Multi-Class and Structured Classification

Simon Lacoste-Julien

Machine Learning Workshop

Friday 8/24/07

[built from slides from Guillaume Obozinksi]

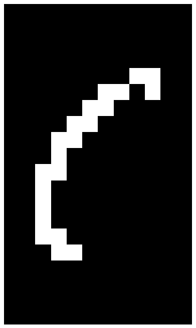
Basic Classification in ML

	Input	Output
	$\mathbf{x} \in \mathcal{X}$	$y \in \mathcal{Y}$

Spam filtering

Binary

Character recognition



Multi-Class

C

[thanks to Ben Taskar for slide!]

Structured Classification

<u>Input</u>

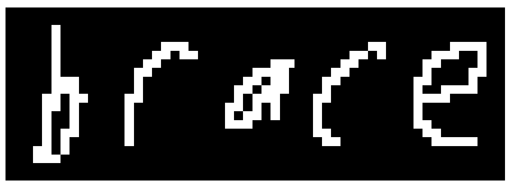
$\mathbf{x} \in \mathcal{X}$

<u>Output</u>

$\mathbf{y} \in \mathcal{Y}$

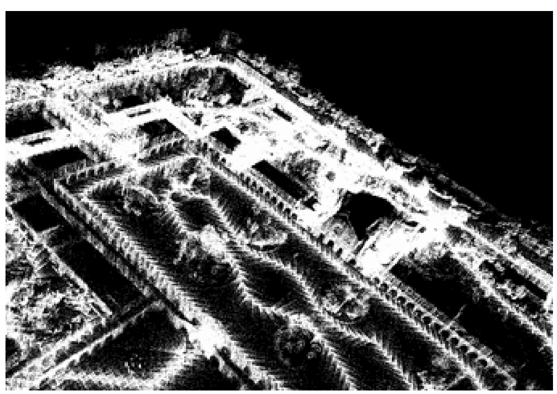
Handwriting recognition

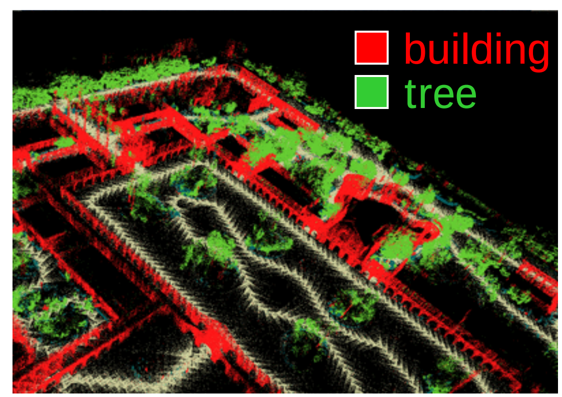
Structured output



brace

3D object recognition





[thanks to Ben Taskar for slide!]

Multi-Class Classification

- Multi-class classification : direct approaches
  - Nearest Neighbor
  - Generative approach & Naïve Bayes
  - Linear classification:
    - geometry
    - Perceptron
    - K-class (polychotomous) logistic regression
    - K-class SVM
- Multi-class classification through binary classification
  - One-vs-All
  - All-vs-all
  - Others
  - Calibration

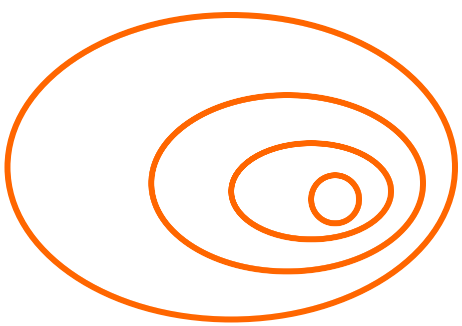
Multi-label classification

- Is it eatable?
- Is it sweet?
- Is it a fruit?
- Is it a banana?

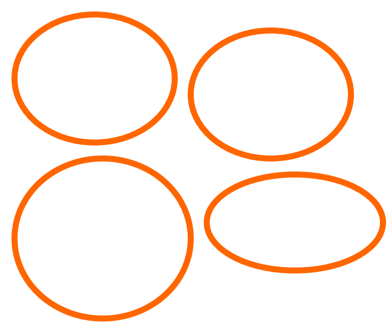
Is it a banana?
Is it an apple?
Is it an orange?
Is it a pineapple?

Is it a banana?
Is it yellow?
Is it sweet?
Is it round?

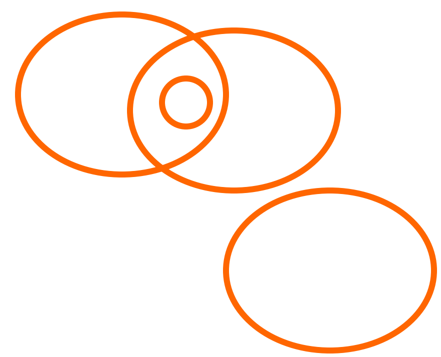
Different structures



Nested/ Hierarchical



Exclusive/ Multi-class



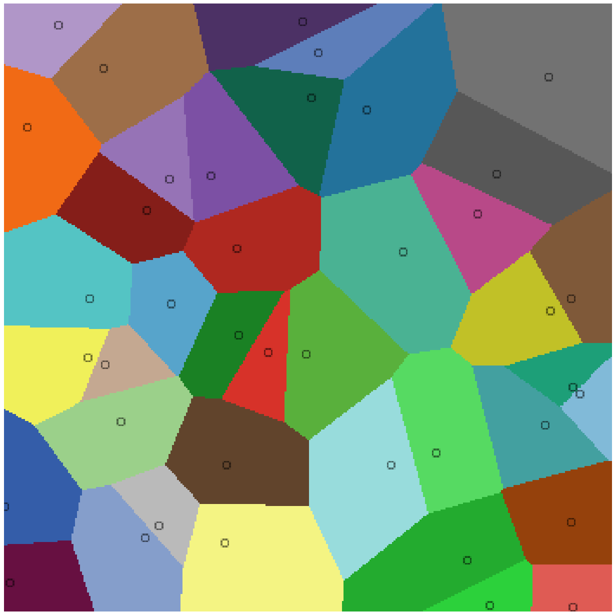
General/Structured

Nearest Neighbor, Decision Trees

- From the classification lecture:

• NN and k-NN were already phrased in a multi-class framework

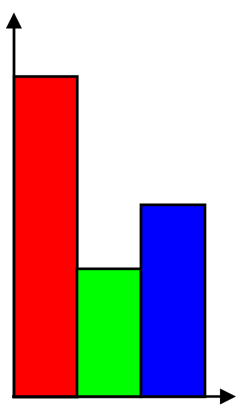
• For decision tree, want purity of leaves depending on the proportion of each class (want one class to be clearly dominant)



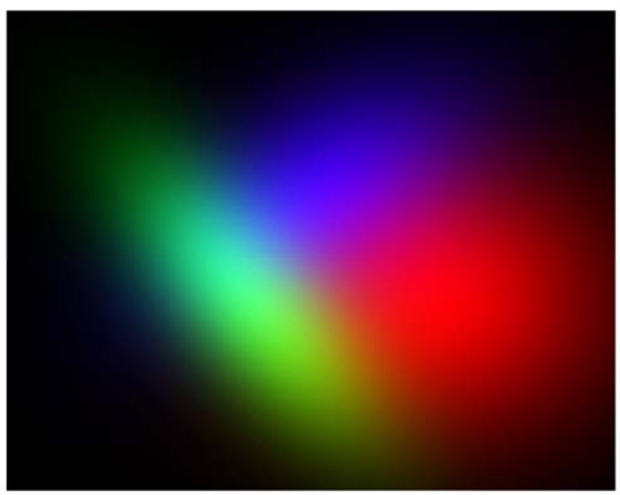
Generative models

As in the binary case:

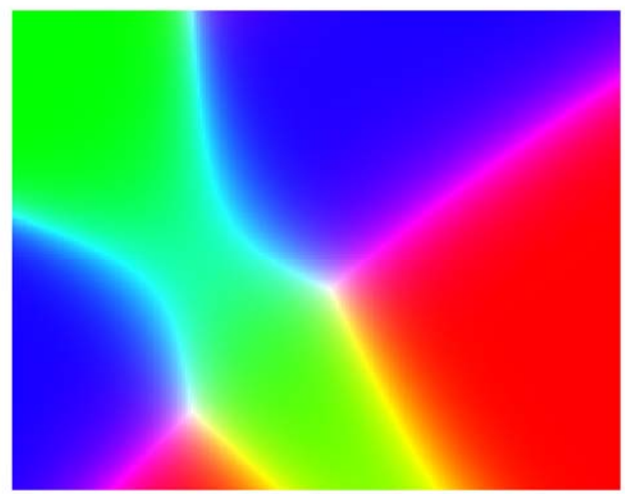
1. Learn p(y) and p(y|x)
2. Use Bayes rule: $p(y=k|x) = \frac{p(x|y=k)p(y=k)}{p(x)}$
3. Classify as $\hat{y}(x) = \mathsf{argmax}_y\, p(y|x)$



p(y)



p(x|y)

