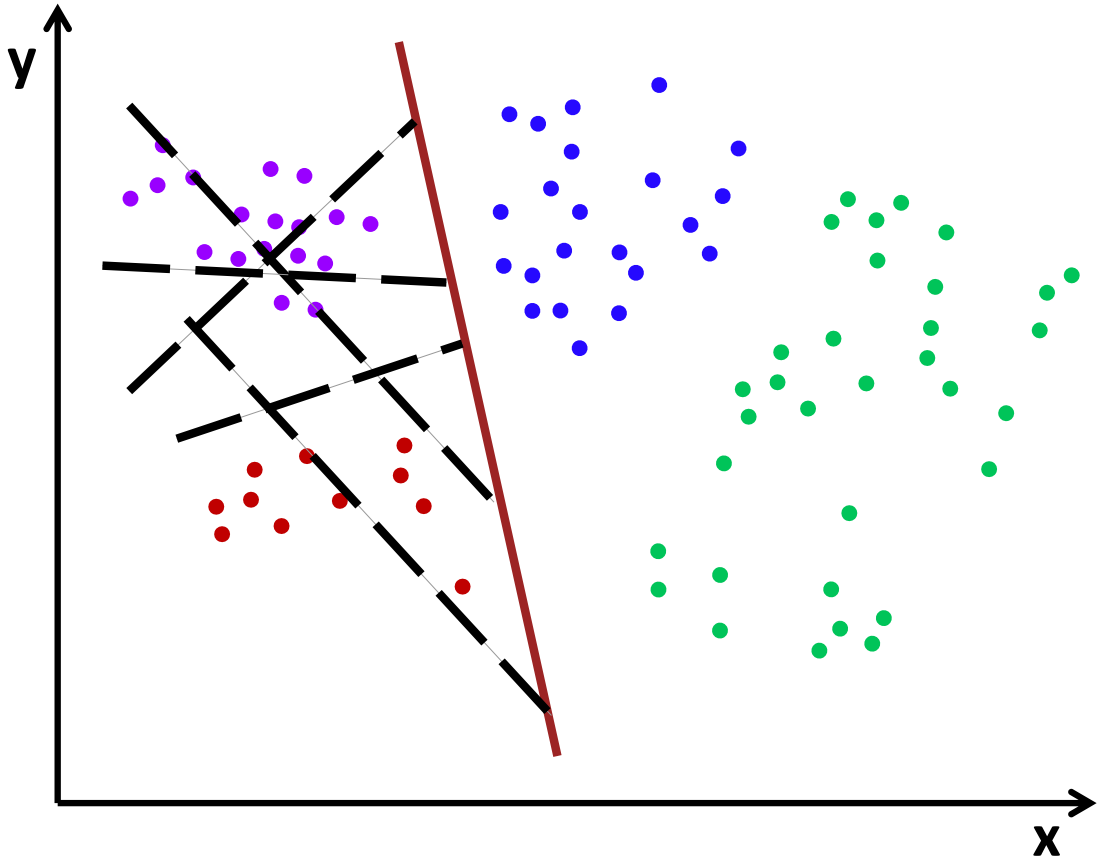
- Try several lines, chosen at random
- Keep line that best separates data
  - information gain
- Recurse



- feature vectors are x, y coordinates: $\mathbf{v} = [x, y]^T$
- split functions are lines with parameters a, b: $f_n(\mathbf{v}) = ax + by$
- threshold determines intercepts: t_n
- four classes: purple, blue, red, green

Toy Learning Example

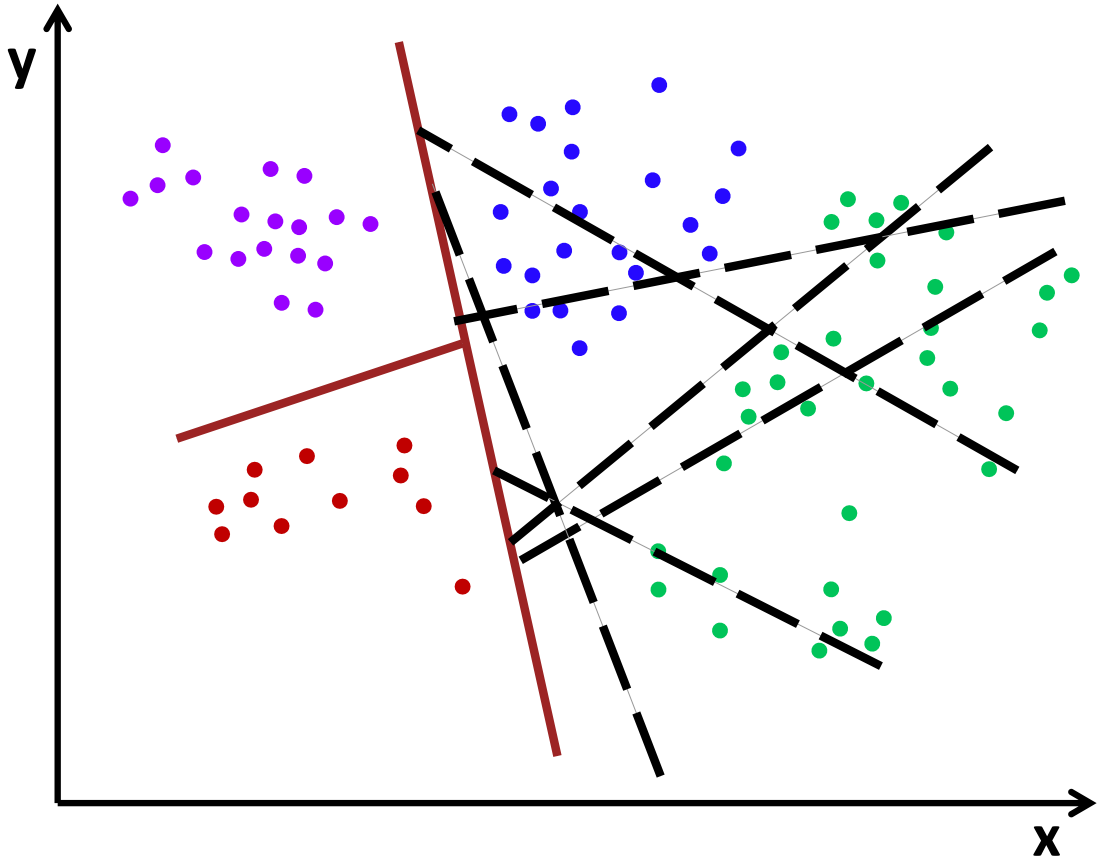
- Try several lines, chosen at random
- Keep line that best separates data
  - information gain
- Recurse



- feature vectors are x, y coordinates: $\mathbf{v} = [x, y]^T$
- split functions are lines with parameters a, b: $f_n(\mathbf{v}) = ax + by$
- threshold determines intercepts: t_n
- four classes: purple, blue, red, green

