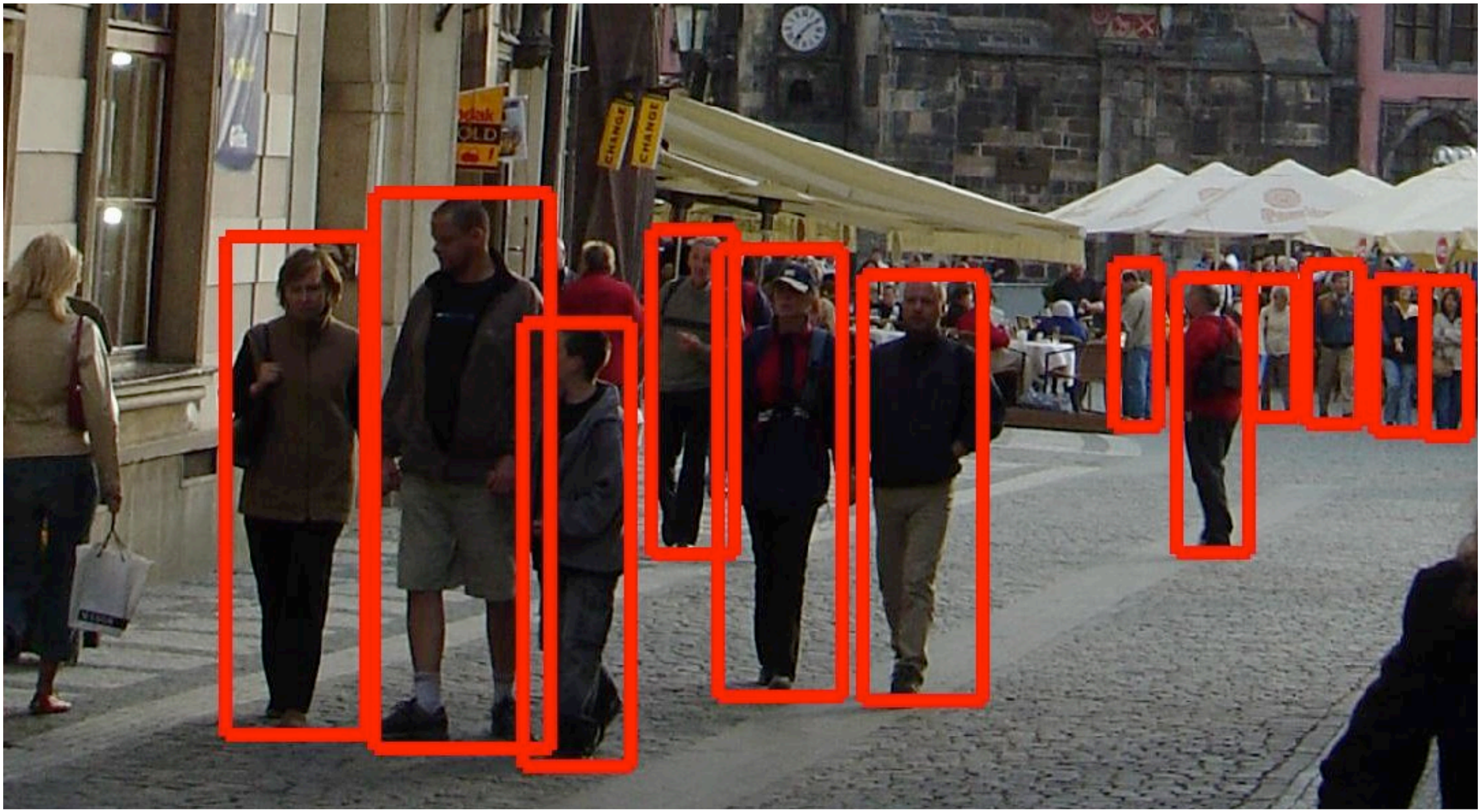
Classifiers

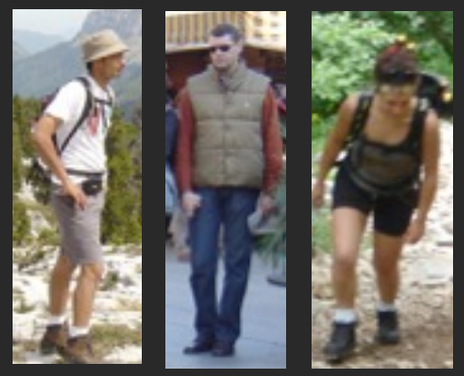
Outline

- Classifiers
- Apearance variation
- Parts

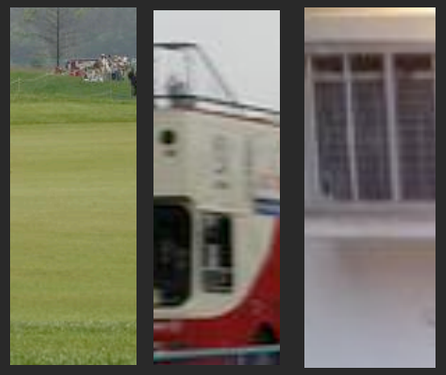

CHANGE
CHANGE

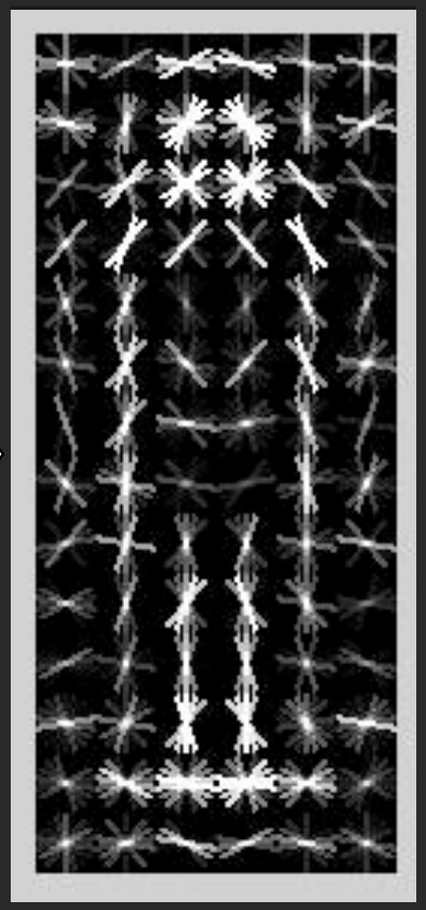
Template classifiers

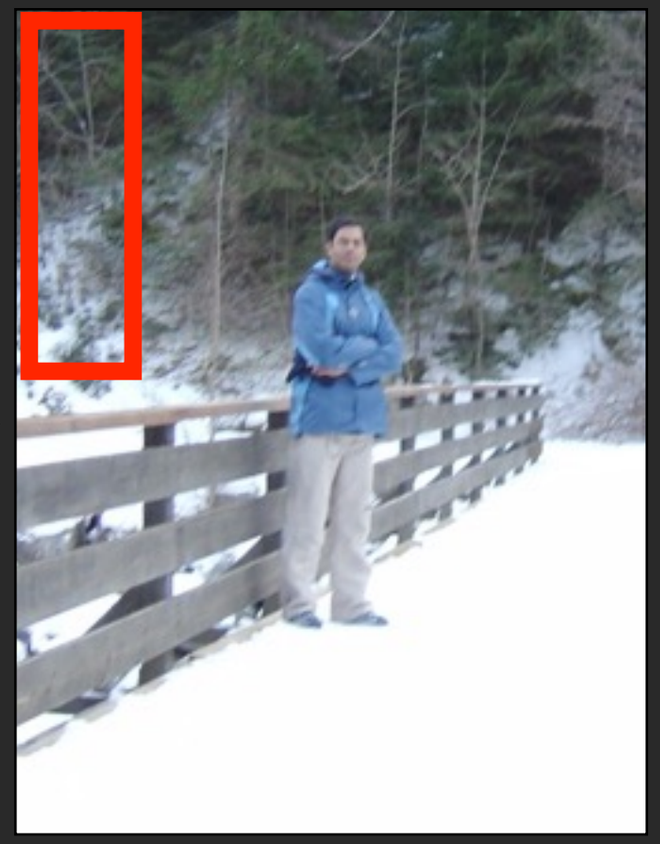
pos



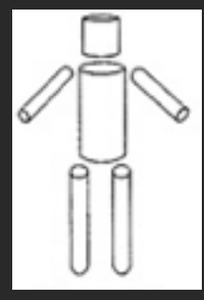
neg





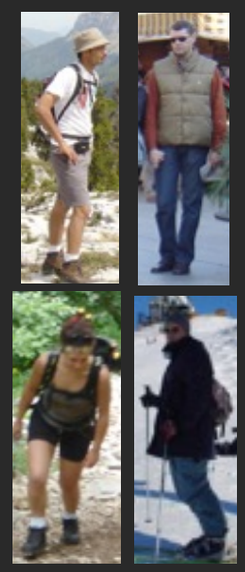


contrast with “old school”:



Large-scale learning

pos



neg

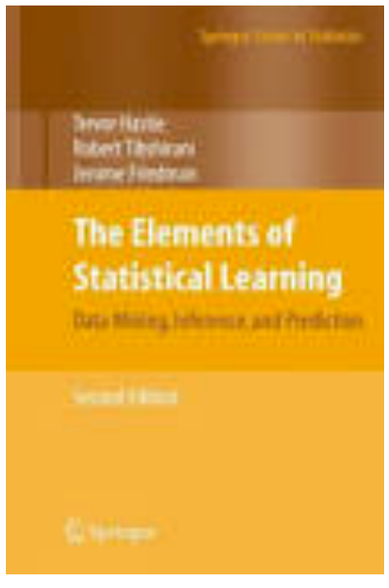
Learning from large-scale training sets is challenging
(optimization can take days or weeks)

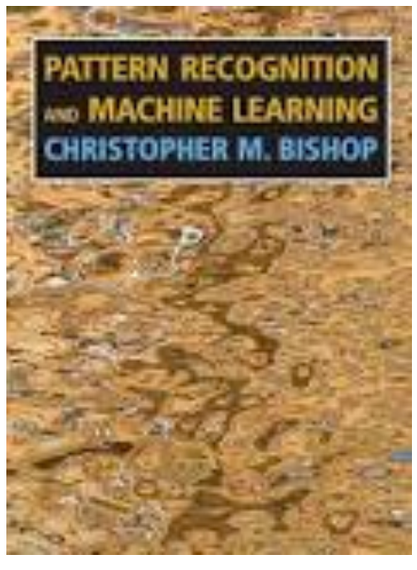
Statistical classification (30 min overview!)

Why?

This is the world in which we live - statistical models from data overpower classic “hand-designed” models

Good texts:



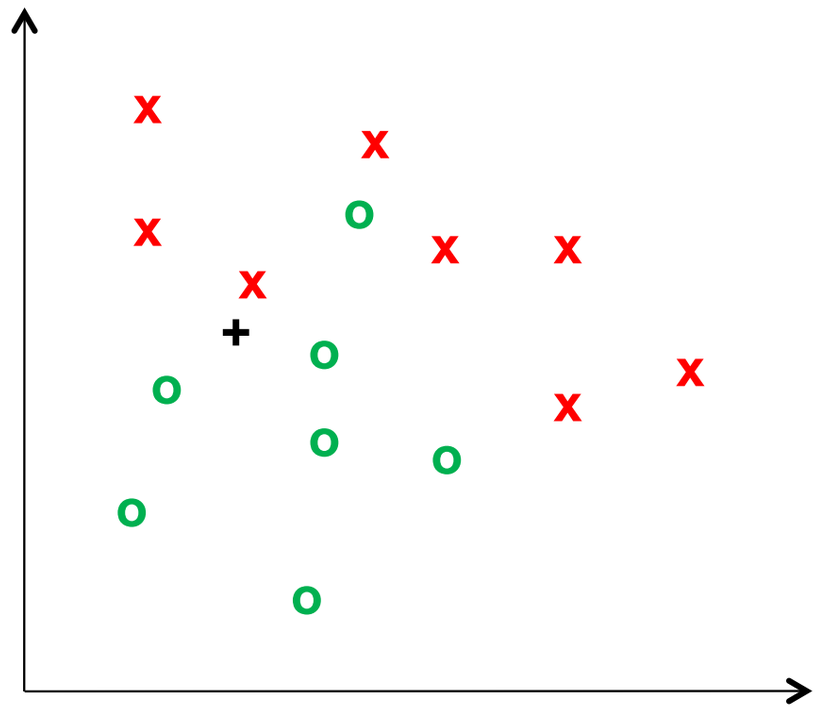


Statistical classification

$x \in R^N$

$y \in \{-1, 1\}$

$\{(x_i, y_i)\}$



Recall N ~ 10x5x9 for typical HOG templates

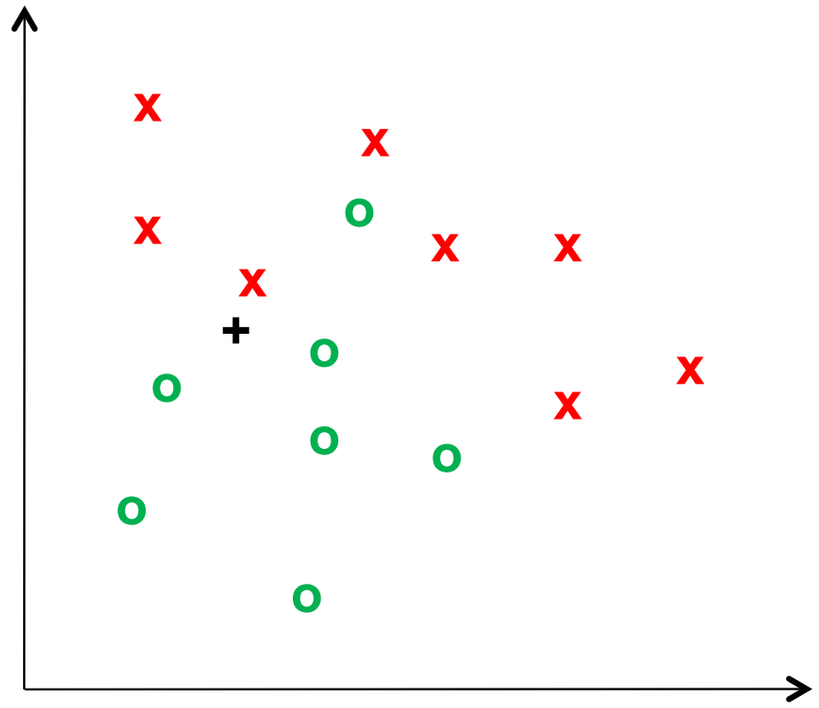
Given training points (xi,yi), learn function f(x) that predicts a label {-1,1}