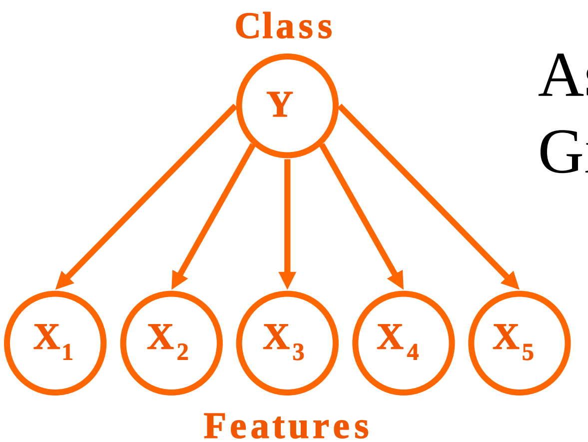


p(y|x)

Generative models

- Advantages:
  - Fast to train: only the data from class k is needed to learn the k^{th} model (reduction by a factor k compared with other method)
  - Works well with little data provided the model is reasonable
- Drawbacks:
  - Depends on the quality of the model
  - Doesn't model $p(y|x)$ directly
  - With a lot of datapoints doesn't perform as well as discriminative methods

Naïve Bayes



Assumption:
Given the class the features are independent

$$p(x|y=k) = \prod_i p(x_i|y=k)$$

$\Rightarrow$ **Bag-of-words models**

$$\log p(y=k|x) = \sum_i \log p(x_i|y=k) + \log p(y=k) - \log p(x)$$

If the features are discrete:

$$\log p(y=k|x) = \sum_i \sum_{u_i} \log p(u_i|y=k) \mathbf{1}\{x_i = u_i\} + \log p(y=k) - \log p(x)$$

$$\log p(y=k|x) = w_k^\top \Phi(x) + \log p(y=k) - \log p(x)$$

$$\log \frac{p(y=k|x)}{p(y=j|x)} = (w_k - w_j)^\top \Phi(x) + \log \frac{p(y=k)}{p(y=j)}$$

Linear classification

- Each class has a parameter vector (w_k, b_k)
- x is assigned to class k iff $w_k^\top x + b_k \geq \max_j w_j^\top x + b_j$

- Note that we can break the symmetry and choose $(w_1, b_1) = 0$
- For simplicity set $b_k = 0$
(add a dimension and include it in w_k)
- So learning goal given separable data: choose w_k s.t.

$$\forall (x^i, y^i), \quad \underbrace{{w_{y^i}}^\top x^i}_{\text{score of truth}} \geq \max_j \underbrace{w_j^\top x^i}_{\text{score of competitor.}}$$

Three discriminative algorithms

Perceptron: $$\max_W \sum_i \left[{w_{y^i}}^\top x^i - \max_k w_k^\top x^i \right]$$

[mistake driven]

K-class logistic regression: $$\max_W \sum_i \left[{w_{y^i}}^\top x^i - \operatorname{softmax}_k w_k^\top x^i \right]$$

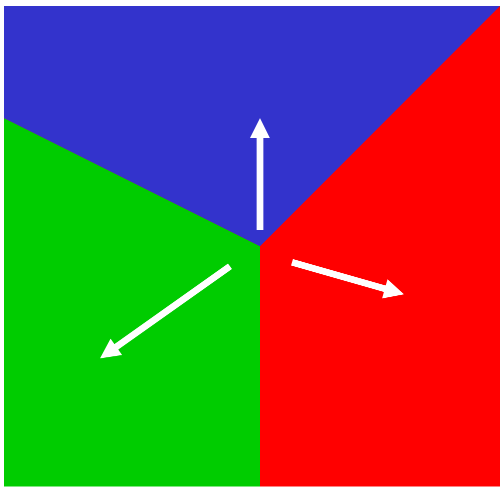
[max conditional likelihood]

K-class SVM: $$\max_W \sum_i \left[{w_{y^i}}^\top x^i - \max_k (w_k^\top x^i + \mathbf{1}\{k \neq y^i\}) \right]$$

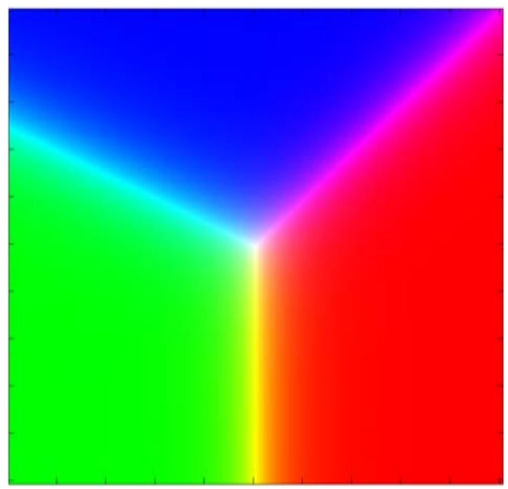
[large margin method]

Geometry of Linear classification

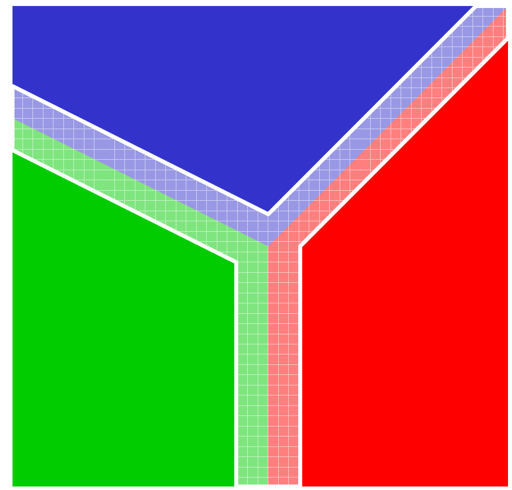
Perceptron



K-class logistic regression



K-class SVM



Multiclass Perceptron

Online: for each datapoint

Predict: $\hat{y}_i = \arg\max_y \; w_y^\top x^i$

Update: if $\hat{y}_i \neq y^i$ then

$$\begin{cases} w_{y^i,t+1} = w_{y^i,t} + \alpha x^i \\ w_{\hat{y}_i,t+1} = w_{\hat{y}_i,t} - \alpha x^i \end{cases}$$

$$\bar{w} = \frac{1}{T}\sum_{t=1}^{T} w_t$$

- Advantages :
  - Extremely simple updates (no gradient to calculate)
  - No need to have all the data in memory (some point stay classified correctly after a while)
- Drawbacks
  - If the data is not separable decrease α slowly…

Polychotomous logistic regression

$$p(y = k|x) = \frac{\exp w_k^\top x}{\sum_j \exp w_j^\top x}$$

distribution in exponential form

$$\log p(y = k|x) = w_k^\top x - \log \sum_j \exp w_j^\top x$$

Online: for each datapoint

"soft mistake update"

$$w_j \leftarrow w_j + \alpha x^i (\, \mathbf{1}\{j = y^i\} - p(y = j | x = x^i)\,)$$

Batch: all descent methods

Especially in large dimension, use regularization

$\|w\|_2^2, \ \|w\|_1$

small flip label probability

(0,0,1) $\longrightarrow$ (.1,.1,.8)

Advantages:

- Smooth function
- Get probability estimates

Drawbacks:

- Non sparse

Multi-class SVM

Intuitive formulation: without regularization / for the separable case $\max_W \left[\sum_i {w_{y^i}}^\top x^i - \max_j (\mathbf{1}\{j \neq y^i\} + w_j^\top x^i) \right]$

Primal problem: QP

$$\min_{w_1,\ldots,w_K} \quad \frac{1}{2}\|(w_1, \ldots, w_K)\|^2 + C \sum_{ik} \xi_{ik}$$

$$\text{s.t.} \quad \forall (i,k), \quad {w_{y^i}}^\top x^i - w_k^\top x^i \geq \mathbf{1}\{k \neq y^i\} - \xi_{ik}$$

Solved in the dual formulation, also Quadratic Program

Main advantage: Sparsity (but not systematic)

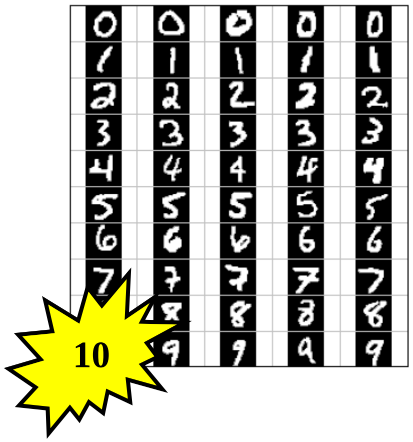
- Speed with SMO (heuristic use of sparsity)
- Sparse solutions

Drawbacks:

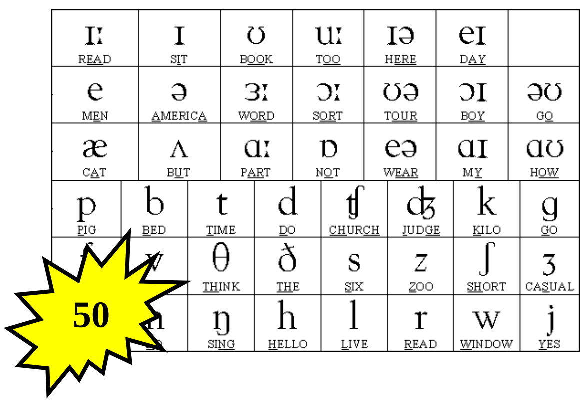
- Need to recalculate or store $x_i^T x_j$
- Outputs not probabilities

Real world classification problems

Digit recognition

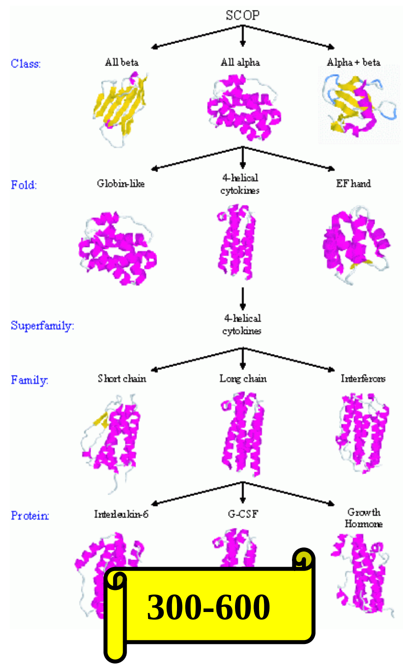


Phoneme recognition

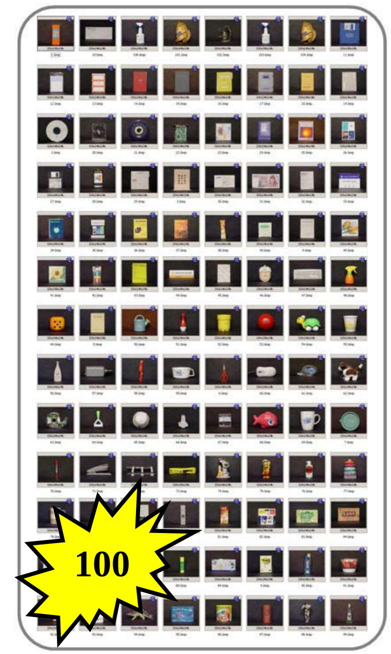


[Waibel, Hanzawa, Hinton,Shikano, Lang 1989]

Automated protein classification



Object recognition



http://www.glue.umd.edu/~zhelin/recog.html

- The number of classes is sometimes big
- The multi-class algorithm can be heavy

Combining binary classifiers

One-vs-all For each class build a classifier for that class vs the rest

- Often very imbalanced classifiers (use asymmetric regularization)

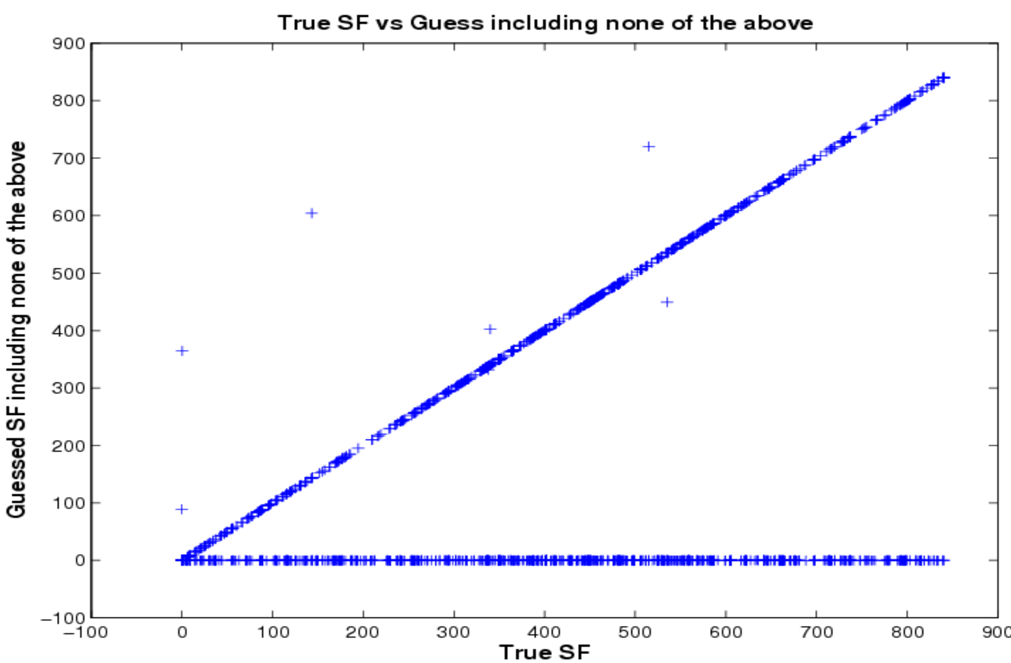
All-vs-all For each class build a classifier for that class vs the rest

- A priori a large number of classifiers $\binom{n}{2}$ to build ***but…***
  - The pairwise classification are way much faster
  - The classifications are balanced (easier to find the best regularization)

… so that in many cases it is clearly faster than one-vs-all

Confusion Matrix

- Visualize which classes are more difficult to learn
- Can also be used to compare two different classifiers
- Cluster classes and go hierachical [Godbole, ‘02]



Classification of 20 news groups

Predicted classes

Actual classes

Classname		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
alt.atheism	1	251	6	1	3	32	1	1	2	1	2	0	0	0	0	0	0	0	0	0	0
soc.religion.christian	2	9	277	0	1	6	0	0	1	0	0	0	0	0	1	2	2	0	0	0	1
sci.space	3	3	1	273	1	0	1	2	0	1	1	9	0	0	1	2	3	0	0	1	1
talk.politics.misc	4	2	0	3	213	24	3	0	17	3	0	0	0	0	0	0	1	0	1	33	0
talk.religion.misc	5	88	36	2	23	132	0	1	0	0	0	0	0	0	0	0	2	0	1	15	0
rec.autos	6	0	0	0	3	1	272	0	0	0	7	1	2	1	6	4	1	0	0	2	0
comp.windows.x	7	1	1	2	1	0	1	246	0	2	2	30	5	3	1	1	2	1	1	0	0
talk.politics.mideast	8	0	3	1	18	0	0	0	275	0	1	0	0	0	0	0	0	0	1	1	0
sci.crypt	9	1	0	1	2	1	0	3	0	284	0	3	0	1	0	0	1	0	0	3	0
rec.motorcycles	10	0	0	0	1	0	4	1	0	0	286	1	2	0	1	2	1	0	0	1	0
comp.graphics	11	0	1	2	1	1	0	10	1	2	0	243	23	7	3	3	3	0	0	0	0
comp.sys.ibm.pc.hardware	12	0	0	0	0	0	2	7	0	1	0	5	243	23	12	3	1	3	0	0	0
comp.sys.mac.hardware	13	0	0	1	1	0	2	1	0	0	0	7	10	260	8	9	1	0	0	0	0
sci.electronics	14	1	0	1	0	1	5	2	0	2	0	7	13	13	245	6	3	0	1	0	0
misc.forsale	15	0	1	4	2	0	12	1	0	0	4	1	19	10	8	233	1	0	1	1	2
sci.med	16	0	1	5	0	1	1	0	0	0	1	2	0	2	7	2	275	0	1	1	1
comp.os.mswindows.misc	17	1	0	2	0	1	1	58	1	3	0	38	71	17	3	6	0	97	1	0	0
rec.sport.baseball	18	2	1	1	0	0	0	0	0	0	0	4	0	0	0	1	1	0	282	1	7
talk.politics.guns	19	0	0	0	9	5	1	0	0	1	0	0	0	0	1	0	0	1	1	281	0
rec.sport.hockey	20	0	1	0	0	0	1	0	0	0	2	0	0	1	1	0	0	0	3	0	291

[Godbole, ‘02]

BLAST classification of proteins in 850 superfamilies

Calibration

How to measure the confidence in a class prediction?

Crucial for:

1. Comparison between different classifiers
2. Ranking the prediction for ROC/Precision-Recall curve
3. In several application domains having a measure of confidence for each individual answer is very important (e.g. tumor detection)

Some methods have an implicit notion of confidence e.g. for SVM the distance to the class boundary relative to the size of the margin other like logistic regression have an explicit one.