Toy Learning Example

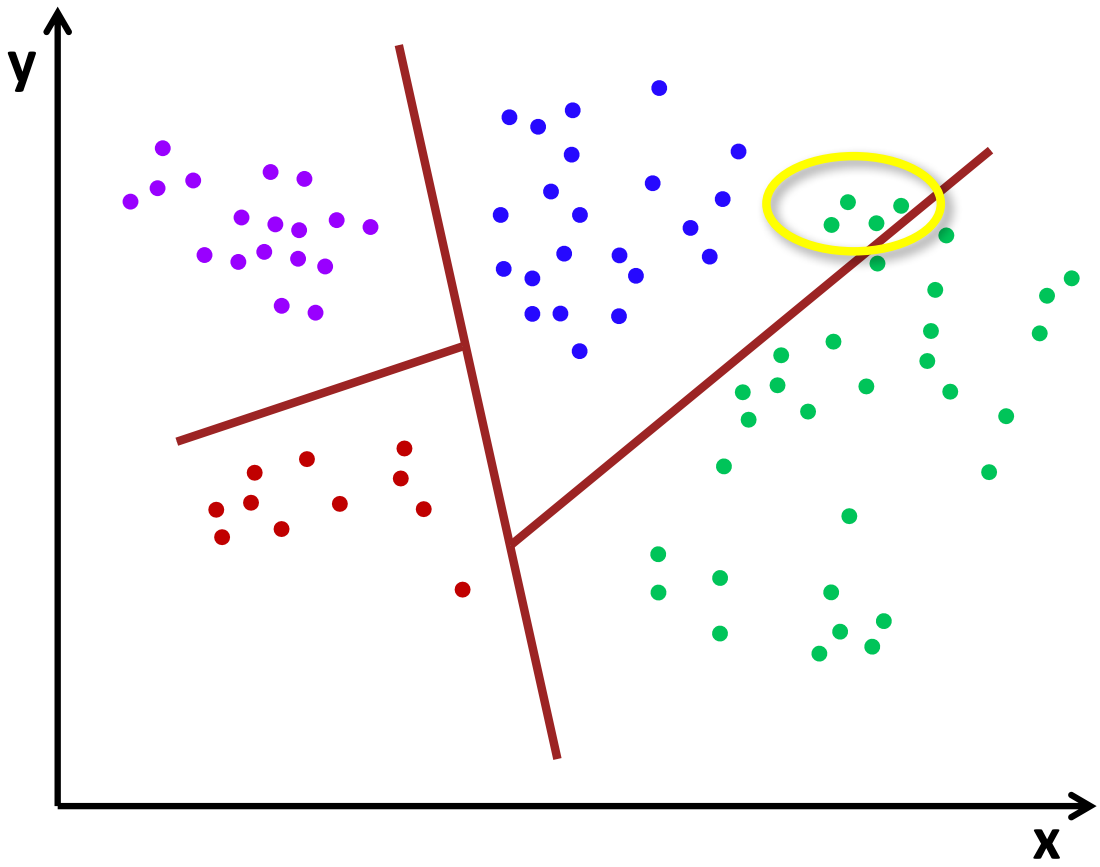
- Try several lines, chosen at random
- Keep line that best separates data
  - information gain
- Recurse



- feature vectors are x, y coordinates: $\mathbf{v} = [x, y]^T$
- split functions are lines with parameters a, b: $f_n(\mathbf{v}) = ax + by$
- threshold determines intercepts: t_n
- four classes: purple, blue, red, green

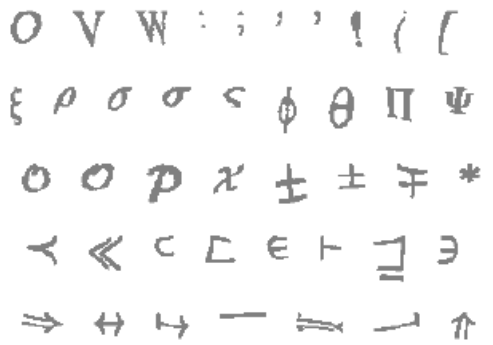
Toy Learning Example

- Try several lines, chosen at random
- Keep line that best separates data
  - information gain
- Recurse



- **feature vectors are x, y coordinates:** $\mathbf{v} = [x, y]^T$
- **split functions are lines with parameters a, b:** $f_n(\mathbf{v}) = ax + by$
- **threshold determines intercepts:** t_n
- **four classes: purple, blue, red, green**

Randomized Forests in Vision

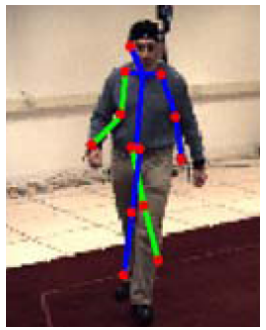


[Amit & Geman, 97]
digit recognition

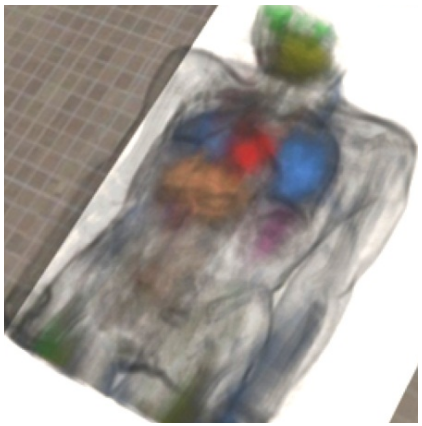
[Lepetit *et al.*, 06]
keypoint recognition