Statistical classification

$x \in R^N$

$y \in \{-1, 1\}$

$\{(x_i, y_i)\}$



Version 0: nearest neighbor classification

Train time:0
Test time: expensive

Trivially handles classes

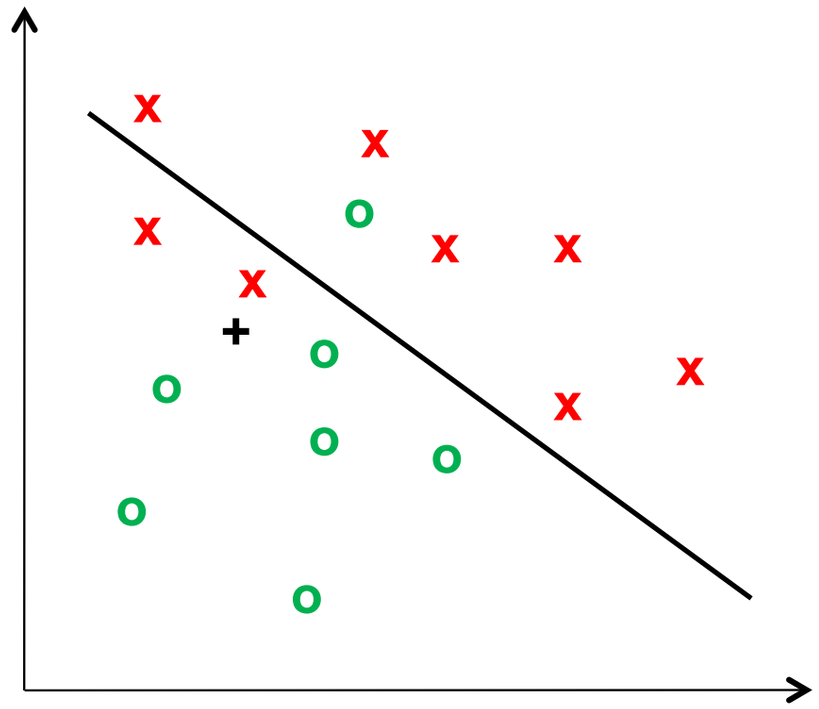
Surprisingly powerful!

Statistical classification

$x \in R^N$

$y \in \{-1, 1\}$

$\{(x_i, y_i)\}$

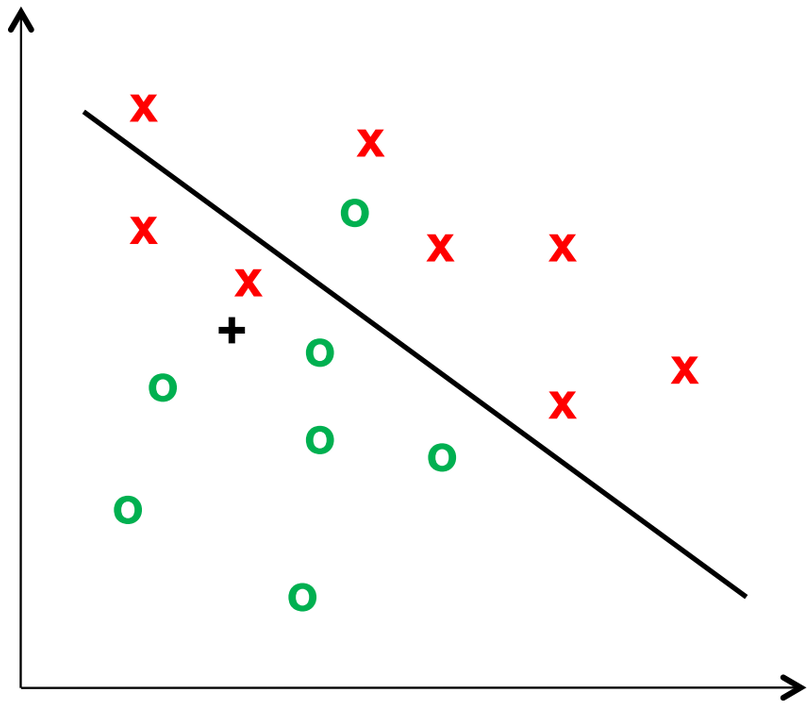


Given training points (xi,yi), learn function f(x) that predicts a label {-1,1}

Version 0: nearest neighbor classification

Version 1: linear classification

What's the best line?



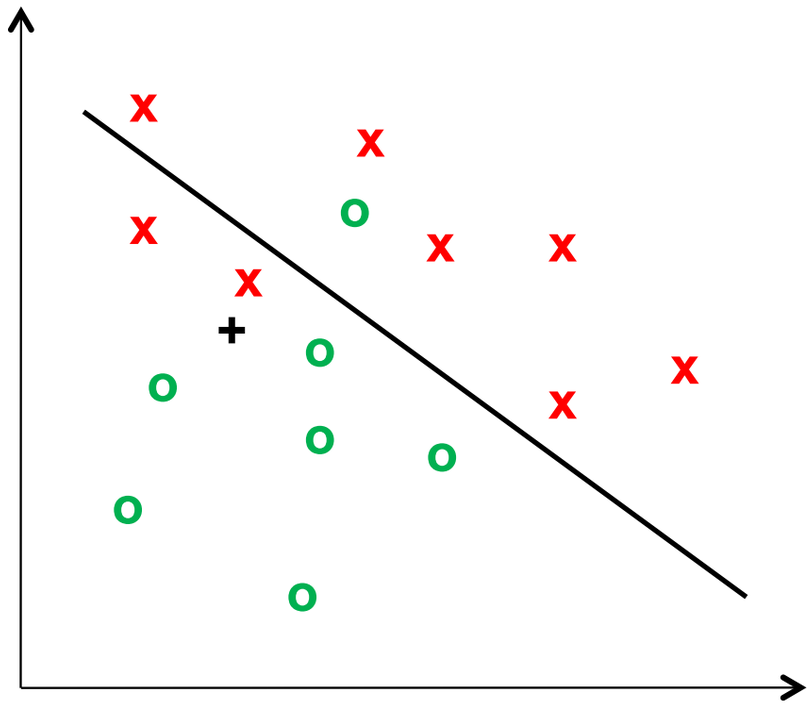
$$\min_{w} \sum_{i} [y_i \neq thresh(w \cdot x_i)]$$

Find w that minimizes mistakes on training data

[Hard to optimize]

Aside: doesn't this restrict line to go through origin?

What's the best line?

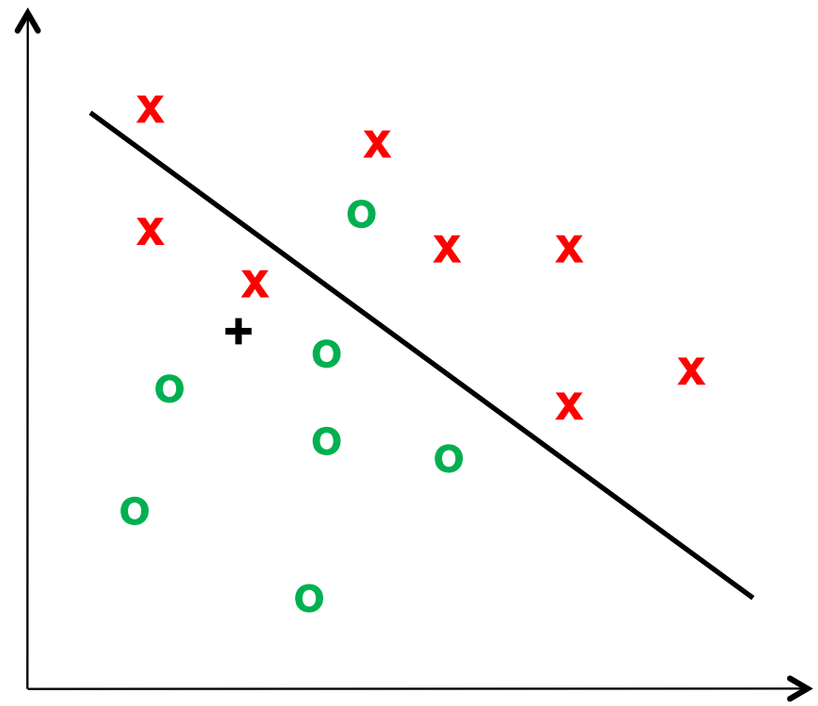


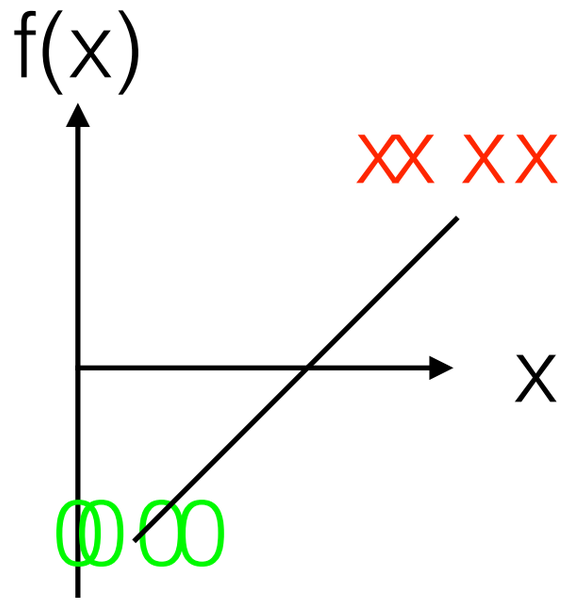
$$\min_w \frac{\lambda}{2}||w||^2 + \sum_i (y_i - w \cdot x_i)^2$$

Easy to optimize - least squares!