Calibration

Definition: the decision function f of a classifier is said to be *calibrated* or *well-calibrated* if

$$\mathbf{P}(x \text{ is correctly classified} \,|\, f(x) = s) \simeq s$$

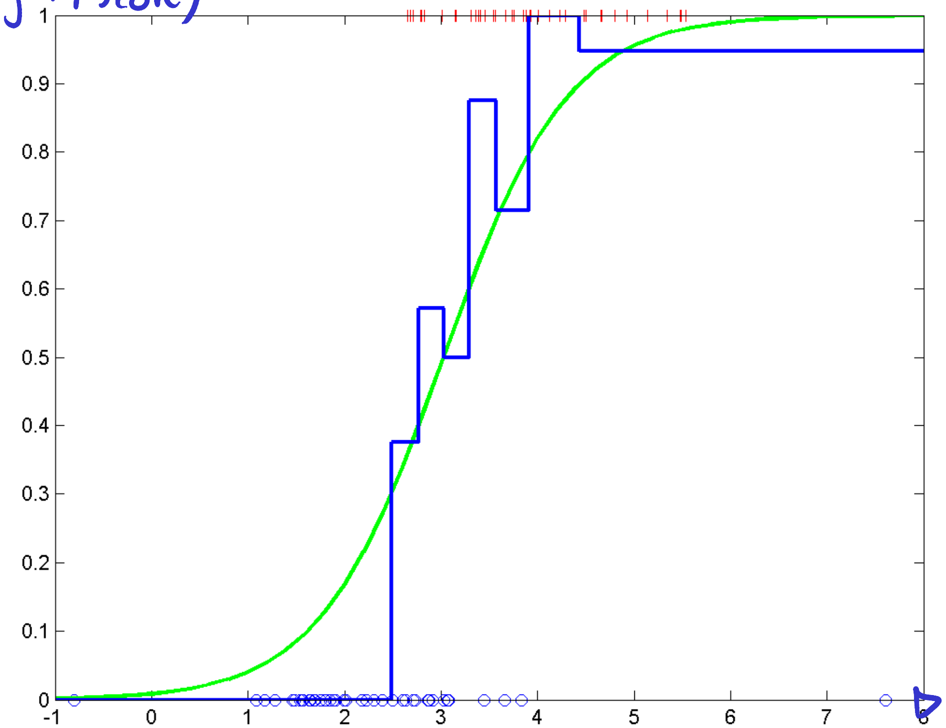
Informally f is a good estimate of the probability of classifying correctly a new datapoint x which would have output value x.

Intuitively if the "raw" output of a classifier is g you can calibrate it by estimating the probability of x being well classified given that g(x)=y for all y values possible.

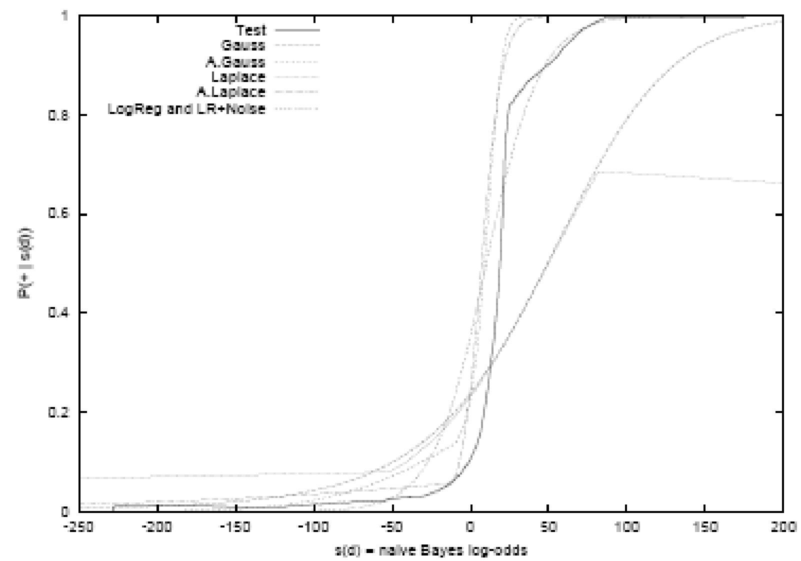
Calibration

Example: a logistic regression, or more generally calculating a Bayes posterior should yield a reasonably *well-calibrated* decision function.

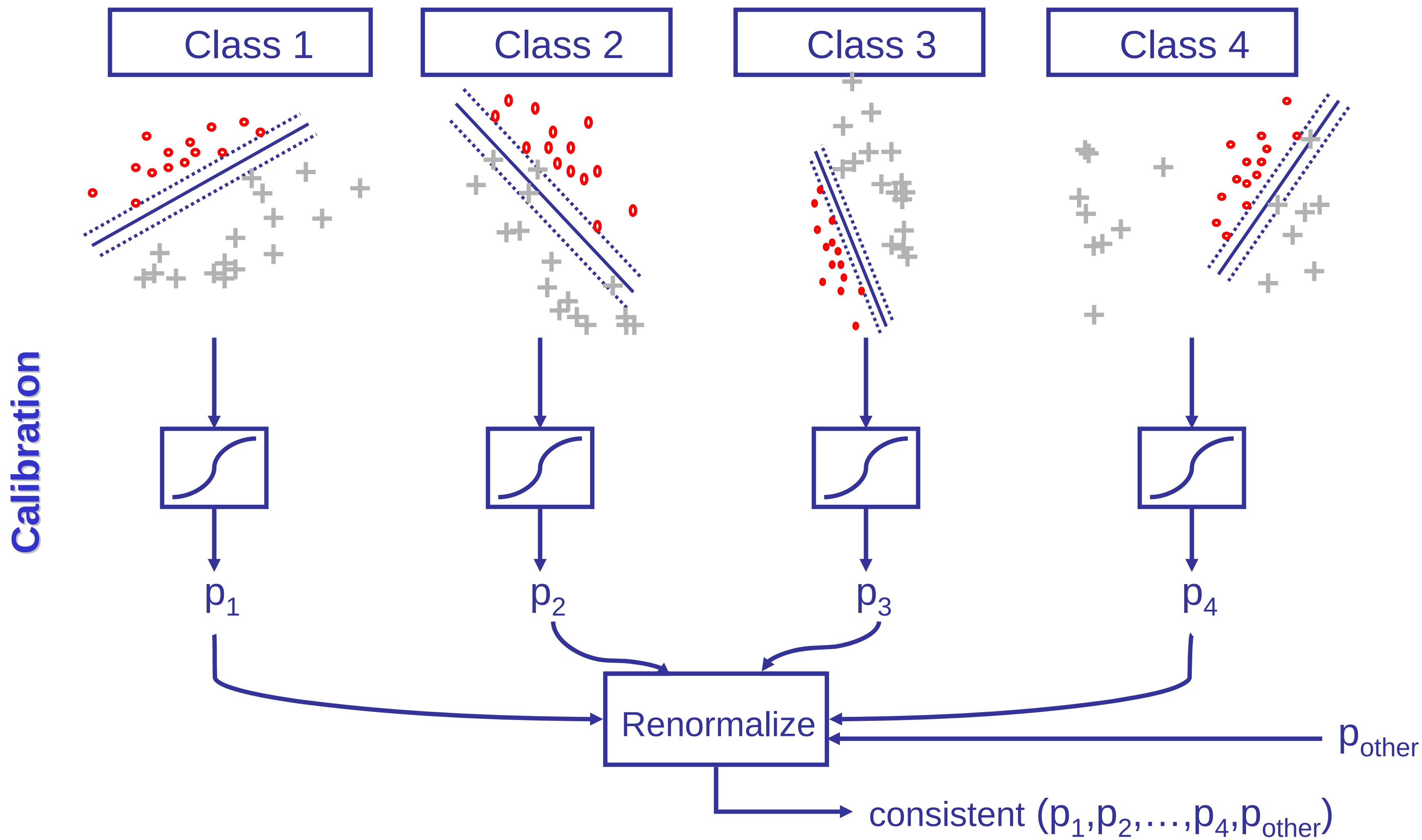
prob(y=1 | score)



score e.g. $w \cdot x_i + b$



Combining OVA calibrated classifiers

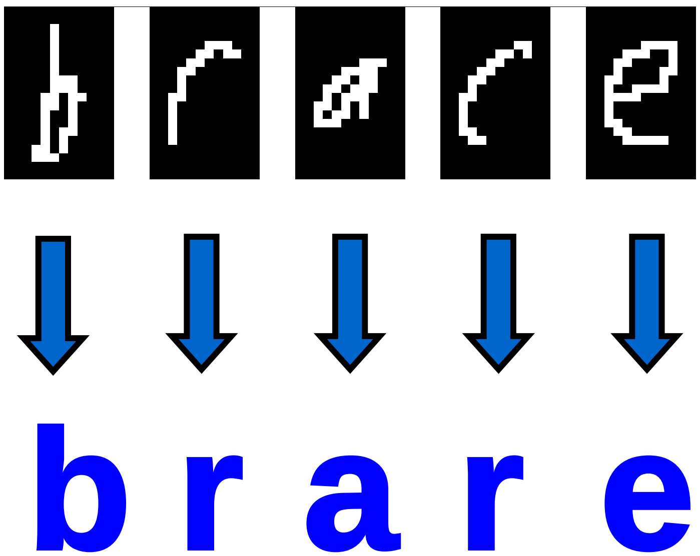


Other methods for calibration

- Simple calibration
    - Logistic regression
    - Intraclass density estimation + Naïve Bayes
    - Isotonic regression
- More sophisticated calibrations
    - Calibration for A-vs-A by Hastie and Tibshirani