[Moosmann *et al.*, 06]
visual word clustering

[Shotton *et al.*, 08]
object segmentation



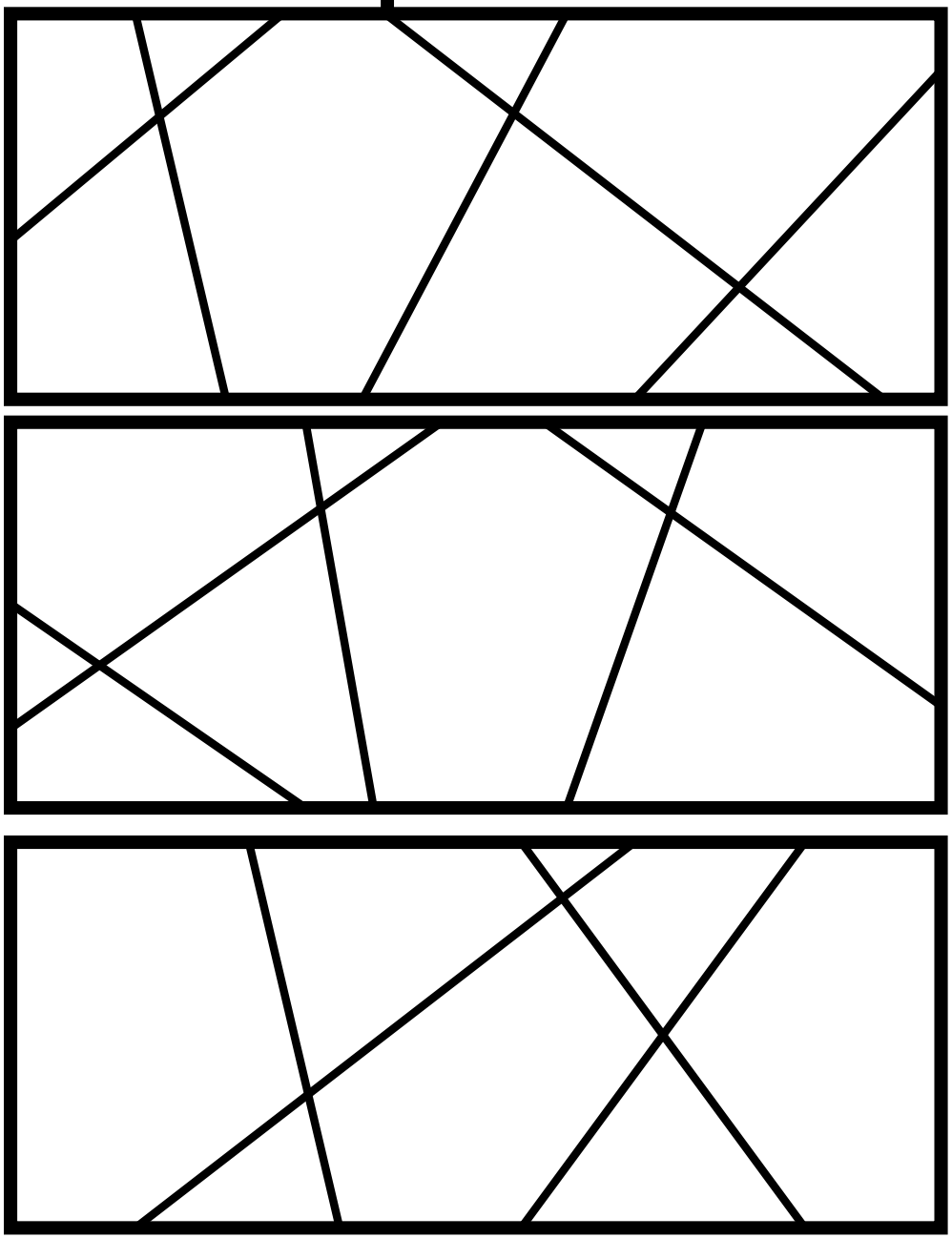
[Rogez *et al.*, 08]
pose estimation



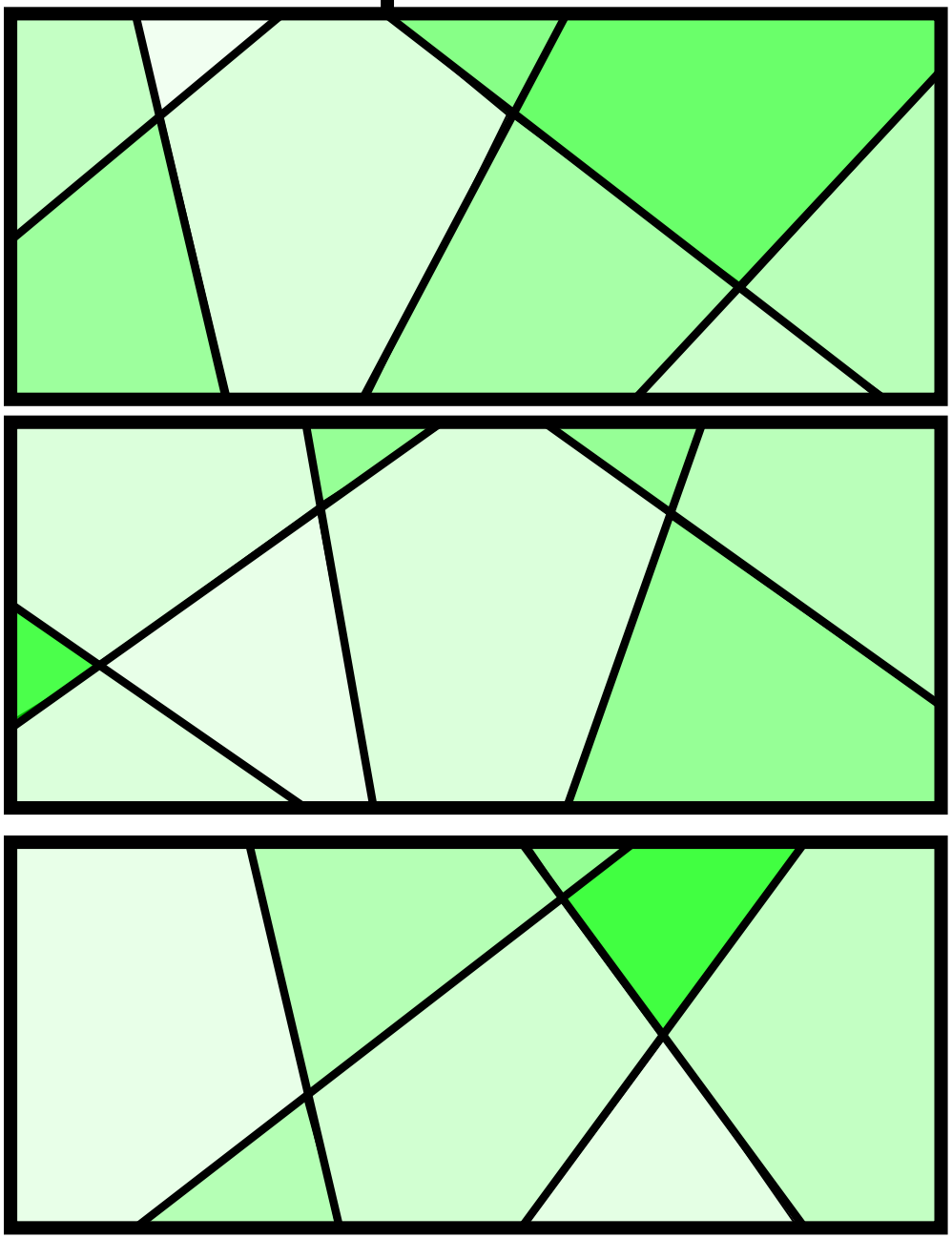
[Criminisi *et al.*, 09]
organ detection

(Among many others...)