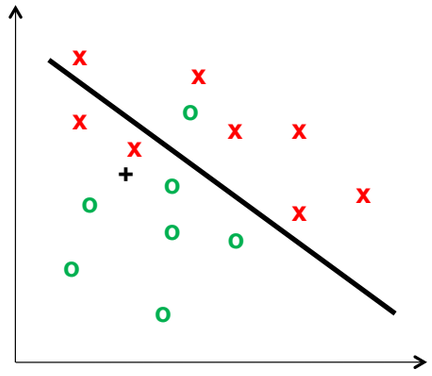
The first term is a *regularizer* (prevents overfitting and makes optimization easier)

Alternate visualization: heightfeild





Unified notation: regularized loss minimization



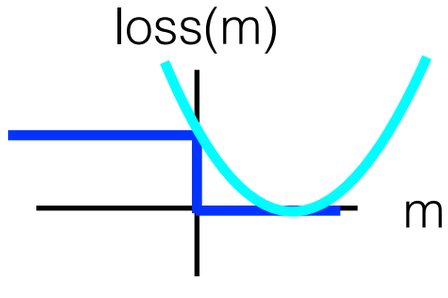
$$\min_w \lambda R(w) + \sum_i loss(y_i f_w(x_i))$$

$$R(w) = \frac{1}{2}||w||^2$$

$$f_w(x_i) = w \cdot x_i$$

$$loss_{squared}(m) = (1-m)^2$$

$$loss_{01}(m) = I(m < 0)$$



Birds-eye view of ML

$$\min_w \lambda R(w) + \sum_i loss(y_i f_w(x_i))$$

$$R_{L2}(w) = \frac{1}{2}||w||^2 \quad \text{(L2 regularization)}$$

$$R_{L1}(w) = \sum_j |w_j| \quad \text{(L1/sparse regularization)}$$

$$loss_{squared}(m) = (1-m)^2 \quad \text{(Linear regression)}$$

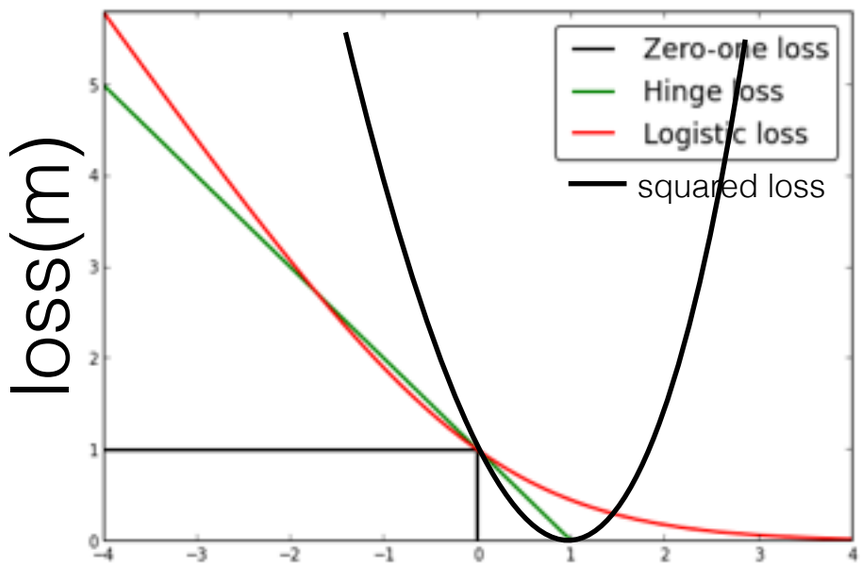
$$loss_{log}(m) = \log(1+e^{-m}) \quad \text{(Logistic regression)}$$

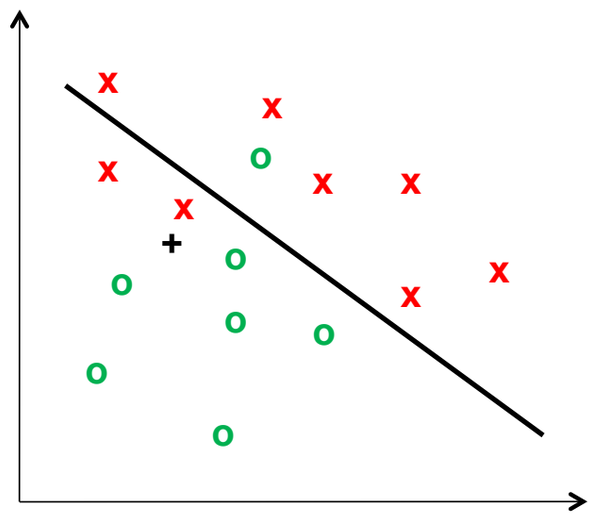
$$loss_{hinge}(m) = \max(0, 1+m) \quad \text{(Support vector machine)}$$

$$f_w(x_i) = w^T x_i \quad \text{Linear classifier}$$

$$f_w(x_i) = CNN_w(x_i) \quad \text{Nonlinear classifier}$$

Learning with losses





$$loss_{01}(m) = I(m < 0)$$

$$loss_{squared}(m) = (1 - m)^2$$

$$loss_{log}(m) = \log(1 + e^{-m})$$

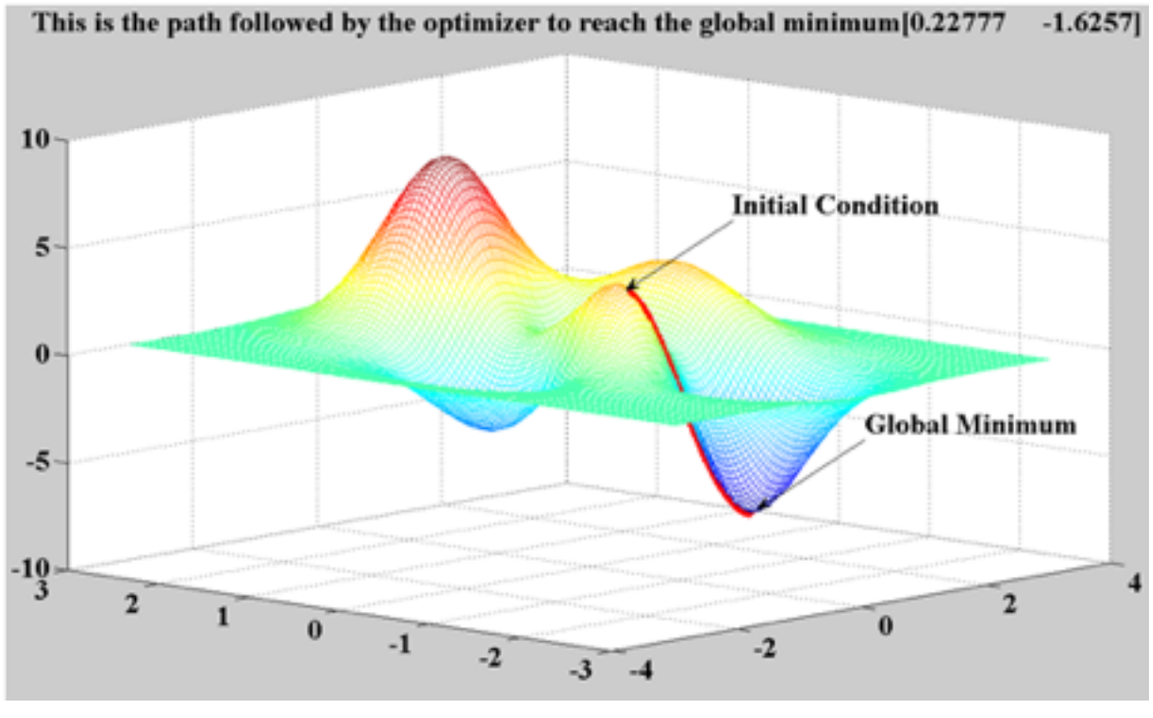
$$loss_{hinge}(m) = \max(0, 1 + m)$$

Bottom line: other than 01 and squared loss, most loss functions look similar (to me)

Learning with gradient descent

$$\min_w L(w) \quad \text{where} \quad L(w) = \frac{1}{2}||w||^2 + \sum_i (f_w(x_i) - y_i)^2$$

$$w := w - step * (w + \sum_i (f_w(x_i) - y_i) \frac{\partial f_w(x_i)}{\partial w})$$

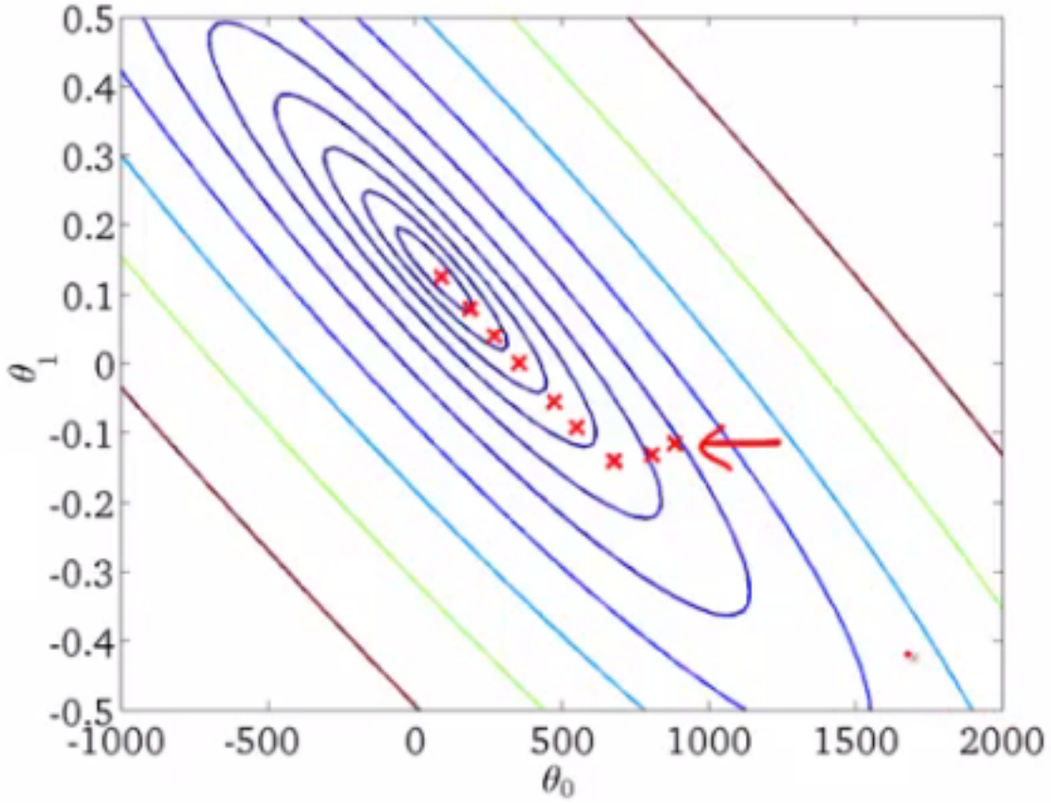


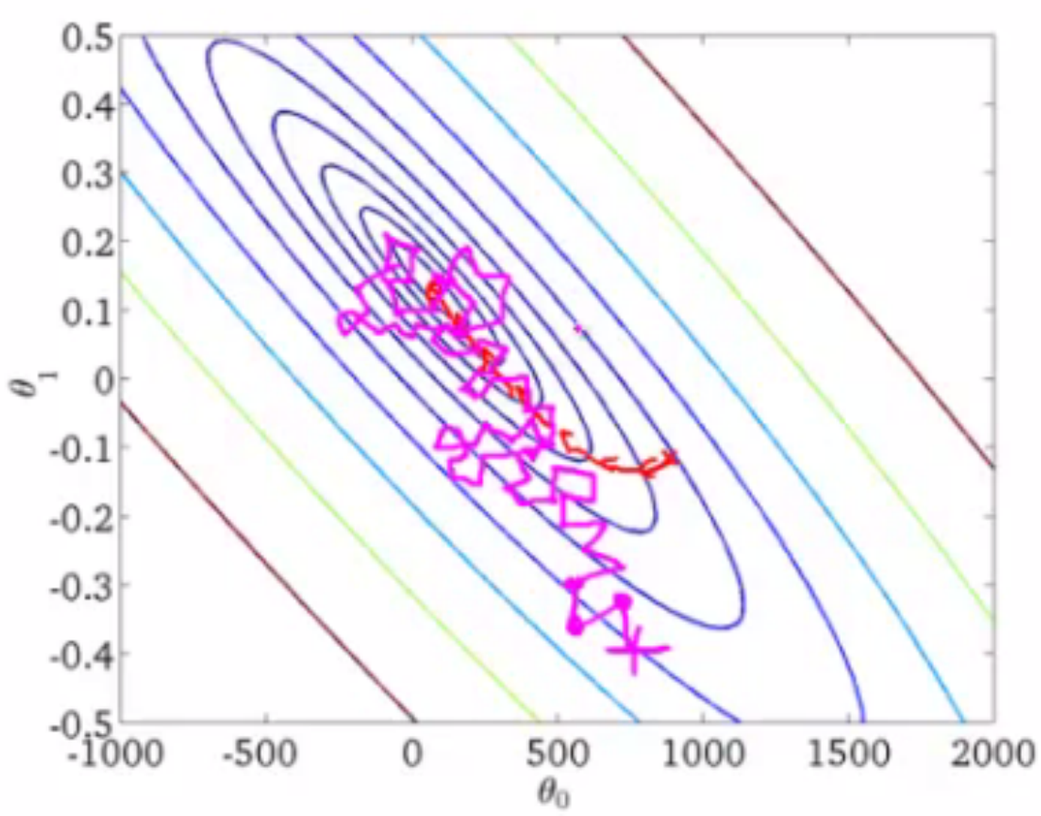
Early 2000's: obsession with convex L(w)

Large-scale learning: stochastic gradient descent

$$\min_w \frac{1}{2}||w||^2 + \sum_i (f_w(x_i) - y_i)^2$$

$$w := w - step * \left(w + (f_w(x_i) - y_i)\frac{\partial f_w(x_i)}{\partial w}\right)$$





Trivially “out-of-core”: most contemporary models are trained in this manner

Special case: SVMs

$$\min_w L(w) \quad \text{where} \quad L(w) = \frac{\lambda}{2}||w||^2 + \sum_i \max(0, 1 - y_i w \cdot x_i)$$

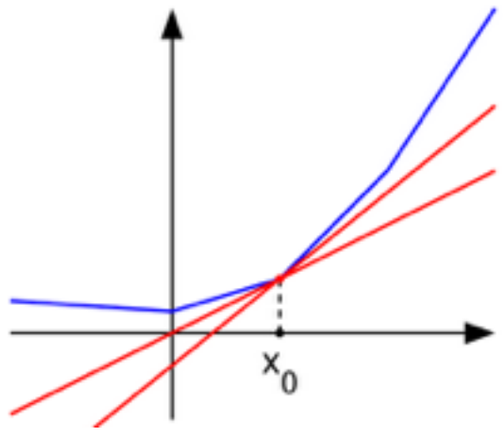
w appears quadratically (sort of)… can we differentiate and set = 0?