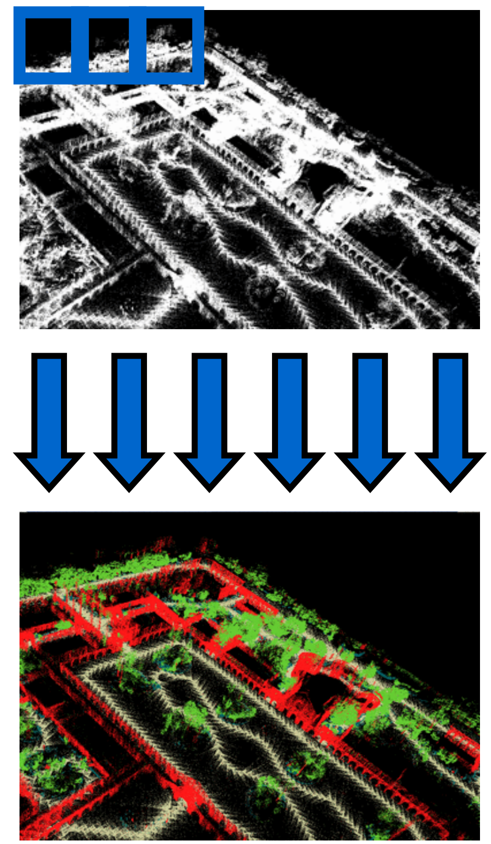
Structured classification

Local Classification



x

y

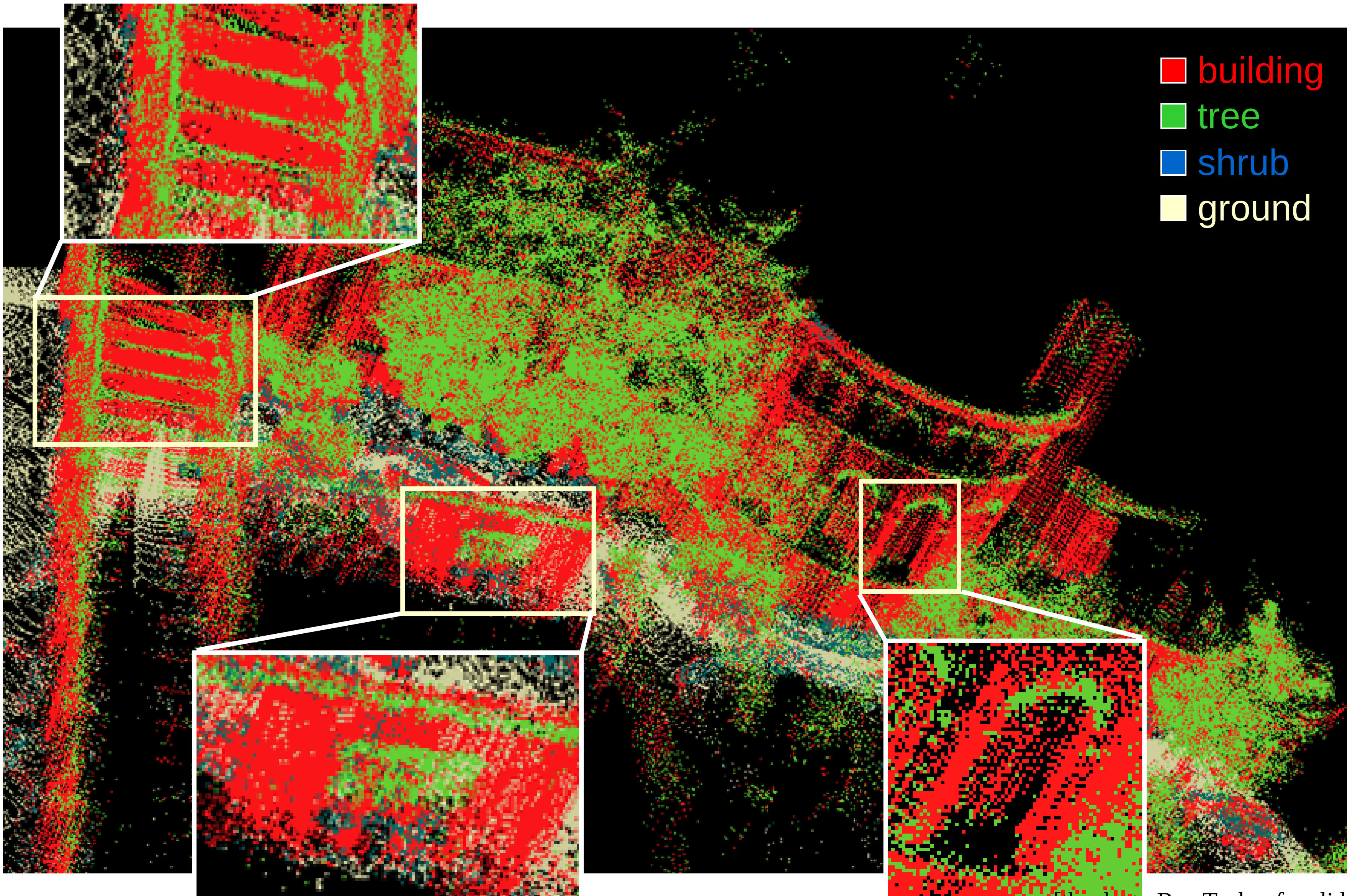


Classify using local information

$\Rightarrow$ Ignores correlations!

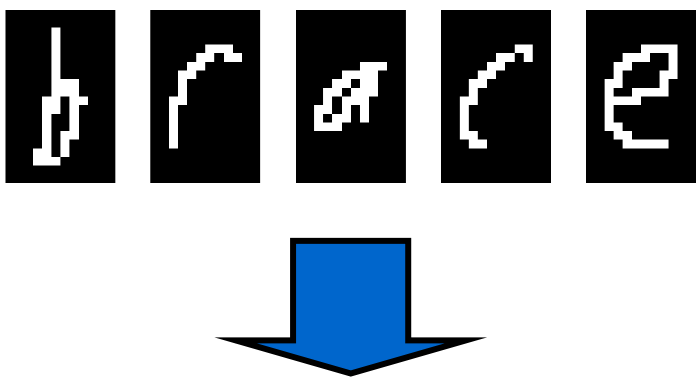
[thanks to Ben Taskar for slide!]

Local Classification



[thanks to Ben Taskar for slide!]