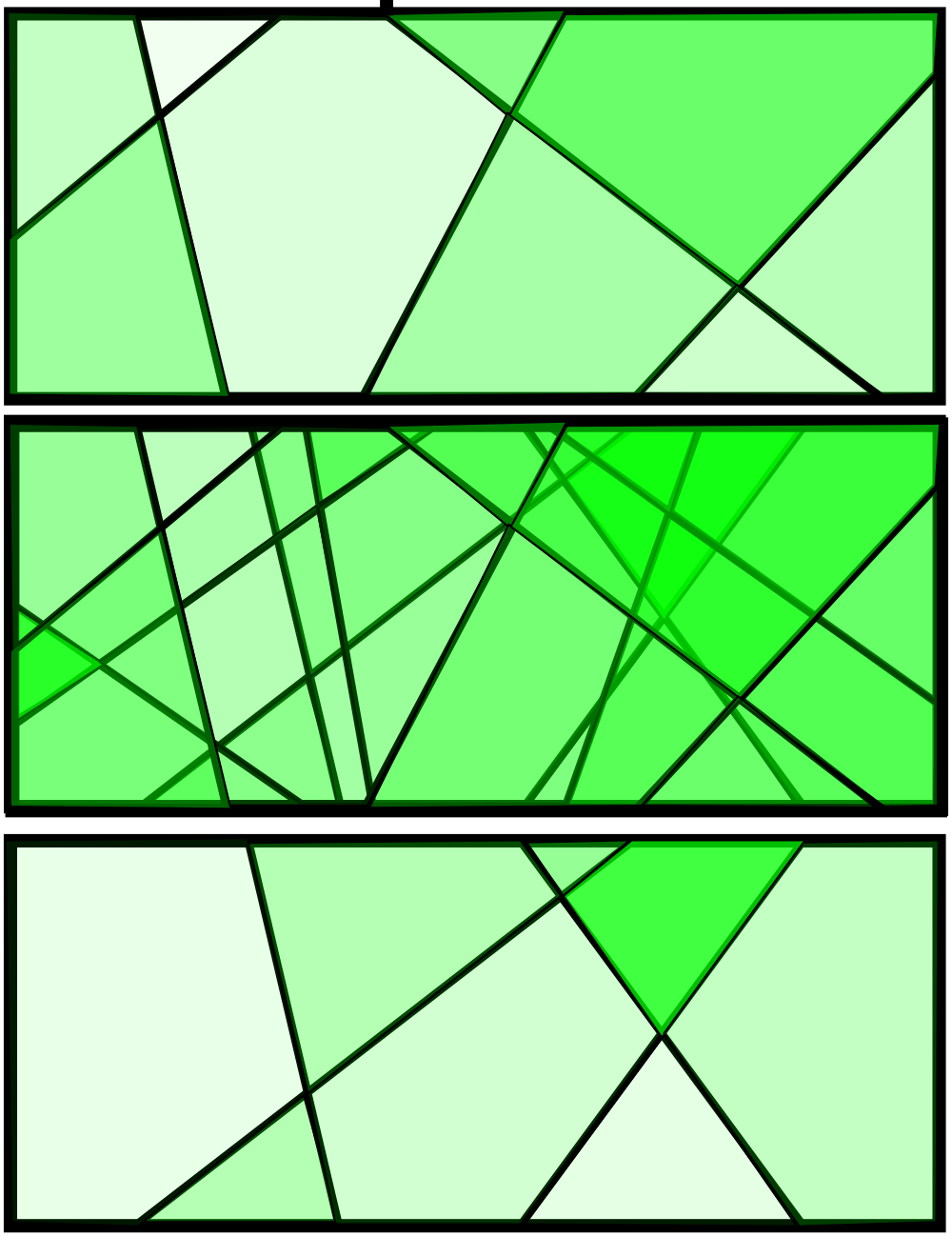
A Graphical Interpretation



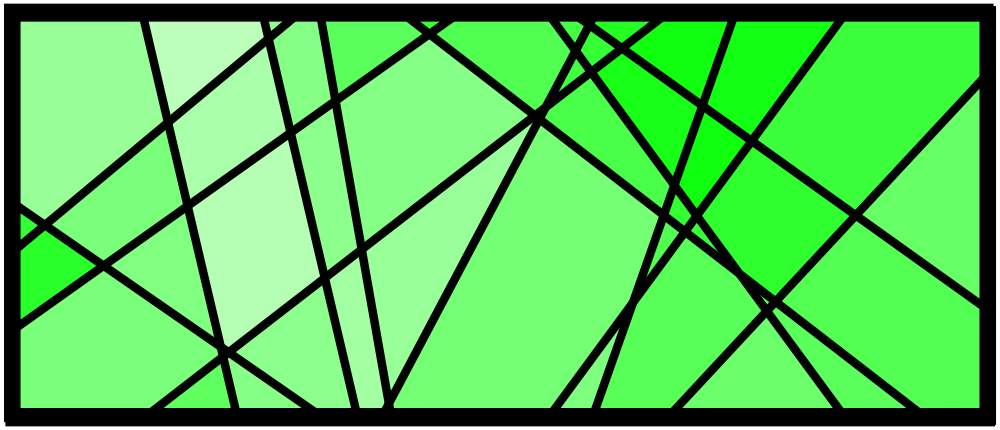
A Graphical Interpretation



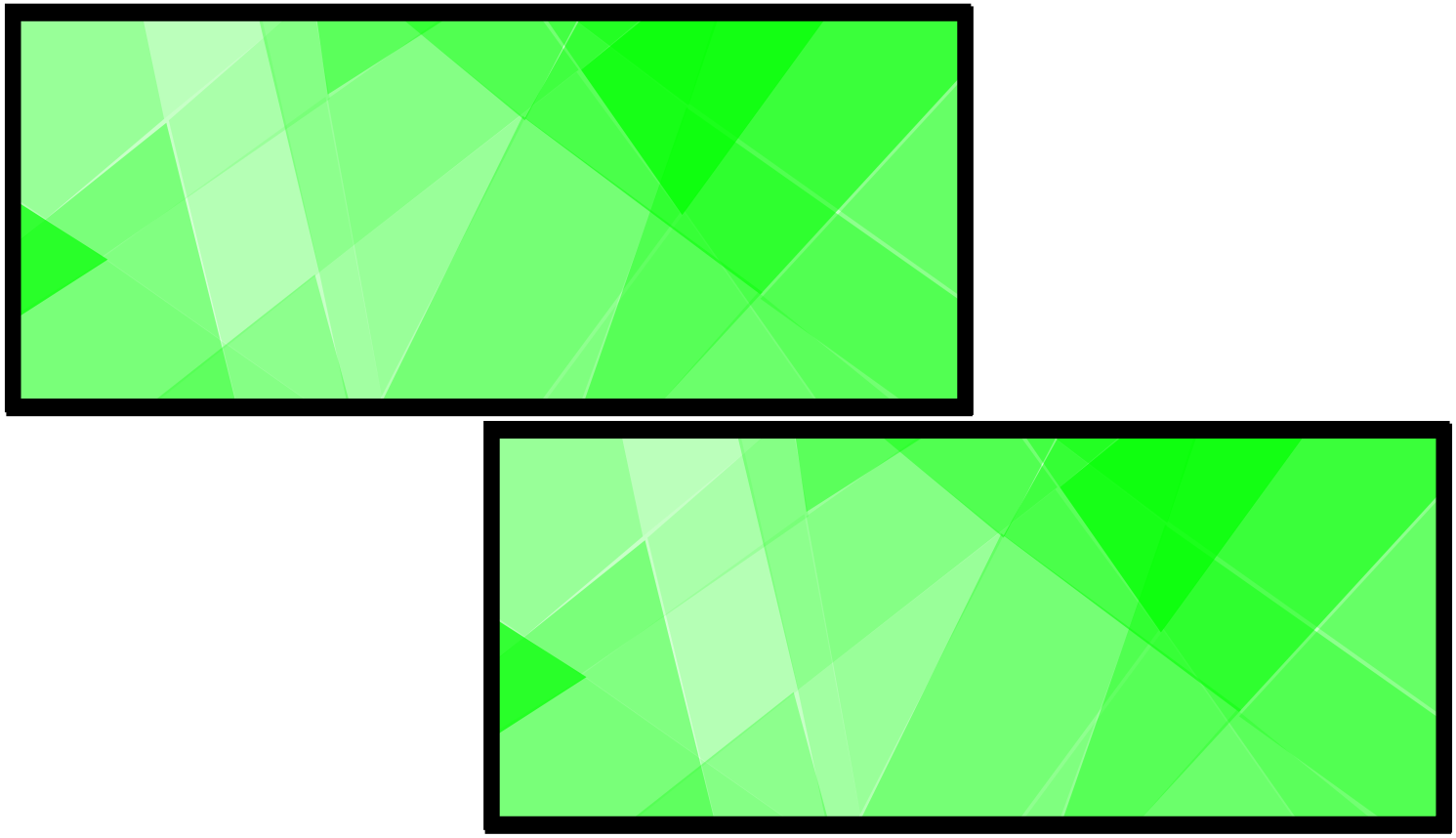
A Graphical Interpretation



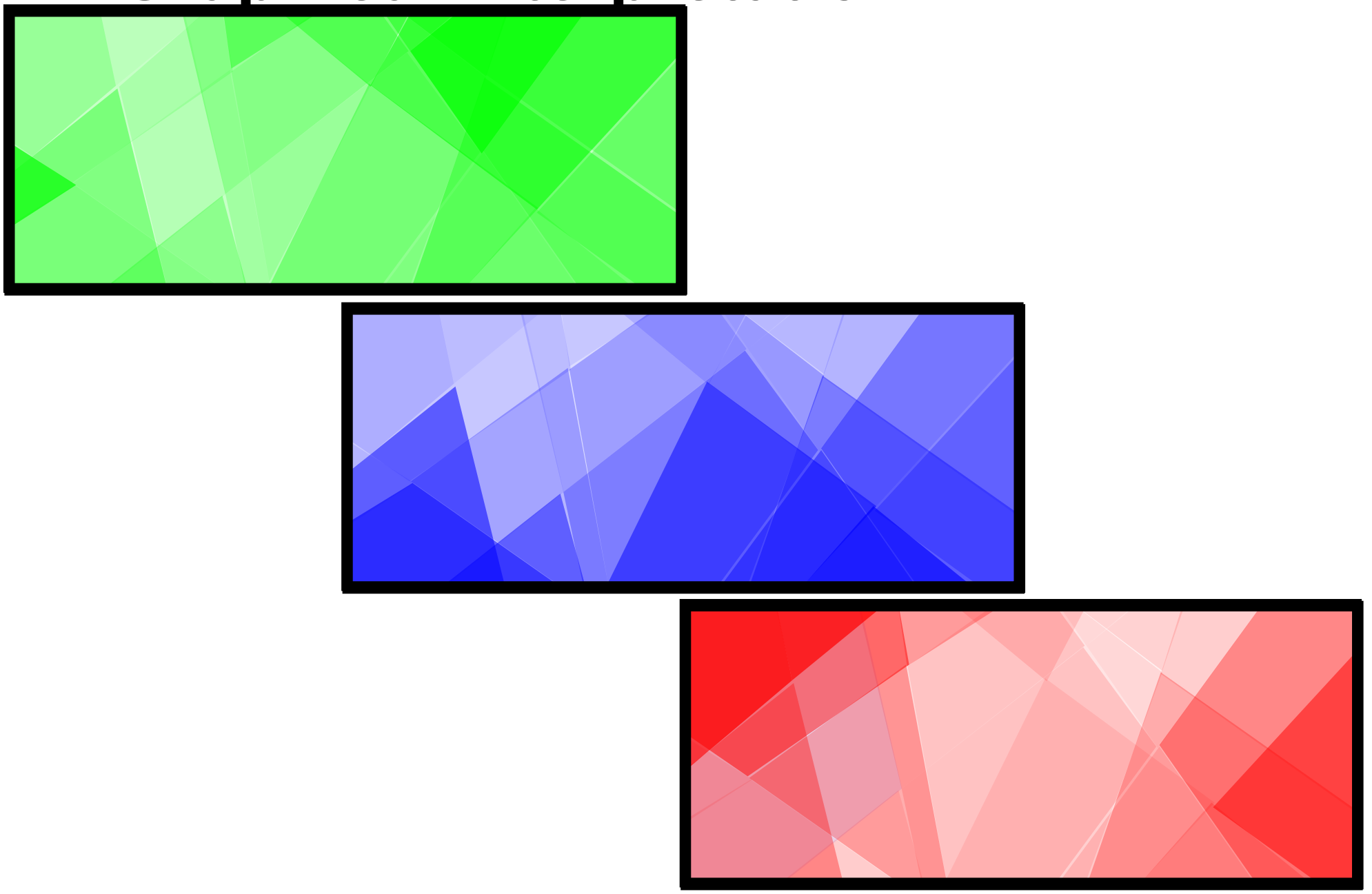
A Graphical Interpretation



A Graphical Interpretation



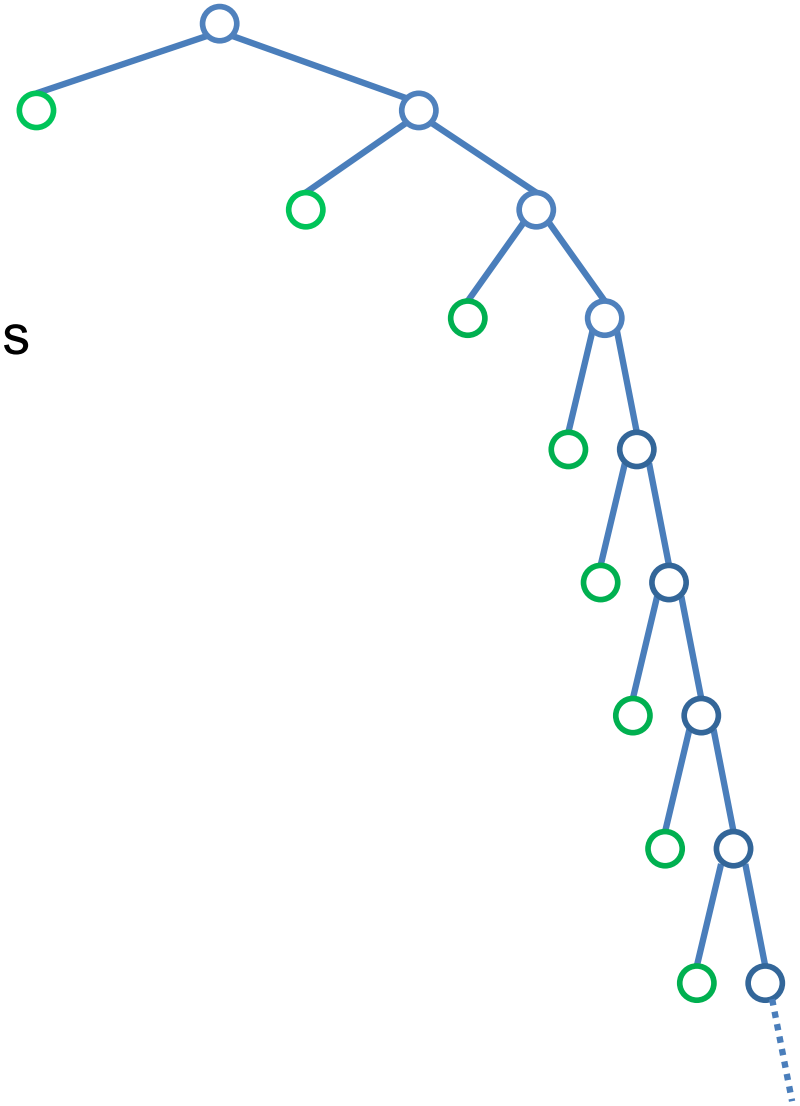
A Graphical Interpretation