[We'll set lambda = 1 to simplify notation]

SVMs

$$\min_w L(w) \quad \text{where} \quad L(w) = \frac{1}{2}||w||^2 + \sum_i \max(0, 1 - y_i w \cdot x_i)$$

w appears quadratically (sort of)… can we differentiate and set = 0?

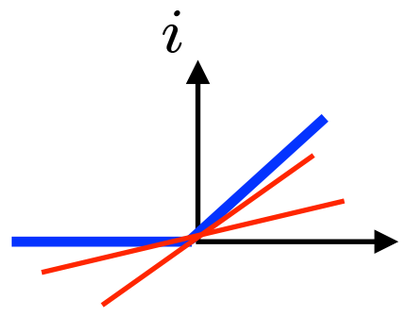


$$\frac{\partial \max(0, w)}{\partial w} \approx \alpha w, \quad 0 \leq \alpha \leq 1$$

SVMs

$$\min_w L(w) \quad \text{where} \quad L(w) = \frac{1}{2}||w||^2 + \sum_i \max(0, 1 - y_i w \cdot x_i)$$

$$w = \sum \alpha_i y_i x_i$$



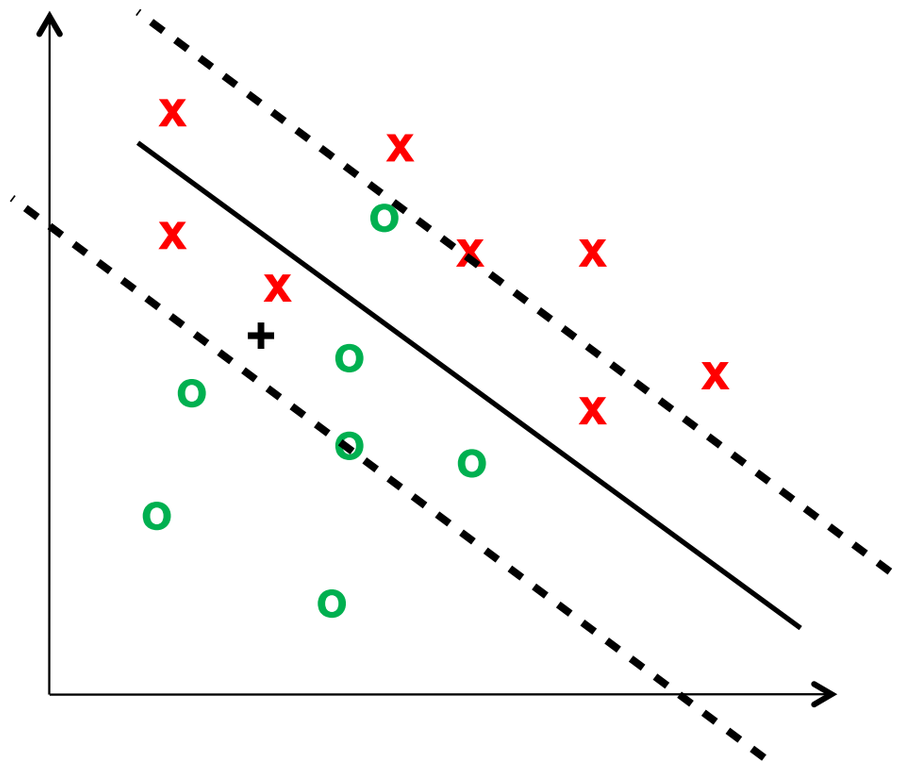
$$Easy: \quad y_i w \cdot x_i \geq 0 \Rightarrow \alpha_i = 0$$

$$Marg: \quad y_i w \cdot x_i = 0 \Rightarrow 0 \leq \alpha_i \leq 1$$

$$Hard: \quad y_i w \cdot x_i = 1 \Rightarrow \alpha_i = 1$$

"KKT conditions"

SVMs

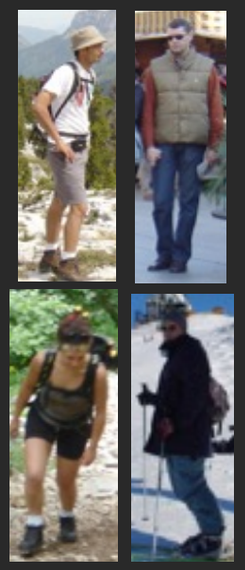


Easily visualize easy, hard, and marginally hard examples

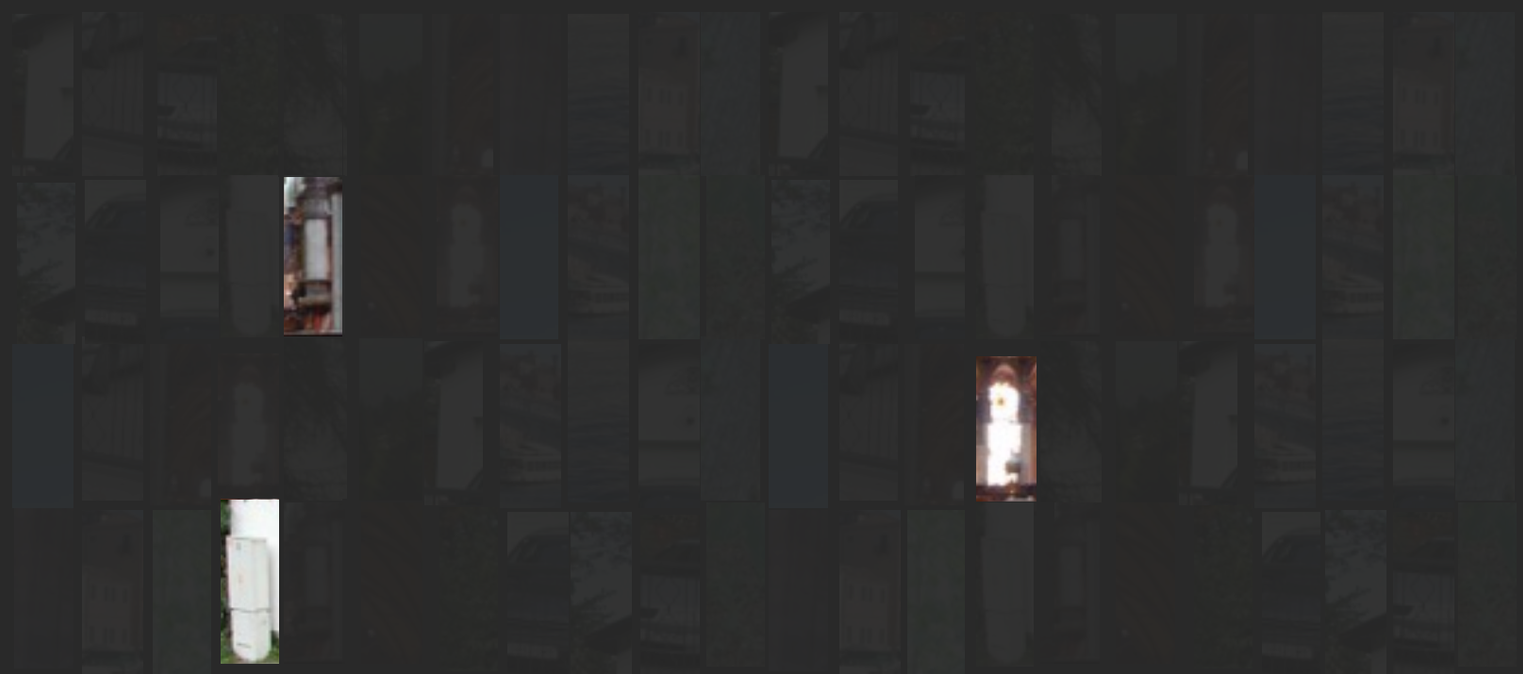
Throwing away easy examples does not change optimization

Large-scale learning

pos



neg



Our test set distribution is highly imbalanced; so should be the training set

(hundreds of positives, hundreds of millions of negatives)

Support vector machines (SVMs) are attractive because they generate sparse learning problems

(turn continuous search over parameters into combinatorial search over data)

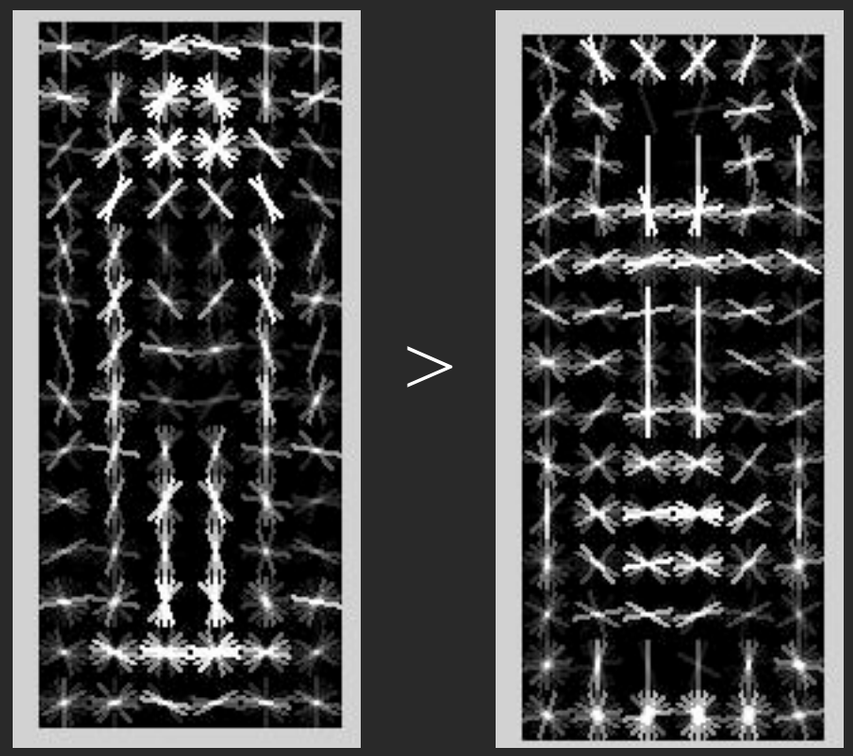
How to interpret positive and negative weights?

$$w \cdot x > 0$$

$$(w_{pos} - w_{neg}) \cdot x > 0$$

$$w_{pos} \cdot x > w_{neg} \cdot x$$

Pedestrian template



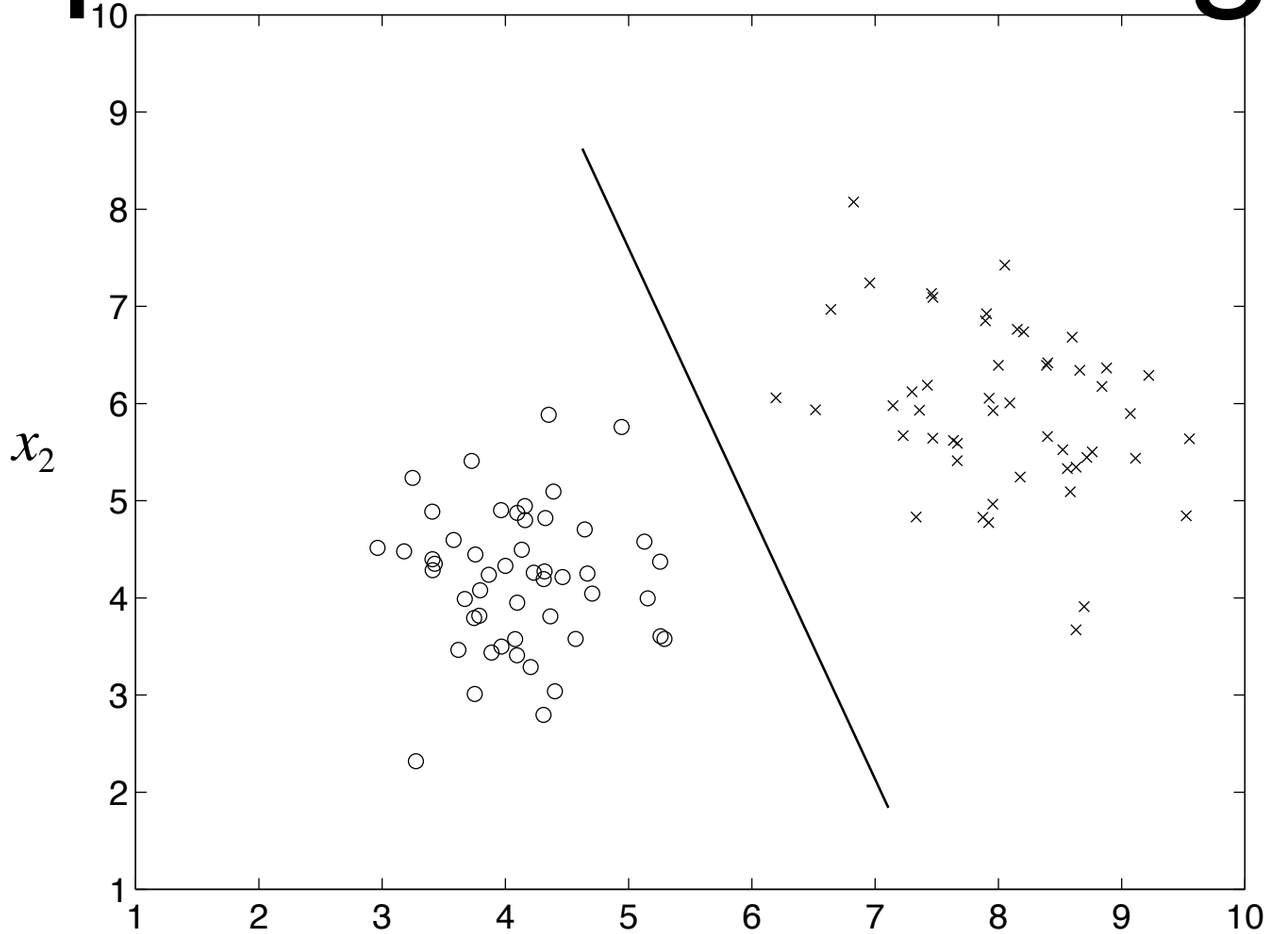
Pedestrian background template

Right approach is to compete pedestrian, pillar, doorway... models

Background class is hard to model - easier to penalize particular vertical edges

Historically, model-based approaches tend not to model negative set

But what if we actually try to model positives and negatives?