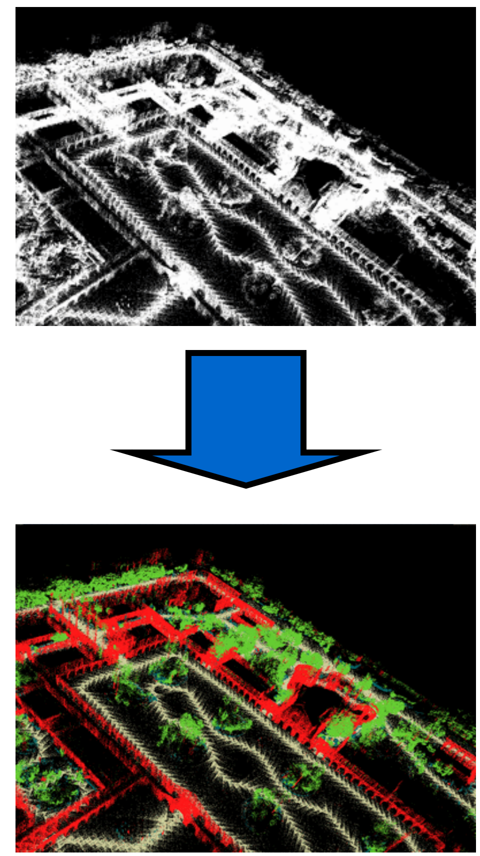
Structured Classification



b r a c e

x

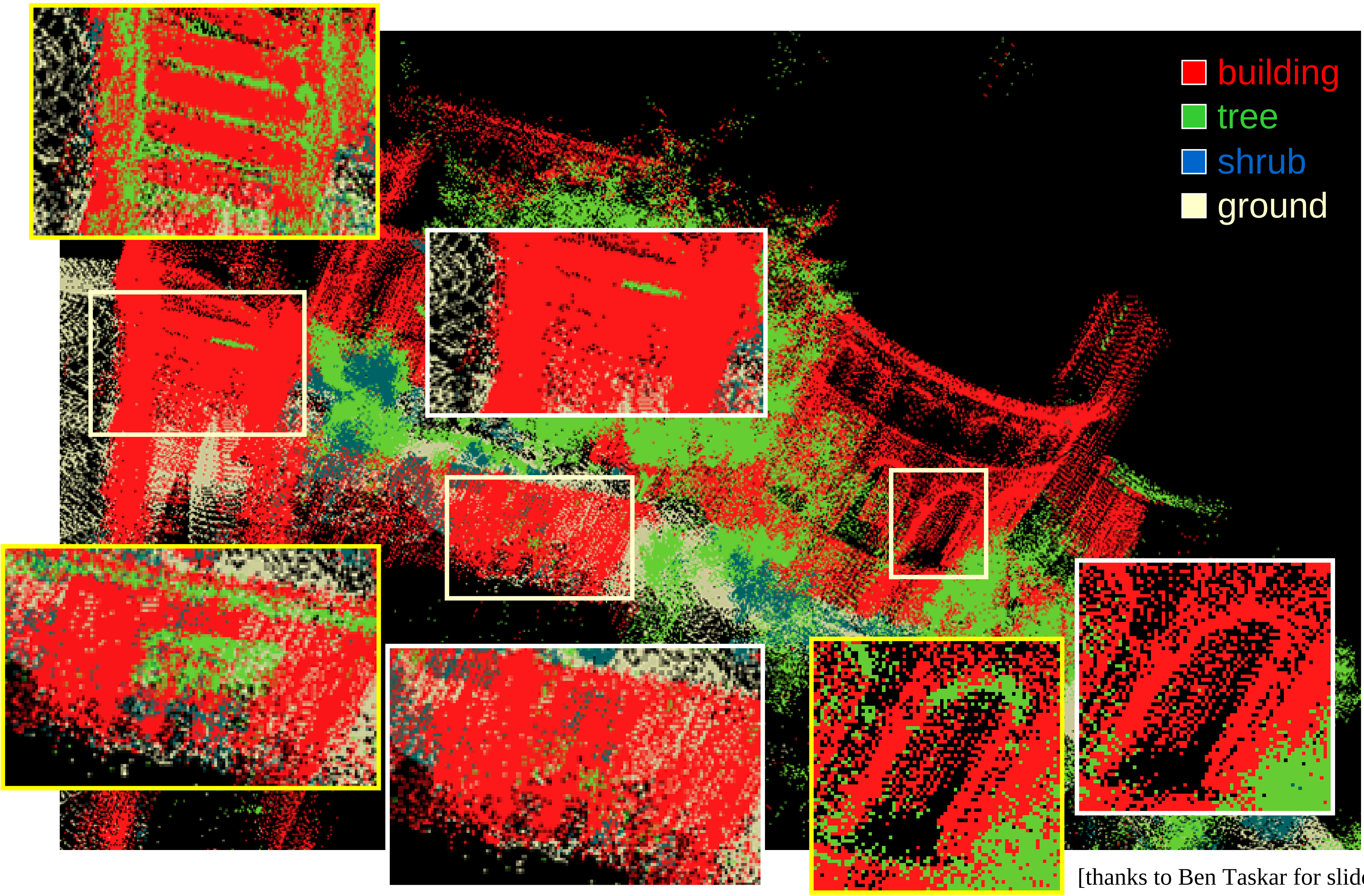
y



- Use local information
- Exploit correlations

[thanks to Ben Taskar for slide!]

Structured Classification



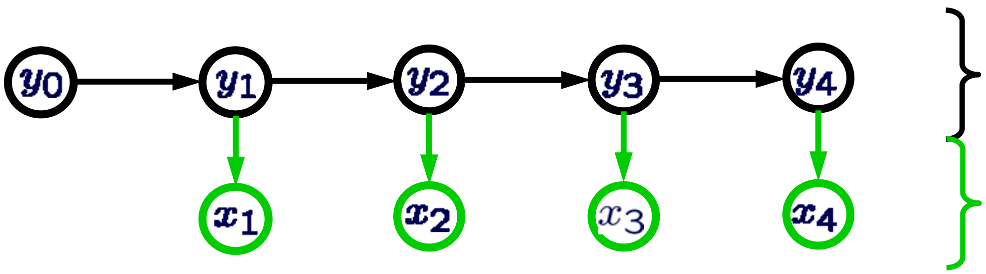
[thanks to Ben Taskar for slide!]

Structured Classification

- Structured classification : direct approaches
  - Generative approach: Markov Random Fields (Bayesian modeling with graphical models)
  - Linear classification:
    - Structured Perceptron
    - Conditional Random Fields (counterpart of logistic regression)
    - Large-margin structured classification

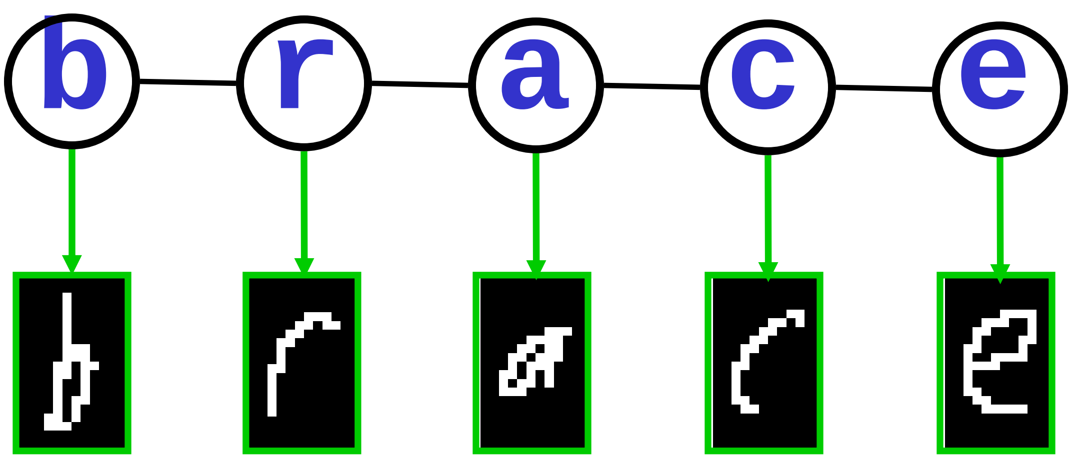
Structured classification

Simple example HMM:



"Label sequence"

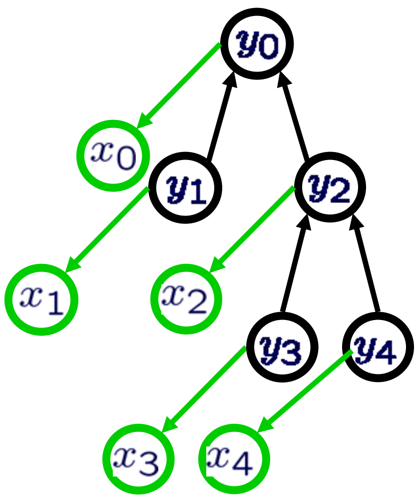
"Observation sequence"



Optical Character Recognition

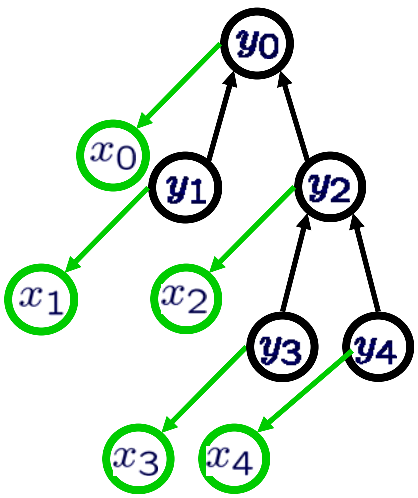
Tree model 1

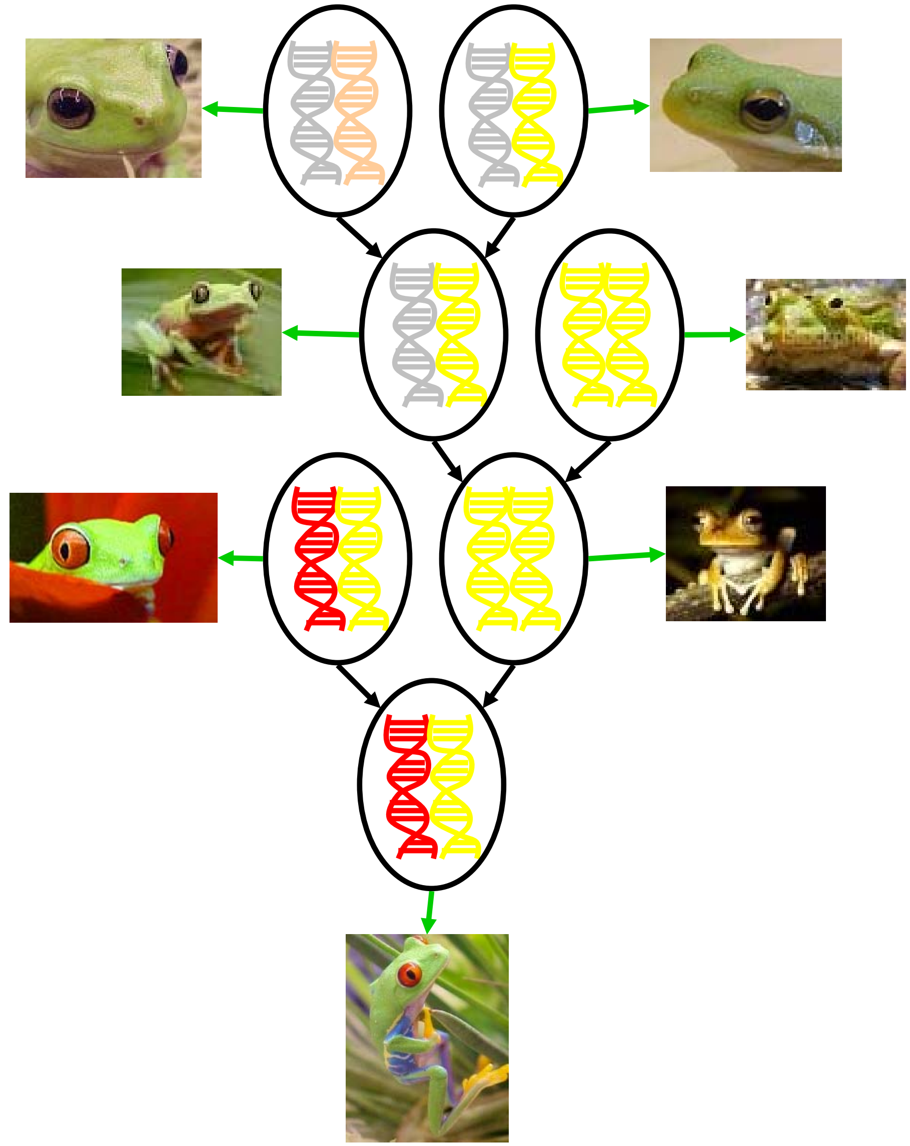
“Label structure”