Relation to Cascades [Viola & Jones 04]

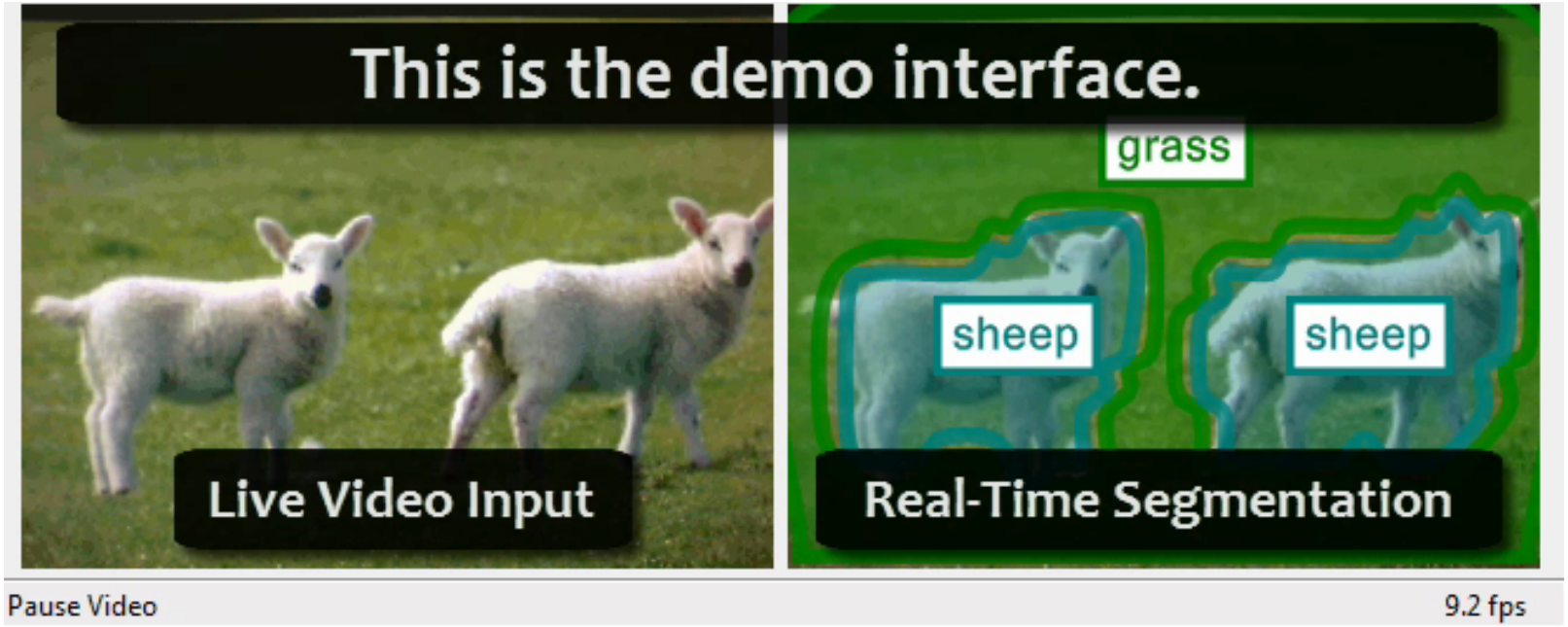
- Boosted Cascades
  - very unbalanced tree
  - good for unbalanced binary problems
    e.g. sliding window object detection

- Randomized forests
  - less deep, fairly balanced
  - ensemble of trees gives robustness
  - good for multi-class problems



Real-Time Object Segmentation [Shotton *et al.* 2008]