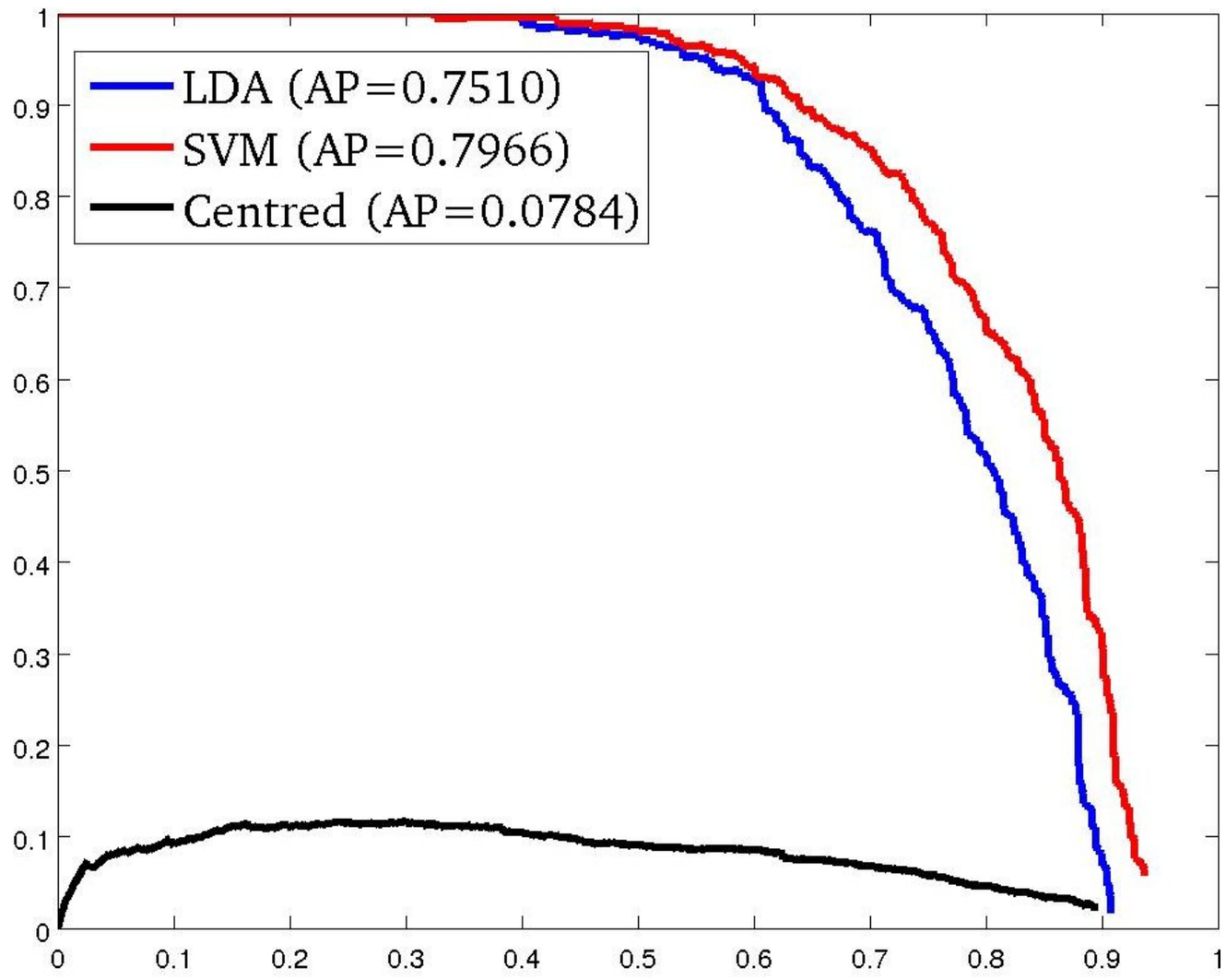
Simple generative model P(x,y) = P(y) P(x|y):

$$y \in \{-1, 1\}$$

$$x \in R^N$$

1. P(y): Flip (biased) coin to select class ‘k’
2. P(x|y): Sample from Gaussian (mu_k, Sigma_k)

A look ahead



(a) AP

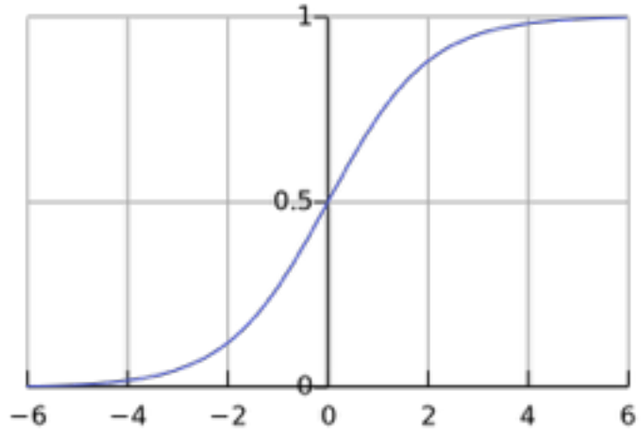
Simple linear discriminant analysis gets us 90% of the way there…

Class-conditional Gaussians

If we assume all classes have same Sigma….

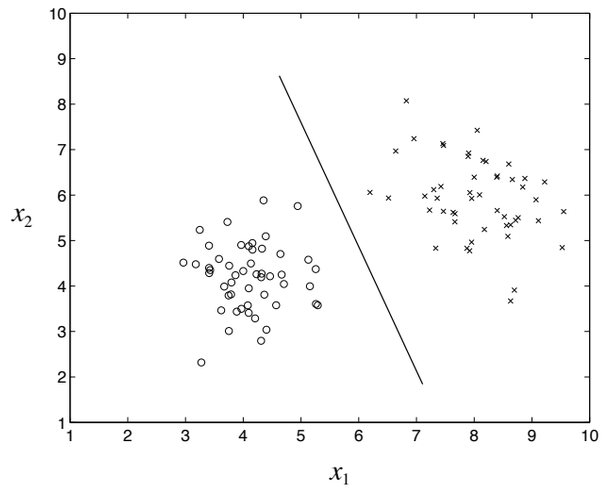
(For notational ease, let's assume priors for class y in {0,1} are equal)

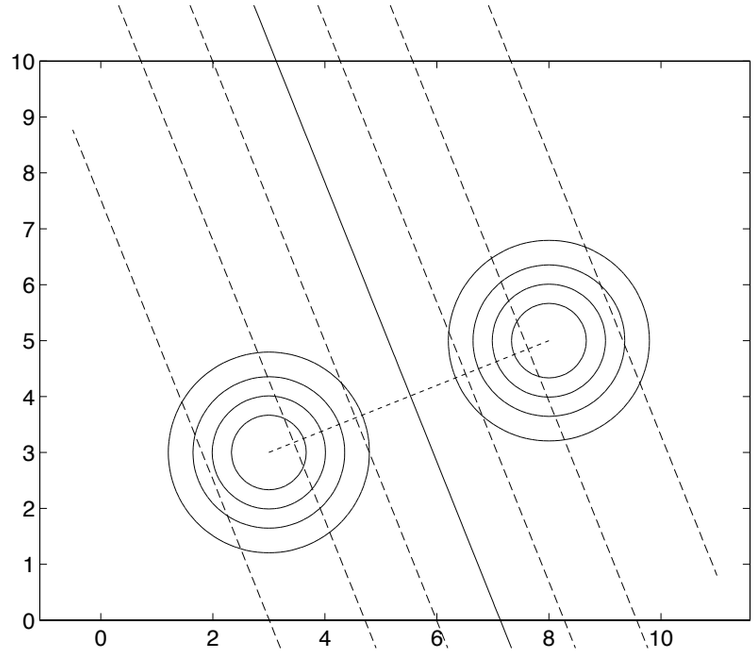
$$
\begin{aligned}
p(y=1|x) &= \frac{p(x|y=1)p(y=1)}{p(x|y=1)p(y=1)+p(x|y=0)p(y=0)} \\
&= \frac{\exp\left\{-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right\}}{\exp\left\{-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\right\}+\exp\left\{-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0)\right\}} \\
&= \frac{1}{1+\exp\left\{-(\mu_1-\mu_0)^T\Sigma^{-1}x-b\right\}} \\
&= \frac{1}{1+e^{-f_w(x)}}
\end{aligned}
$$



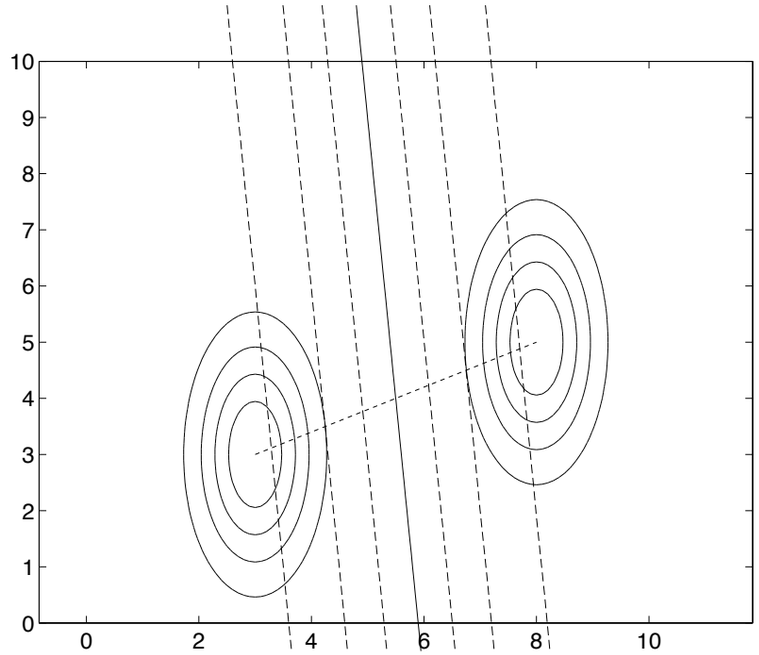
$$p(y=1|x) > .5 \quad \text{when} \quad f_w(x) = w^T x + b > 0, \quad w = \Sigma^{-1}(\mu_1 - \mu_0)$$

Class-conditional Gaussians





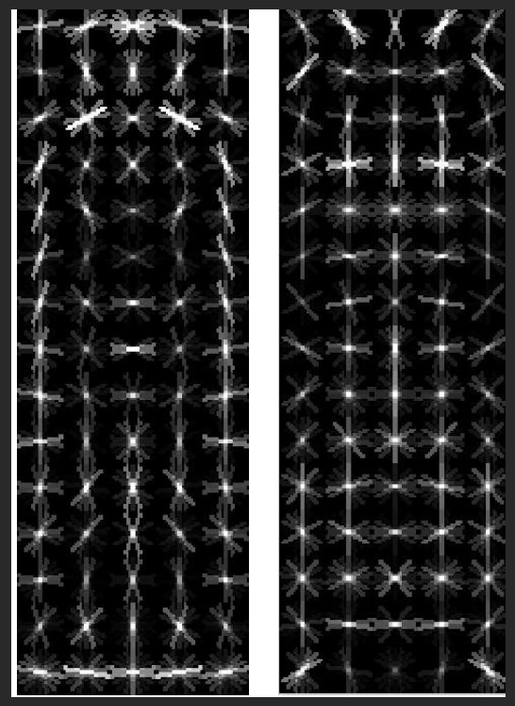
(a)



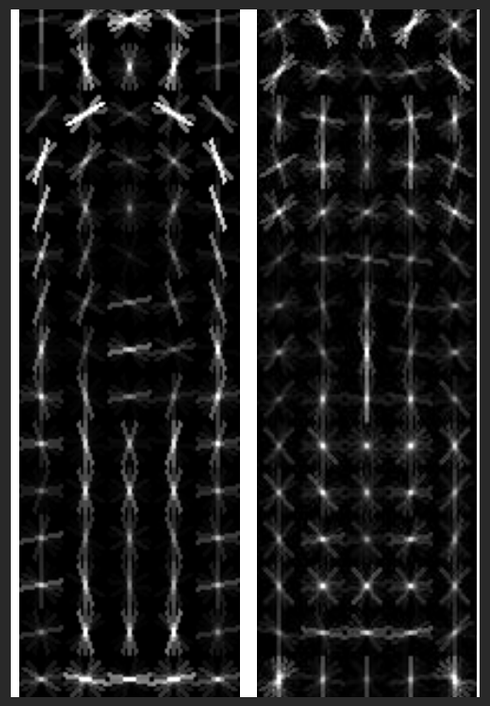
(b)

What happens if Sigmas for the two classes are different?