“Observations”

Tree model 1

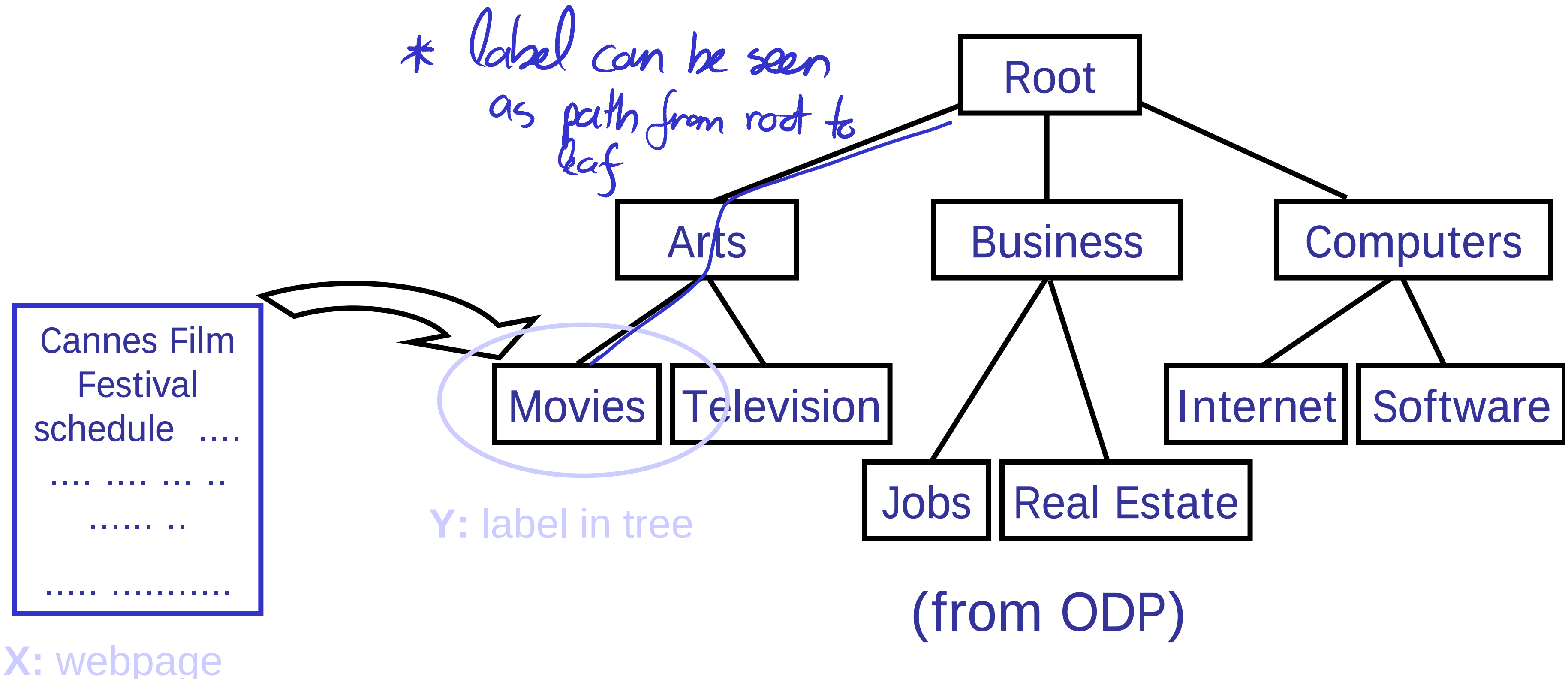




Eye color inheritance:

haplotype inference

Tree Model 2:
Hierarchical Text Classification



Grid model

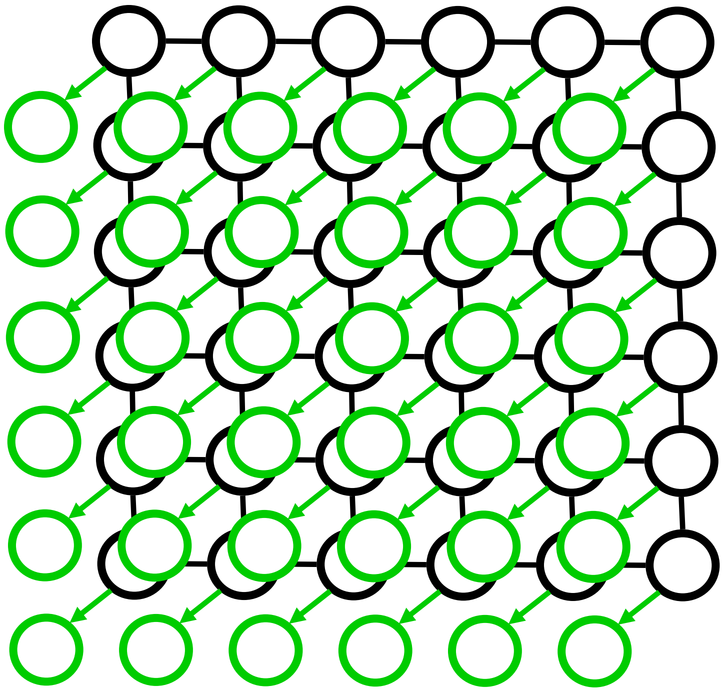


Image segmentation

Original image

Segmented = “Labeled” image

Structured Model

- Main idea: define scoring function which **decomposes** as sum of features scores k on "**parts**" p:

$$score(\mathbf{x}, \mathbf{y}, \mathbf{w}) = \mathbf{w}^\top \Phi(\mathbf{x}, \mathbf{y}) = \sum_{k,p} w_k^\top \phi_k(\mathbf{x}_p, \mathbf{y}_p)$$

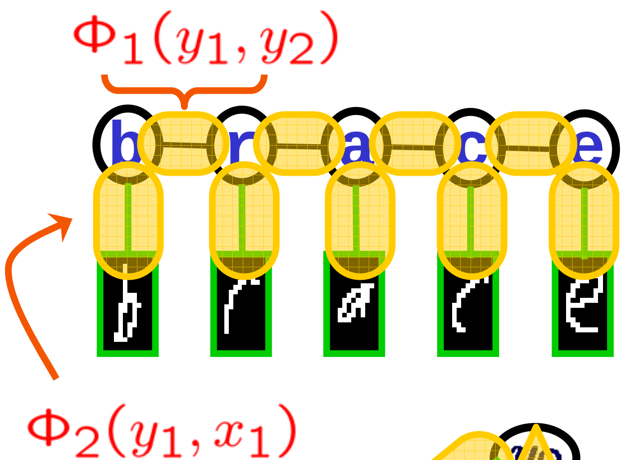
- Label examples by looking for max score:

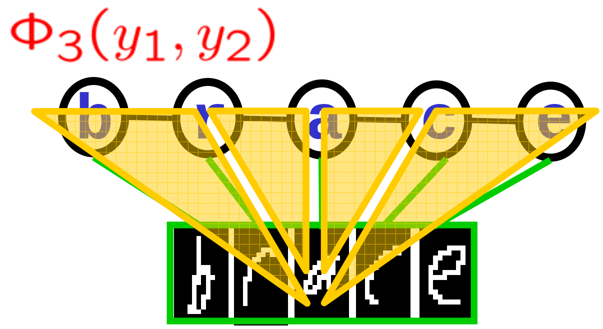
$$prediction(\mathbf{x}, \mathbf{w}) = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} score(\mathbf{x}, \mathbf{y}, \mathbf{w})$$

space of feasible outputs

- Parts = nodes, edges, etc.