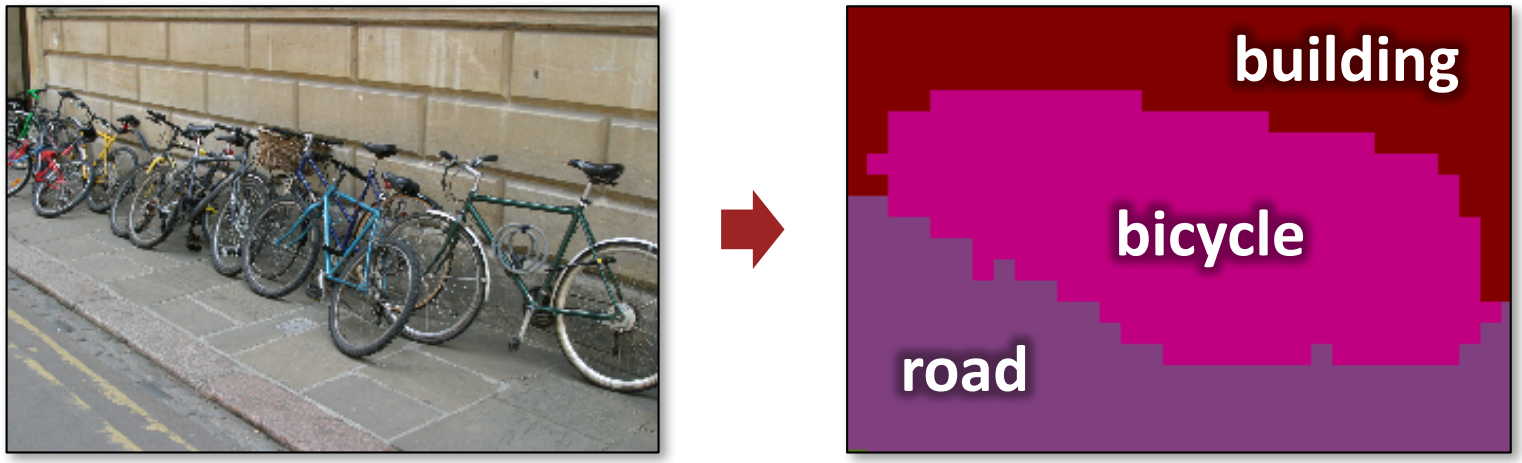
- Segment image and label segments in real-time



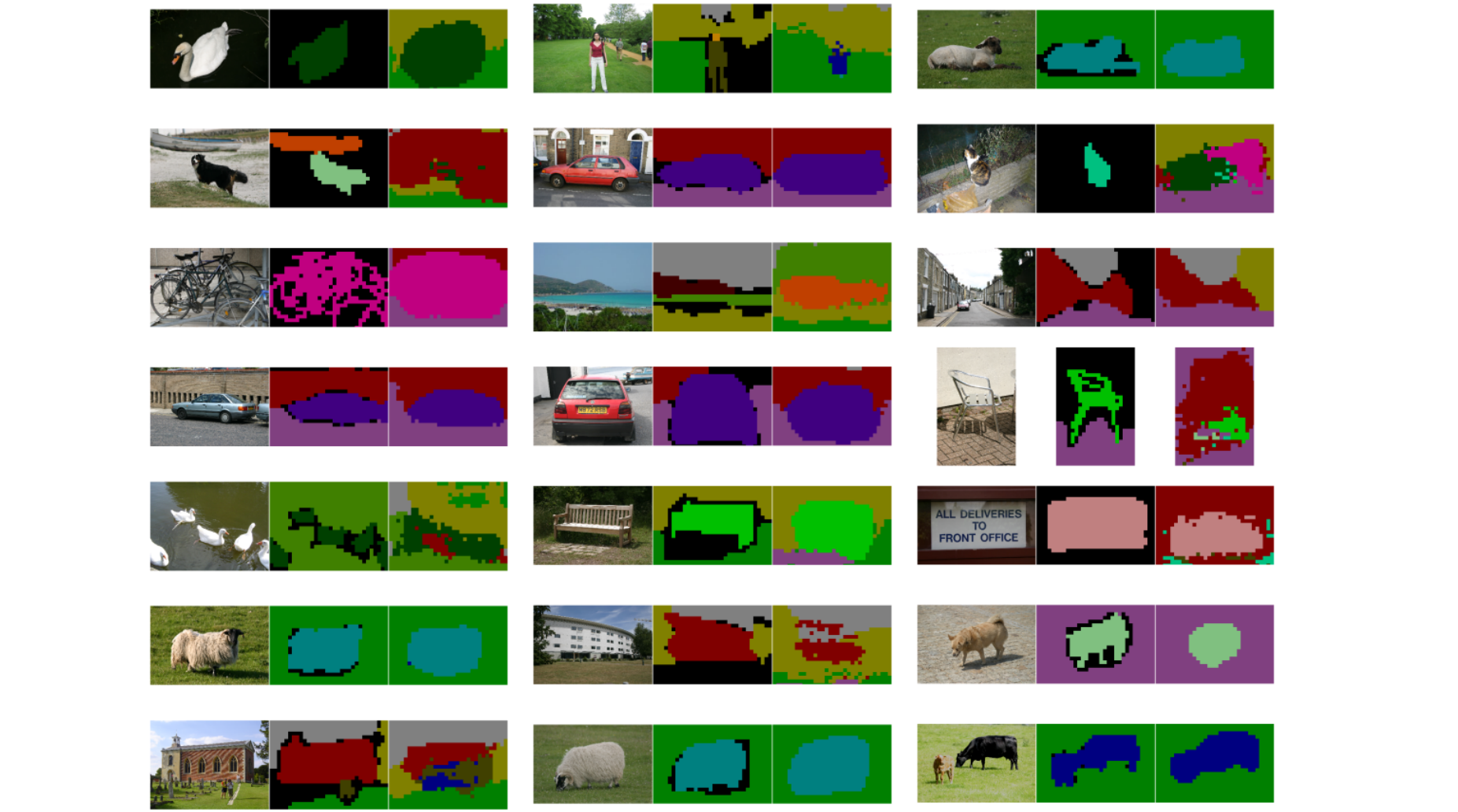
CVPR 2008 Best Demo Award

Segmentation Forest

- Object segmentation



MSRC Dataset Results



building	grass	tree	cow	sheep	sky	airplane	water	face	car	boat
bicycle	flower	sign	bird	book	chair	road	cat	dog	body	