LDA: a closer look

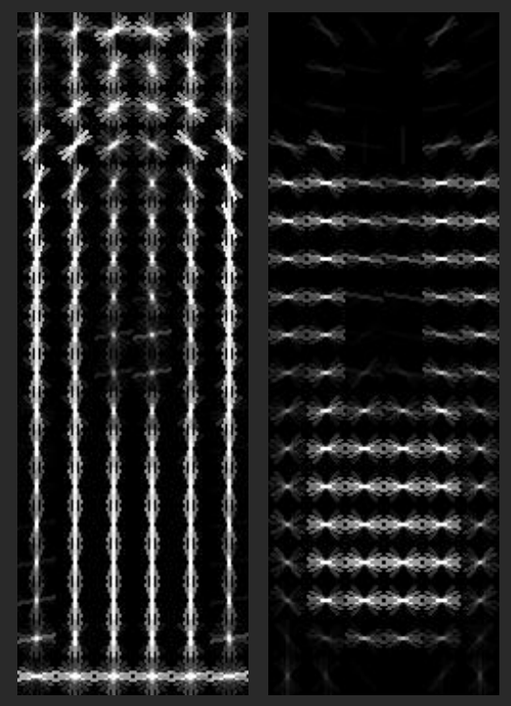


SVM



Gaussian model

$$w = \Sigma^{-1}(\mu_1 - \mu_0)$$



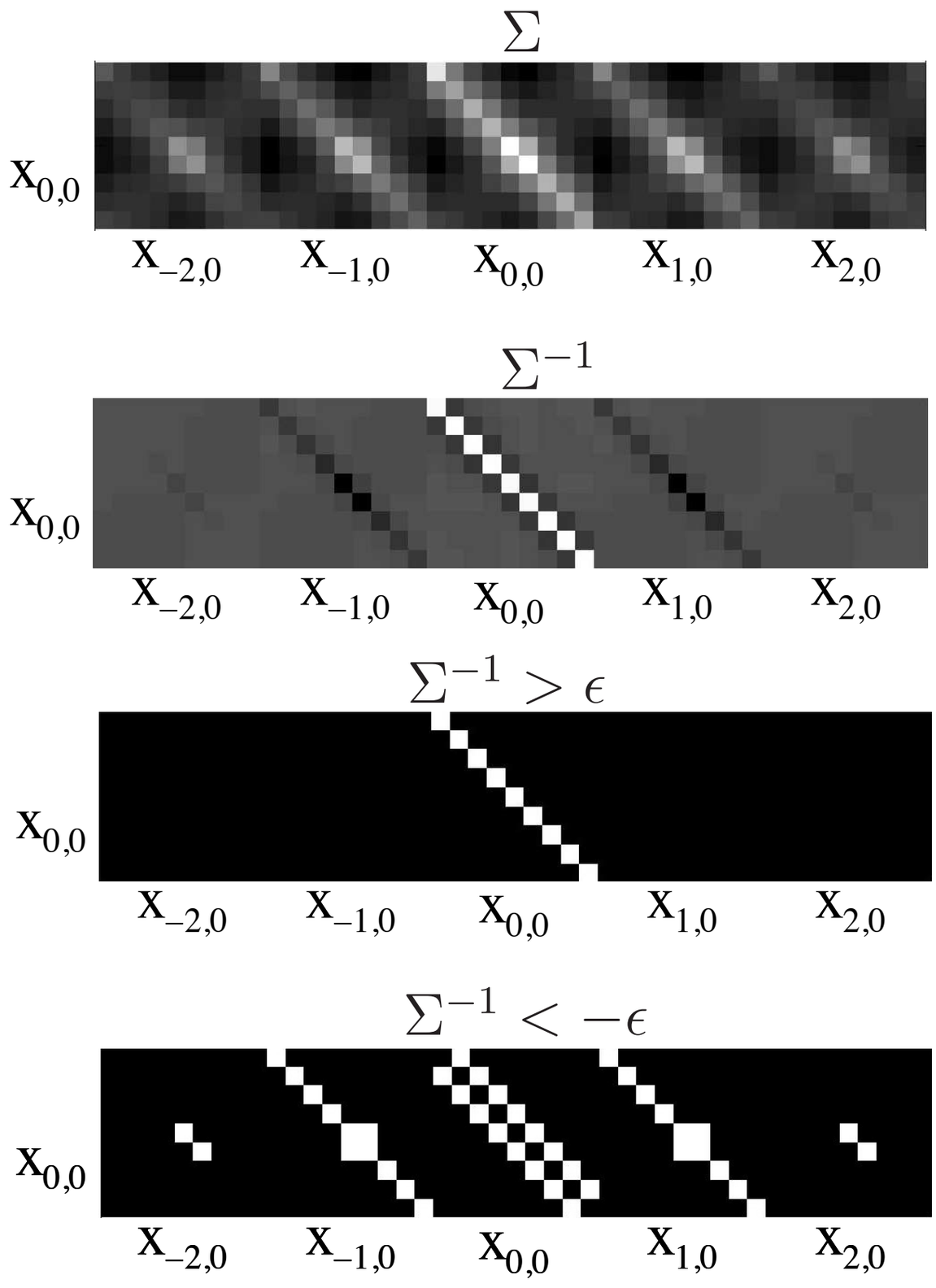
Centered model

$$w = \mu_1 - \mu_0$$

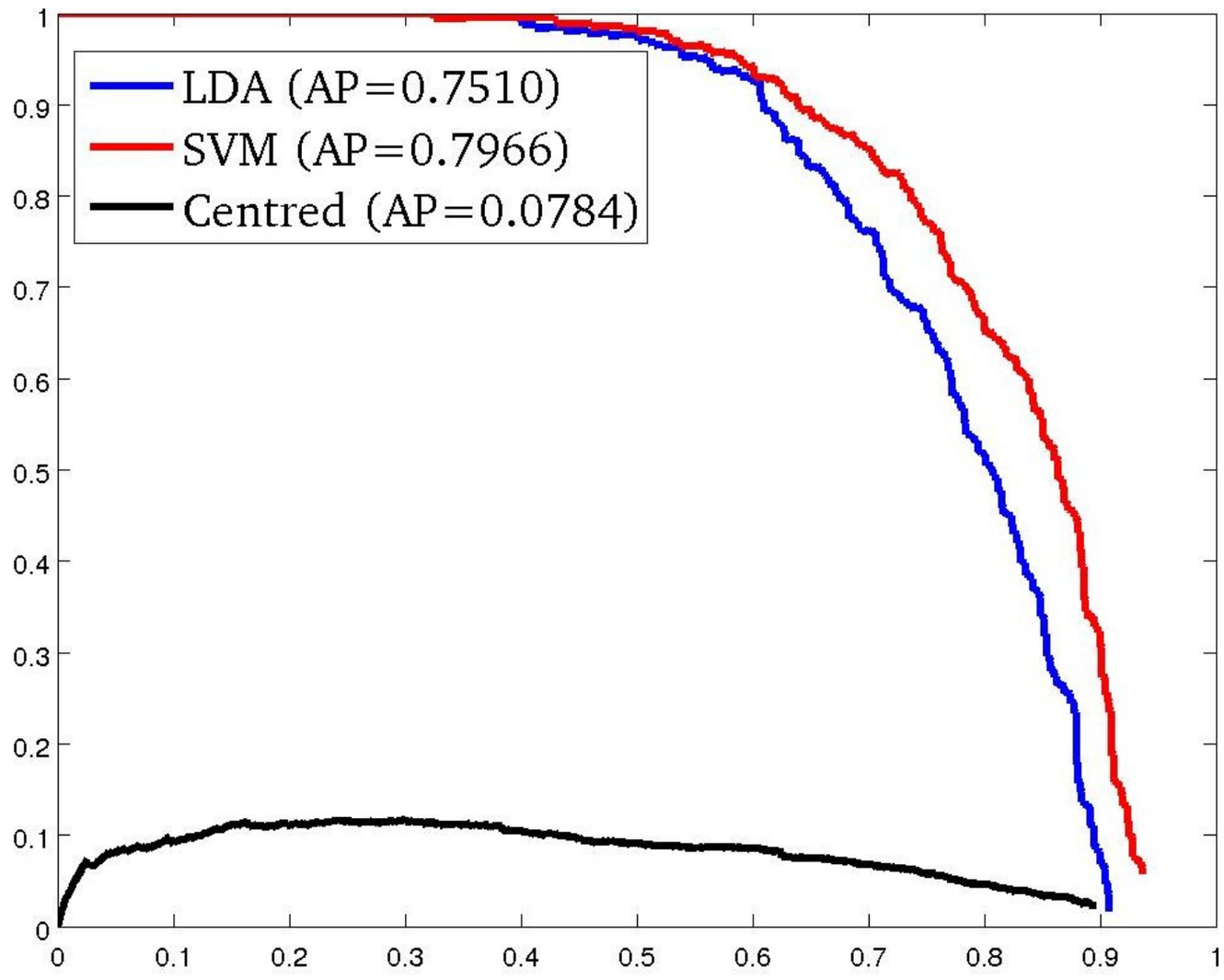
Learn templates with generic (de)correlation model

Hariharan, Malik, Ramanan ECCV 12

What's the covariance matrix capturing?



The numbers



(a) AP

Simple linear discriminant analysis gets us 90% of the way there…

Parting thoughts on LDA

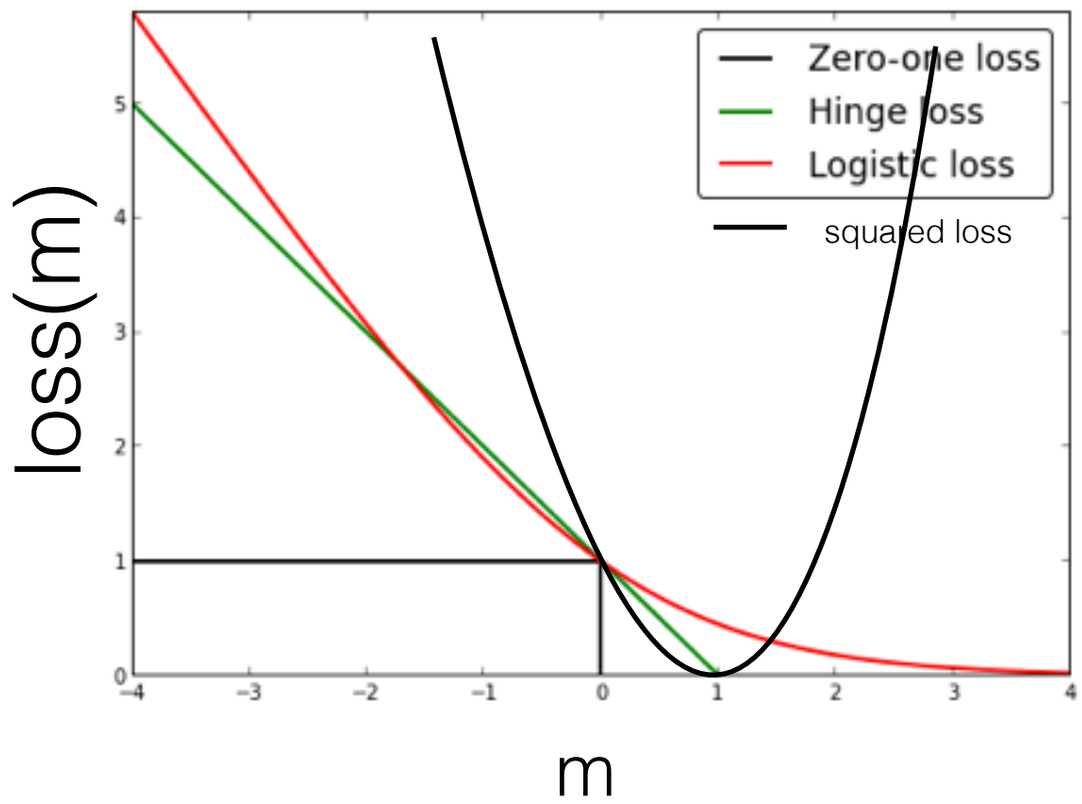
1. Fischer Discriminant Analysis when derived using other criteria (maximizing ratio of between to within-class variances)