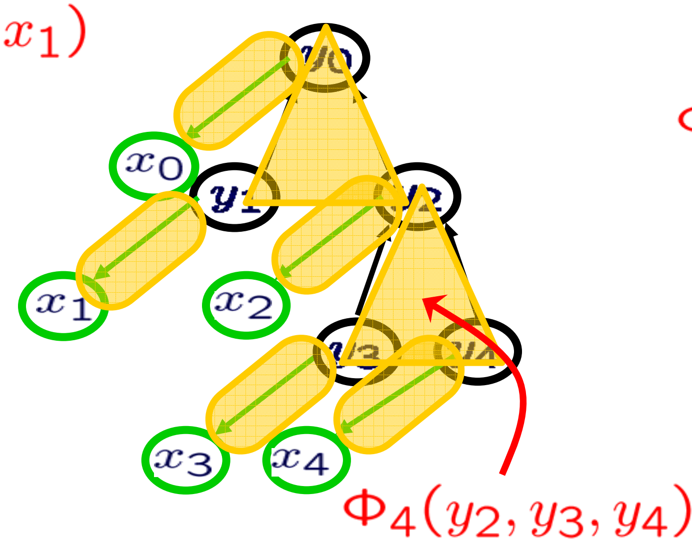
Cliques and Features





In undirected graphs:

cliques = groups of completely interconnected variables



$\Phi_1(y_1, y_2) = 1$ iff $y_1 =$ "b" & $y_2 =$ "r"

$\Phi_2(y_1, x_1) = 1$ iff $y_1 =$ "b" & some "b" pixels are "on" in x_1

$\Phi_4(y_2, y_3, y_4) = \bar{y}_2 \vee y_3 \vee y_4$

In directed graphs:

cliques = variable+its parents

Exponential form

Once the graph is defined the model can be written in exponential form

$$p(x, y) = \frac{1}{Z} \exp \sum_{k,C} w_k^\top \phi_k(y_C, x_C)$$

parameter vector

$$p(x, y) = \frac{1}{Z} \exp \mathbf{w}^\top \Phi(y, x)$$

feature vector

Comparing two labellings with the likelihood ratio

$$\frac{p(x, \tilde{y})}{p(x, y)} = \frac{\exp \mathbf{w}^\top \Phi(\tilde{y}, x)}{\exp \mathbf{w}^\top \Phi(y, x)}$$

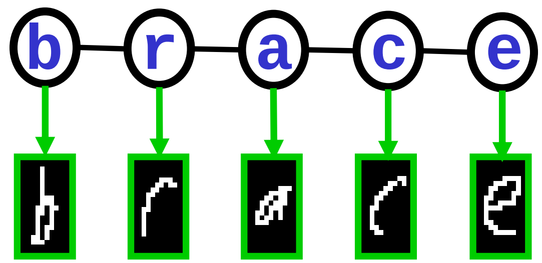
$\tilde{y}$ wins over y when $\mathbf{w}^\top \Phi(\tilde{y}, x) > \mathbf{w}^\top \Phi(y, x)$

Decoding and Learning

Three important operations on a general structured (e.g. graphical) model:

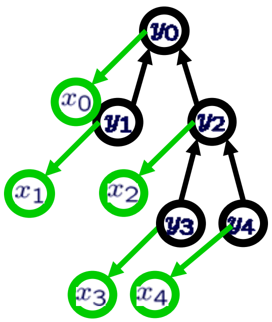
- **Decoding:** find the right label sequence $\underset{y_1,\ldots,y_n}{\operatorname{argmax}}\, p(y_1, \ldots, y_n | x)$
- **Inference:** compute probabilities of labels $\forall j, \quad p(y_j | x)$
- **Learning:** find model + parameters *w* so that decoding works

HMM example:



- **Decoding:** Viterbi algorithm
- **Inference:** forward-backward algorithm
- **Learning:** e.g. transition and emission counts
(case of learning a generative model from fully labeled training data)

Decoding and Learning



- **Decoding:** algorithm on the graph (eg. max-product)
- **Inference:** algorithm on the graph
(eg. sum-product, belief propagation, junction tree, sampling)
- **Learning:** inference + optimization

Use **dynamic programming** to take advantage of the structure

1. Focus of graphical model class
2. Need 2 essential concepts:
   1. cliques: variables that directly depend on one another
   2. features (of the cliques): some functions of the cliques

Our favorite (discriminative) algorithms

Perceptron: $$\max_{\mathbf{w}} \sum_i \left[\mathbf{w}^\top \Phi(x^i, y^i) - \max_y \mathbf{w}^\top \Phi(x^i, y) \right]$$

[mistake driven]

CRF: $$\max_{\mathbf{w}} \sum_i \left[\mathbf{w}^\top \Phi(x^i, y^i) - \operatorname{softmax}_y \mathbf{w}^\top \Phi(x^i, y) \right]$$

(Conditional Random Field)

[max conditional likelihood]

M^{3}net: $$\max_{\mathbf{w}} \sum_i \left[\mathbf{w}^\top \Phi(x^i, y^i) - \max_y (\ell(y, y^i) + \mathbf{w}^\top \Phi(x^i, y)) \right]$$

[max. margin]

The devil is the details...

(Averaged) Perceptron

For each datapoint $\mathbf{x}^i$

Predict: $$\hat{\mathbf{y}}_i = \arg\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}_t^\top \Phi(\mathbf{x}^i, \mathbf{y})$$

Update: $$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \underbrace{\Big(\Phi(\mathbf{x}, \mathbf{y}^i) - \Phi(\mathbf{x}^i, \hat{\mathbf{y}}_i)\Big)}_{\text{update if } \hat{\mathbf{y}}_i \neq \mathbf{y}^i}$$

Averaged perceptron: $$\bar{\mathbf{w}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_t$$

Example: multiclass setting

Predict: $\hat{y}_i = \arg\max_y \; w_y^\top x^i$

Update: if $\hat{y}_i \neq y^i$ then

$$w_{y^i,t+1} = w_{y^i,t} + \alpha x^i$$
$$w_{\hat{y}_i,t+1} = w_{\hat{y}_i,t} - \alpha x^i$$

Feature encoding:

$$\begin{aligned}
\Phi(\mathbf{x}^i, y=1)^\top &= [\mathbf{x}^{i\top}\ 0 \ldots 0] \\
\Phi(\mathbf{x}^i, y=2)^\top &= [0\ \mathbf{x}^{i\top} \ldots 0] \\
&\vdots \\
\Phi(\mathbf{x}^i, y=K)^\top &= [0\ 0 \ldots \mathbf{x}^{i\top}] \\
\mathbf{w}^\top &= [w_1^\top\ w_2^\top \ldots w_K^\top]
\end{aligned}$$

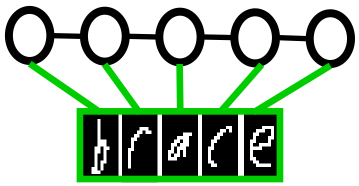
Predict: $\hat{\mathbf{y}}_i = \arg\max_{\mathbf{y}\in\mathcal{Y}} \; \mathbf{w}_t^\top \Phi(\mathbf{x}^i, \mathbf{y})$

Update: $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha \underbrace{\left(\Phi(\mathbf{x}, \mathbf{y}^i) - \Phi(\mathbf{x}^i, \hat{\mathbf{y}}_i)\right)}_{\text{update if } \hat{\mathbf{y}}_i \neq \mathbf{y}^i}$

CRF

Z difficult to compute with complicated graphs

$$\frac{\exp \mathbf{w}^\top \Phi(y^i|x^i)}{\sum_y \exp \mathbf{w}^\top \Phi(y|x^i)}$$



Conditioned on all the observations

Introduction by Hannah M.Wallach

http://www.inference.phy.cam.ac.uk/hmw26/crf/

MEMM & CRF, Mayssam Sayyadian, Rob McCann

anhai.cs.uiuc.edu/courses/498ad-fall04/local/my-slides/crf-students.pdf

M³net

No Z …

The margin penalty $\ell(y, y^i)$ can "factorize" according to the problem structure

Introduction by Simon Lacoste-Julien

http://www.cs.berkeley.edu/~slacoste/school/cs281a/project_report.html

Practical Summary

- For multiclass, usually more efficient for now to use one-vs-all approach + logistic calibration

- For structured classification:
  - define a structured score which you can easily maximize (using dyn. prog., etc...)
  - for simple start, use avg. structured perceptron with randomized order + decreasing learning rate
  - for better performance, use CRF or a max-margin method (M^3-net, SVM struct, etc.)

    code online

[thanks to Ben Taskar for slide!]

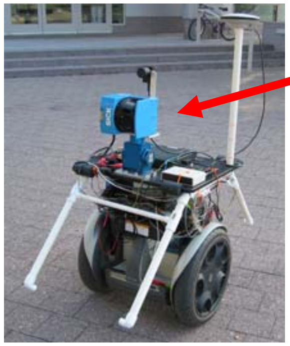
Object Segmentation Results

Data: [Stanford Quad by Segbot]

Trained on 30,000 point scene

Tested on 3,000,000 point scenes

Evaluated on 180,000 point scene



Segbot

M. Montemerlo

S. Thrun

Model	Error
Local learning Local prediction	32%
Local learning +smoothing	27%
Structured method	7%

[Taskar+al 04, Anguelov+Taskar+al 05]