$UCCE$$ $TORY: Kinect

12.1 million sold by end 2011

GUINNESS WORLD RECORDS™

Kinect's pipeline

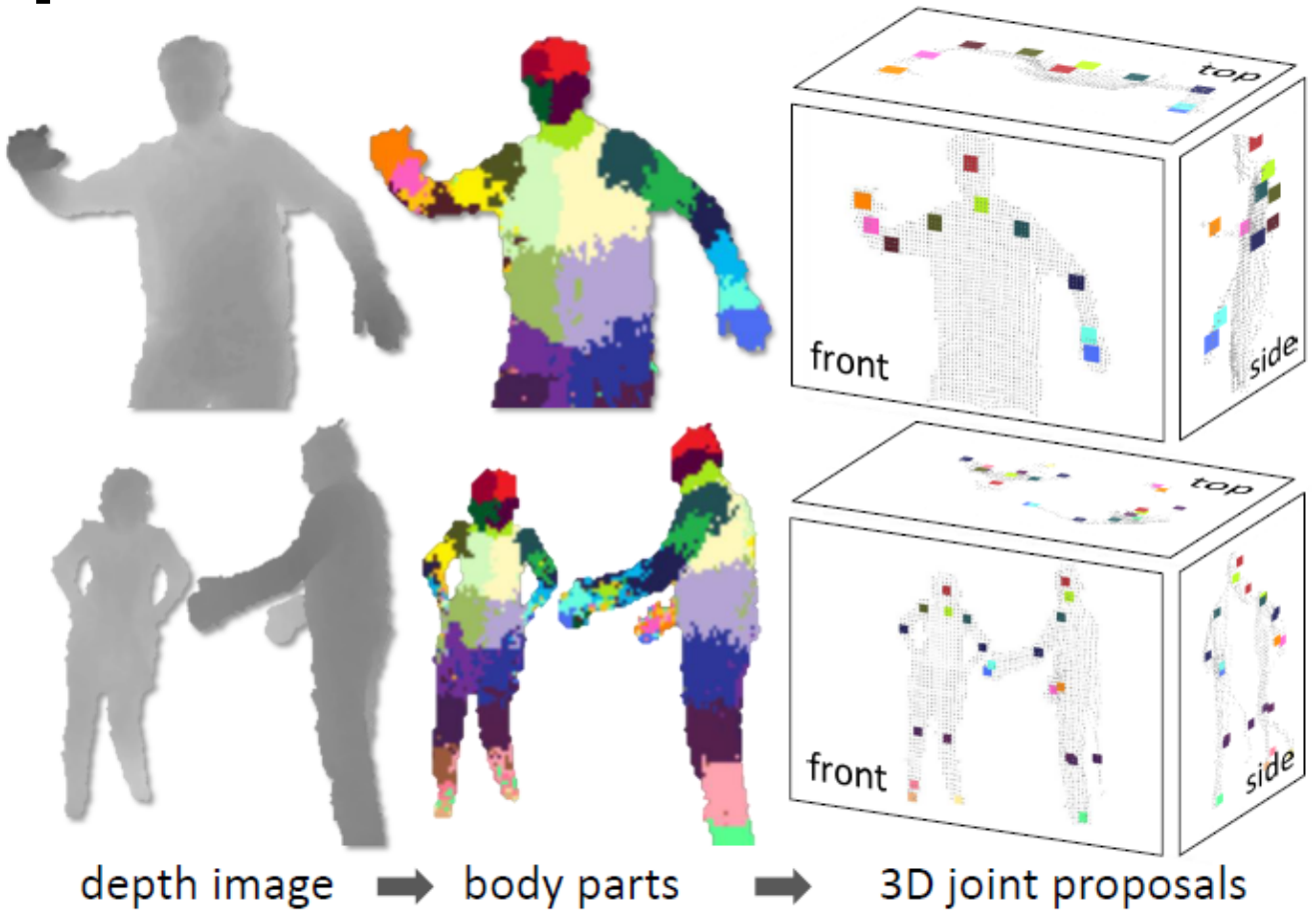
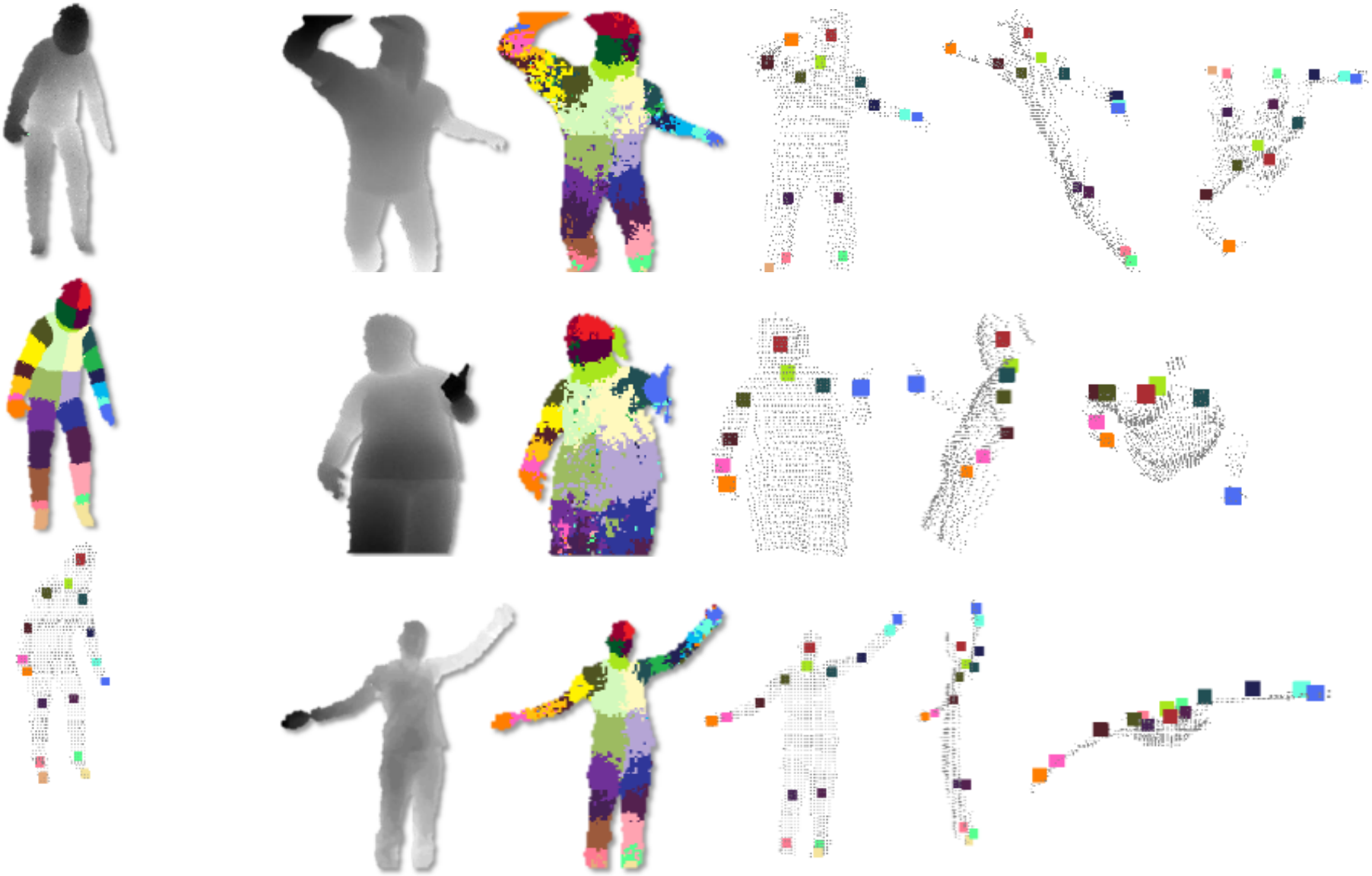


Figure 1. **Overview.** From an single input depth image, a per-pixel body part distribution is inferred. (Colors indicate the most likely part labels at each pixel, and correspond in the joint proposals). Local modes of this signal are estimated to give high-quality proposals for the 3D locations of body joints, even for multiple users.

- Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake, Real-Time Human Pose Recognition in Parts from a Single Depth Image, in *CVPR*, IEEE, June 2011

- **Abstract:** We propose a new method to quickly and accurately predict 3D positions of body joints from a single depth image, using no temporal information. We take an object recognition approach, designing an intermediate body parts representation that **maps the difficult pose estimation problem into a simpler per-pixel classification problem.** Our **large and highly varied training dataset** allows the classifier to estimate body parts invariant to pose, body shape, clothing, etc. Finally we generate confidence-scored 3D proposals of several body joints by reprojecting the classification result and finding local modes.
- **The system runs at 200 frames per second on consumer hardware**. Our evaluation shows high accuracy on both synthetic and real test sets, and investigates the effect of several training parameters. We achieve state of the art accuracy in our comparison with related work and demonstrate improved generalization over exact whole-skeleton nearest neighbor matching.



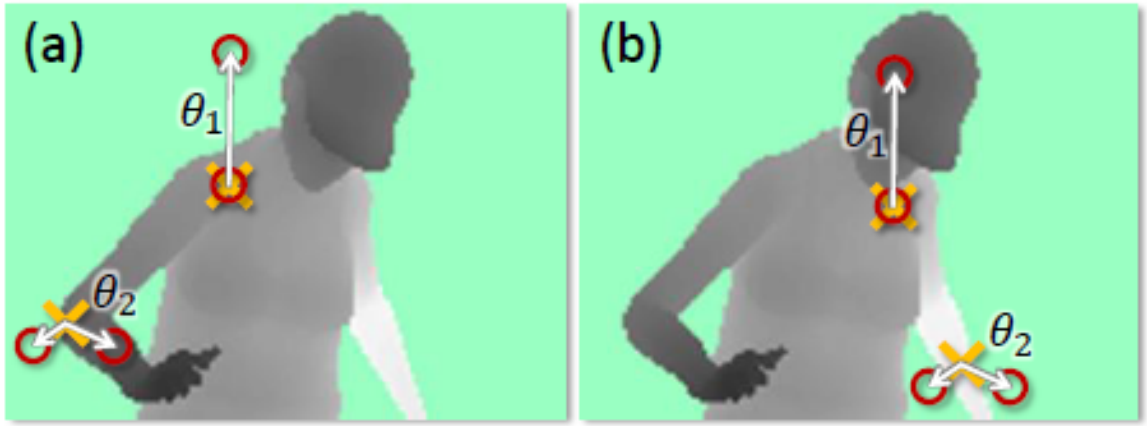


Figure 3. **Depth image features.** The yellow crosses indicates the pixel x being classified. The red circles indicate the offset pixels as defined in Eq. 1. In (a), the two example features give a large depth difference response. In (b), the same two features at new image locations give a much smaller response.