2. Can be easily generalized to multiple classes - how?

2. One can also obtain the LDA / FDA solution by discriminative learning with a squared error loss

(Hastie et al, Elements of Statistical Learning)

Implies the distinction between generative and discriminative models can be blurred…



Parting thoughts on statistical classification

Loss functions: hinge, log-loss, squared loss

SVMs generate sparse optimization problems

Generative models are promising, but current state-of-the-art relies on discrimative loss minimization

Things that will appear later:

Cross-entropy loss: minimizing cross-entropy of binary prediction is equivalent to log-loss

Soft-max loss: minimizing cross-entropy of K-way prediction is equivalent to soft-max

Outline

- Classifiers
- Apearance variation
- Parts

Back to vision….

variation in illumination

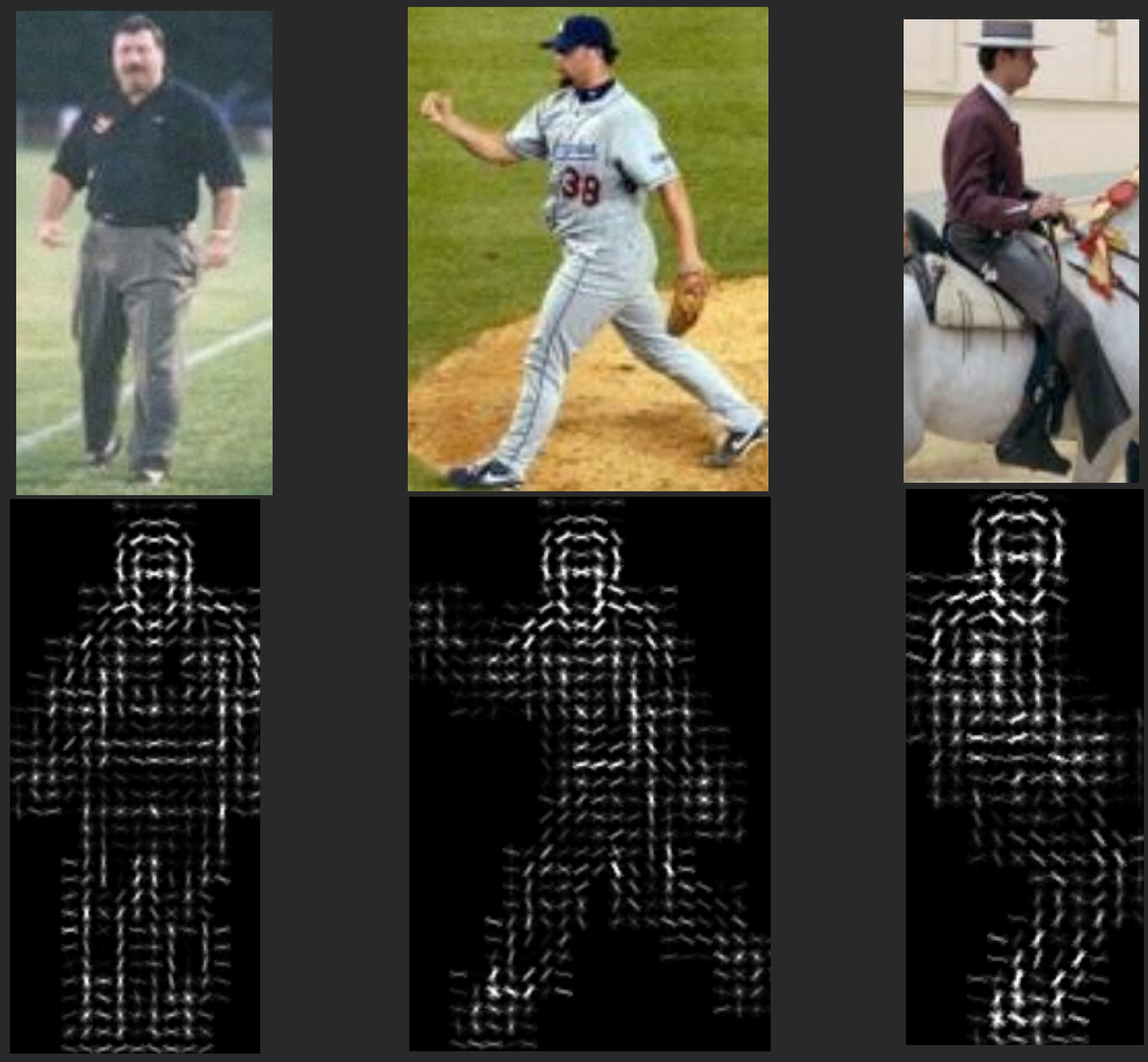
variation in appearance

variation in pose, viewpoint

occlusion & clutter

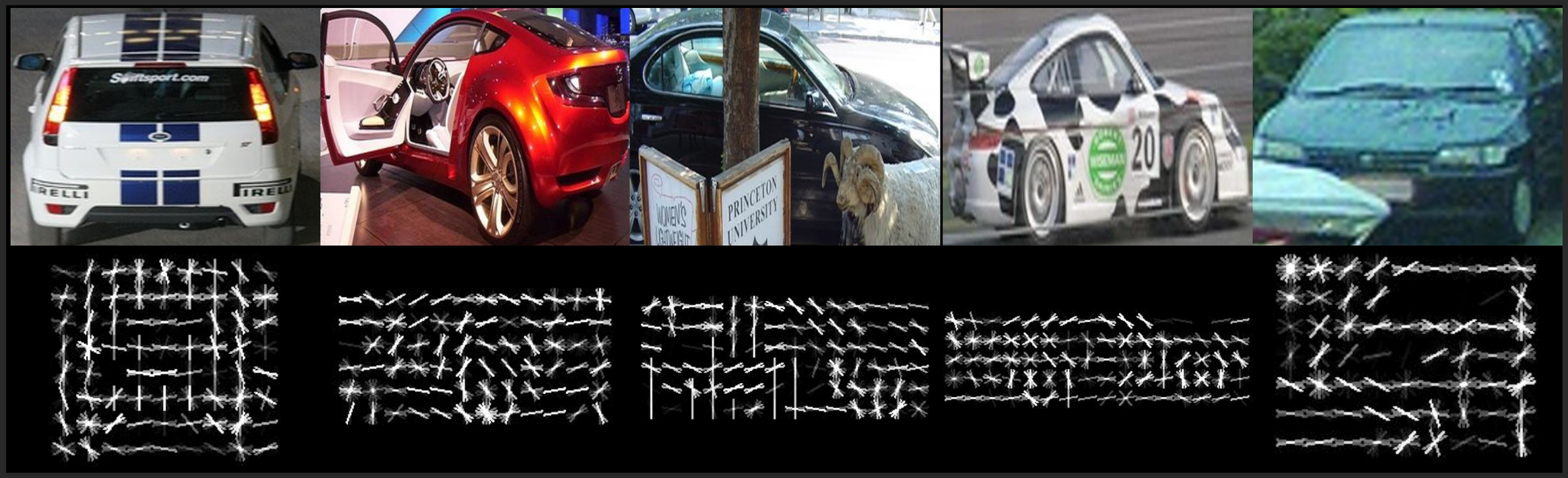
Classic “nuisance factors” for general object recognition

“Sub”categories



Train sub-category templates for each type of pose, body-shape, etc.

Also applies to other objects…



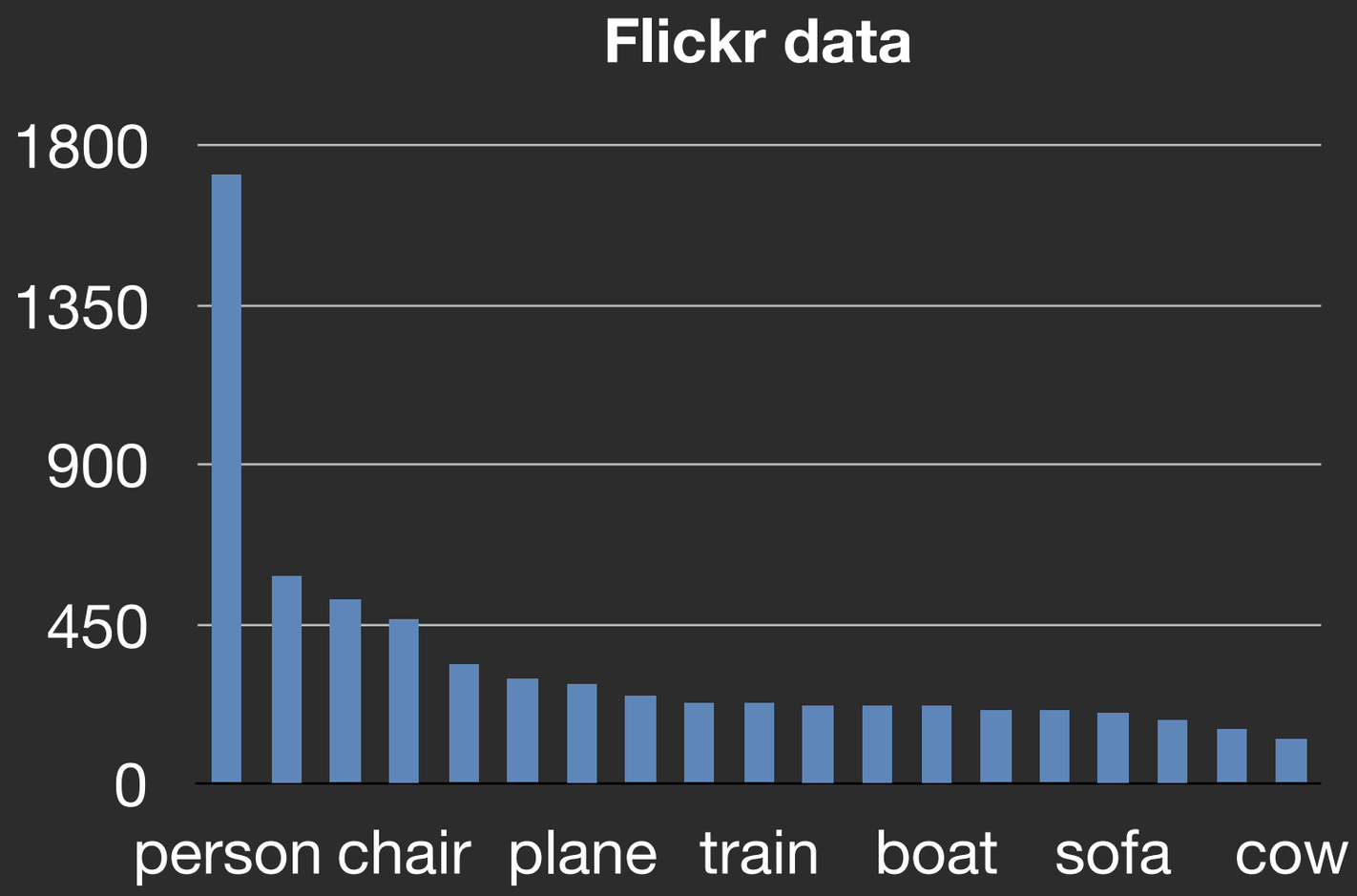
But how to handle...