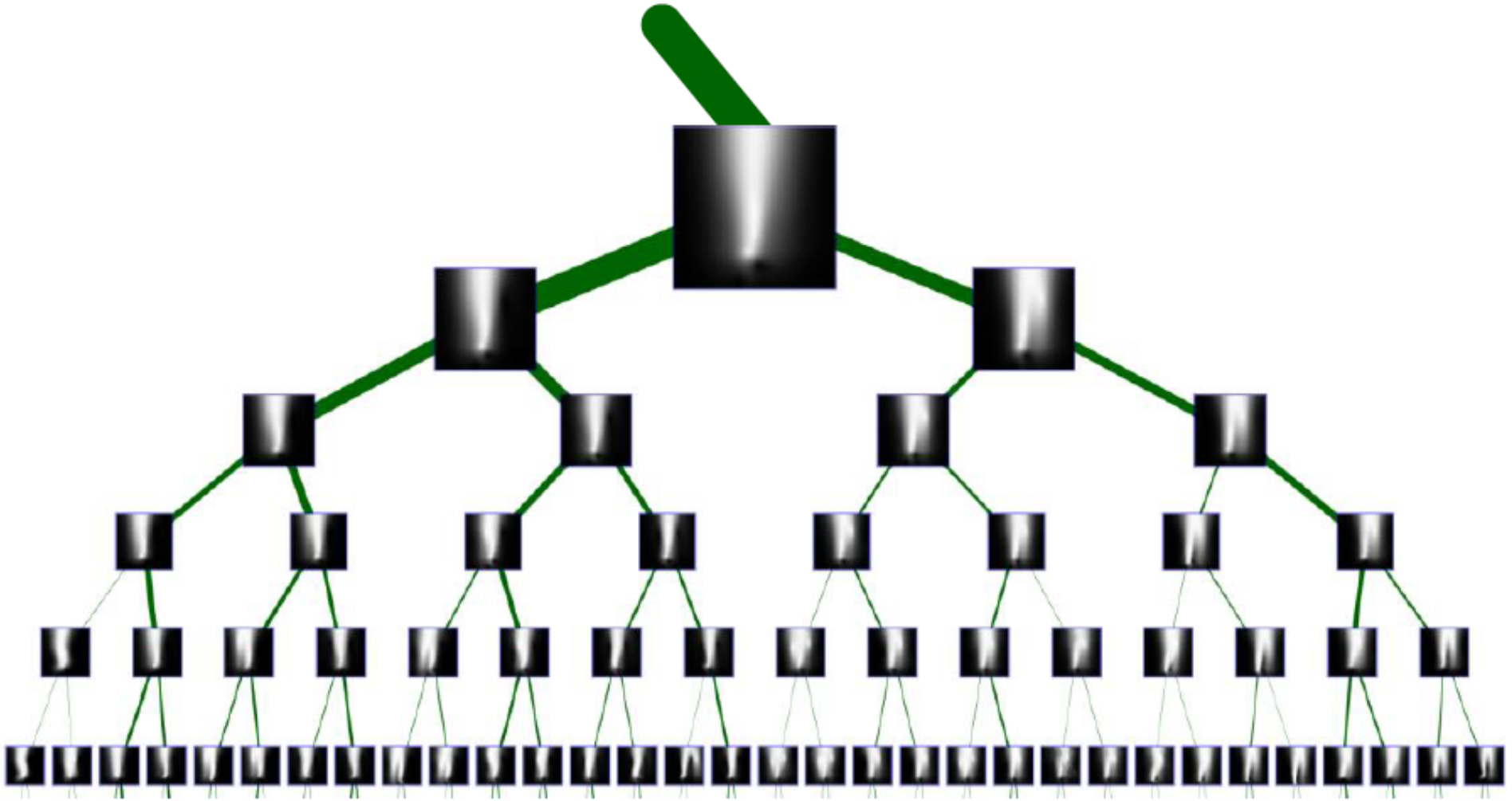
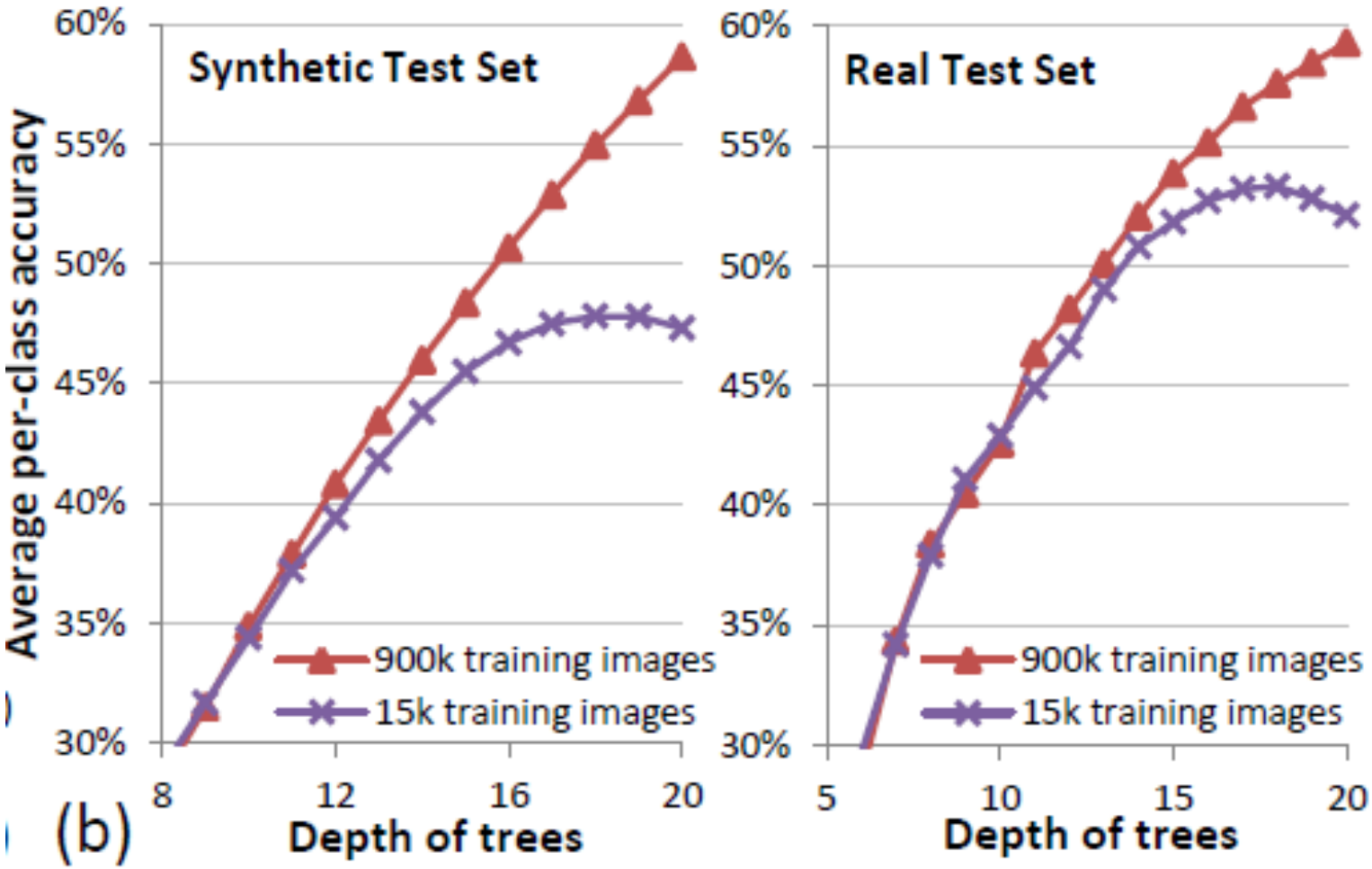


Figure 11. **Visualization of a trained decision tree.** Two separate subtrees are shown. A depth image patch centered on each pixel is taken, depth normalized, and binarized to a foreground/background silhouette. The patches are averaged across all pixels that reached any given tree node. The thickness of the edges joining the tree nodes is proportional to the number of pixels, and here shows fairly balanced trees. All pixels from 15k images are used to build the visualization shown.


Synthetic Test Set
Real Test Set
Average per-class accuracy
60%
55%
50%
45%
40%
35%
30%
900k training images
15k training images
8
12
16
20
5
10
15
20
Depth of trees
(b)

Why the class is not over yet