We need lots of templates, but will likely have little data of ‘twisted’ poses

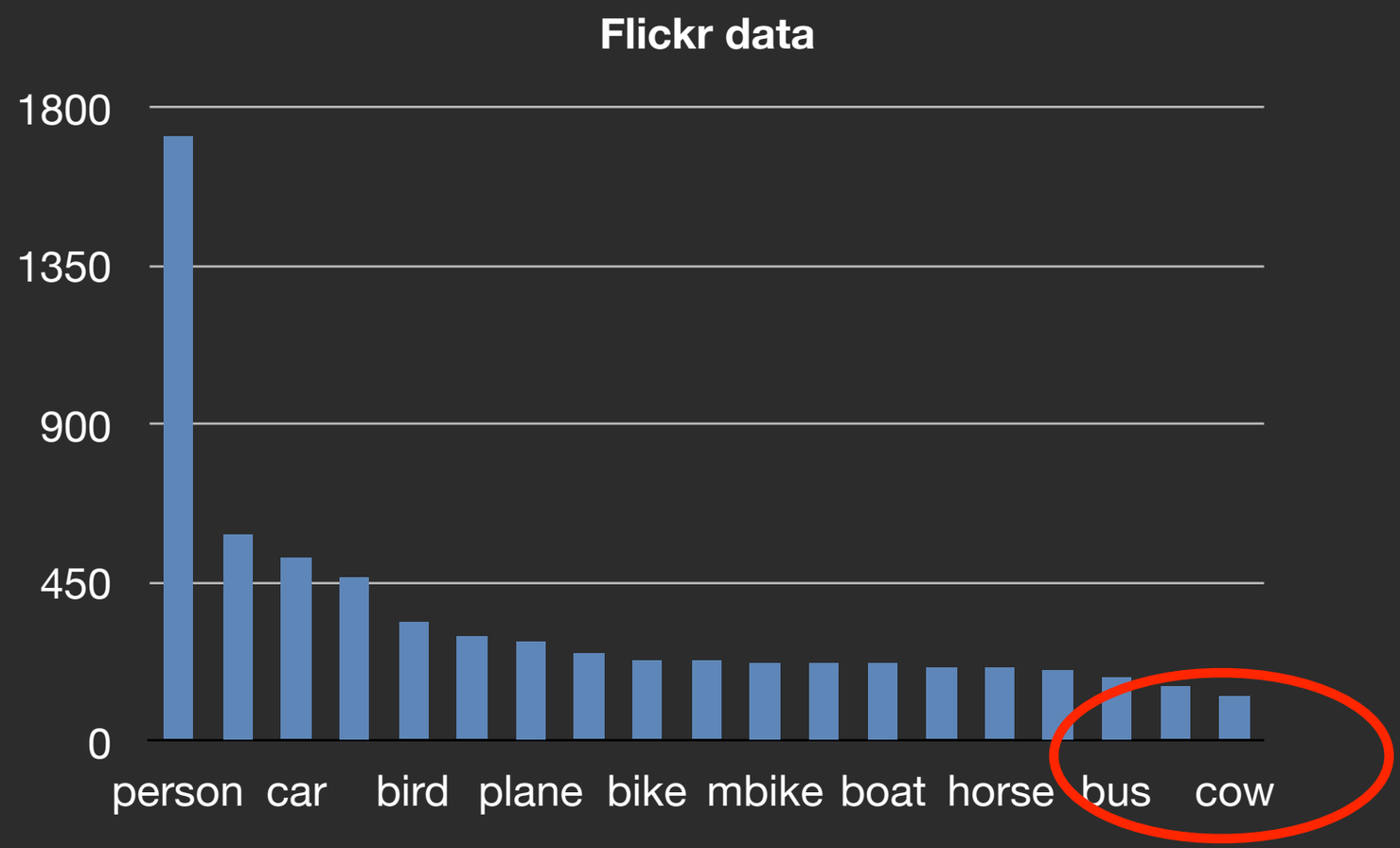
But how to handle...

We need lots of templates, but will likely have little data of ‘rare’ car-appearances

Difficulties: long tails

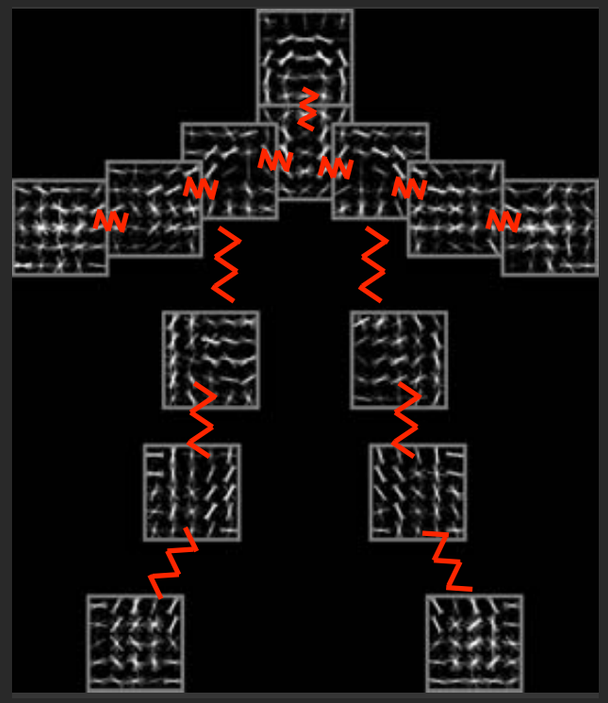


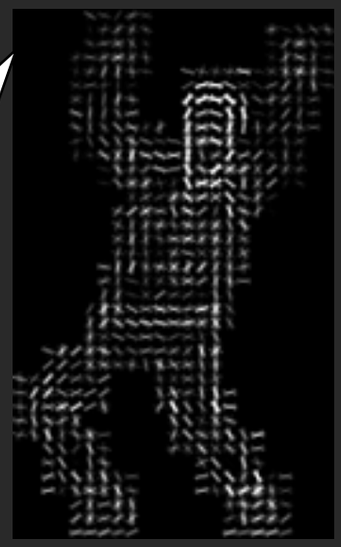
Difficulties: long tails



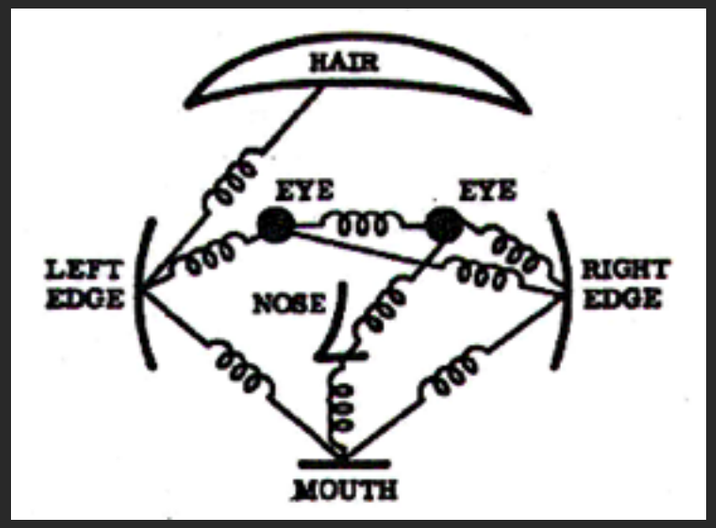
"One-shot learning": sharing

Parts to the rescue!

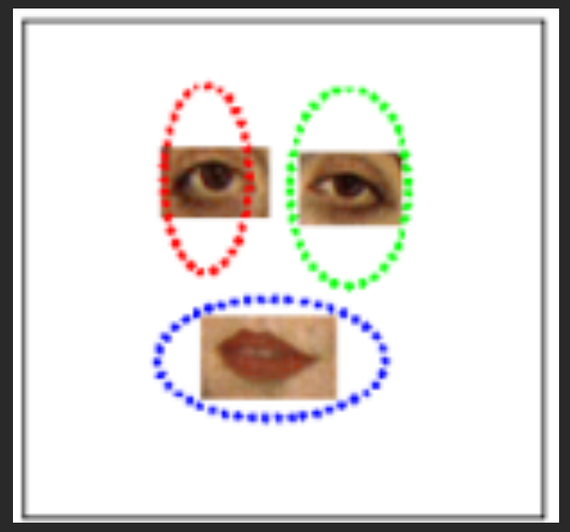




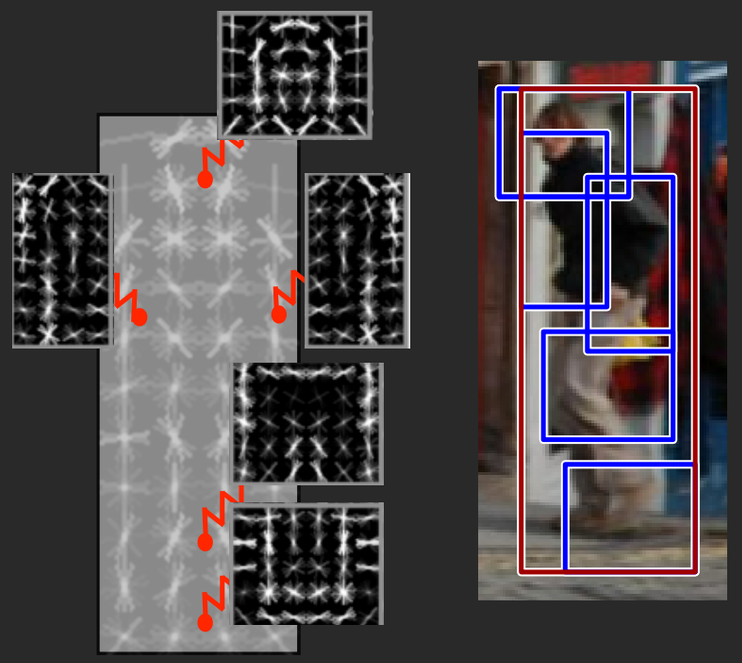
History over 40 years



Pictorial structures



Constellation models

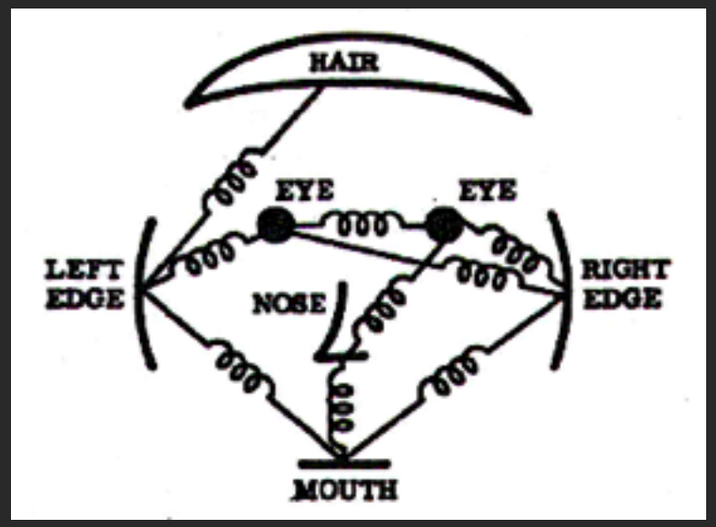


Deformable part models

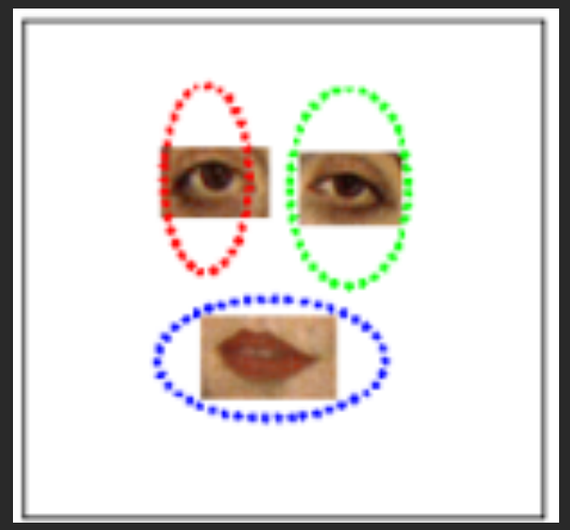
Model encodes local appearance + pairwise geometry

Pictorial Structures (Fischler & Elschlager 73, Felzenswalb and Huttenlocher 00)
Cardboard People (Yu et al 96)
Body Plans (Forsyth & Fleck 97)
Active Appearance Models (Cootes & Taylor 98)
Constellation Models (Burl et all 98, Fergus et al 03)

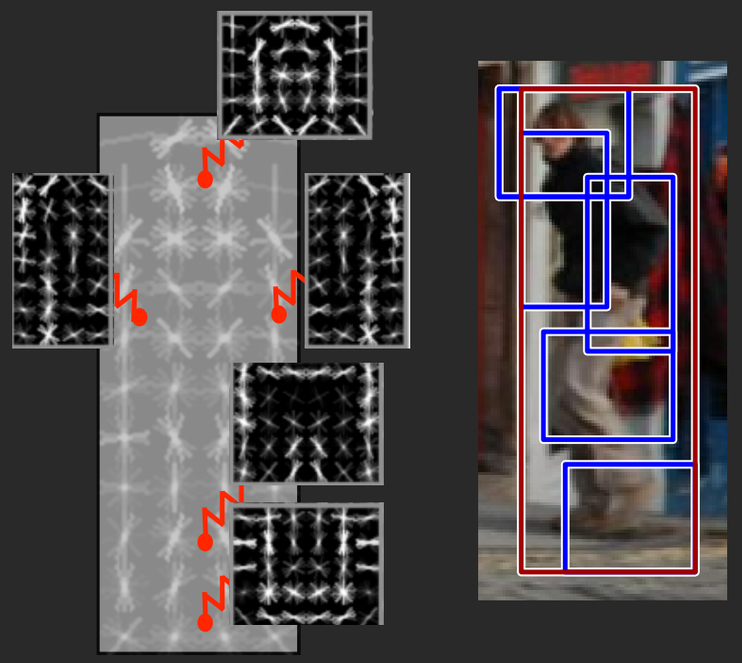
Part models



Pictorial structures



Constellation models



Deformable part models

I'll talk about DPMs, but give an alternate "long tail" perspective

Felzenszwalb, Girshick, McAllester, Ramanan CACM 2013