Visual Classification 2

CSE P576

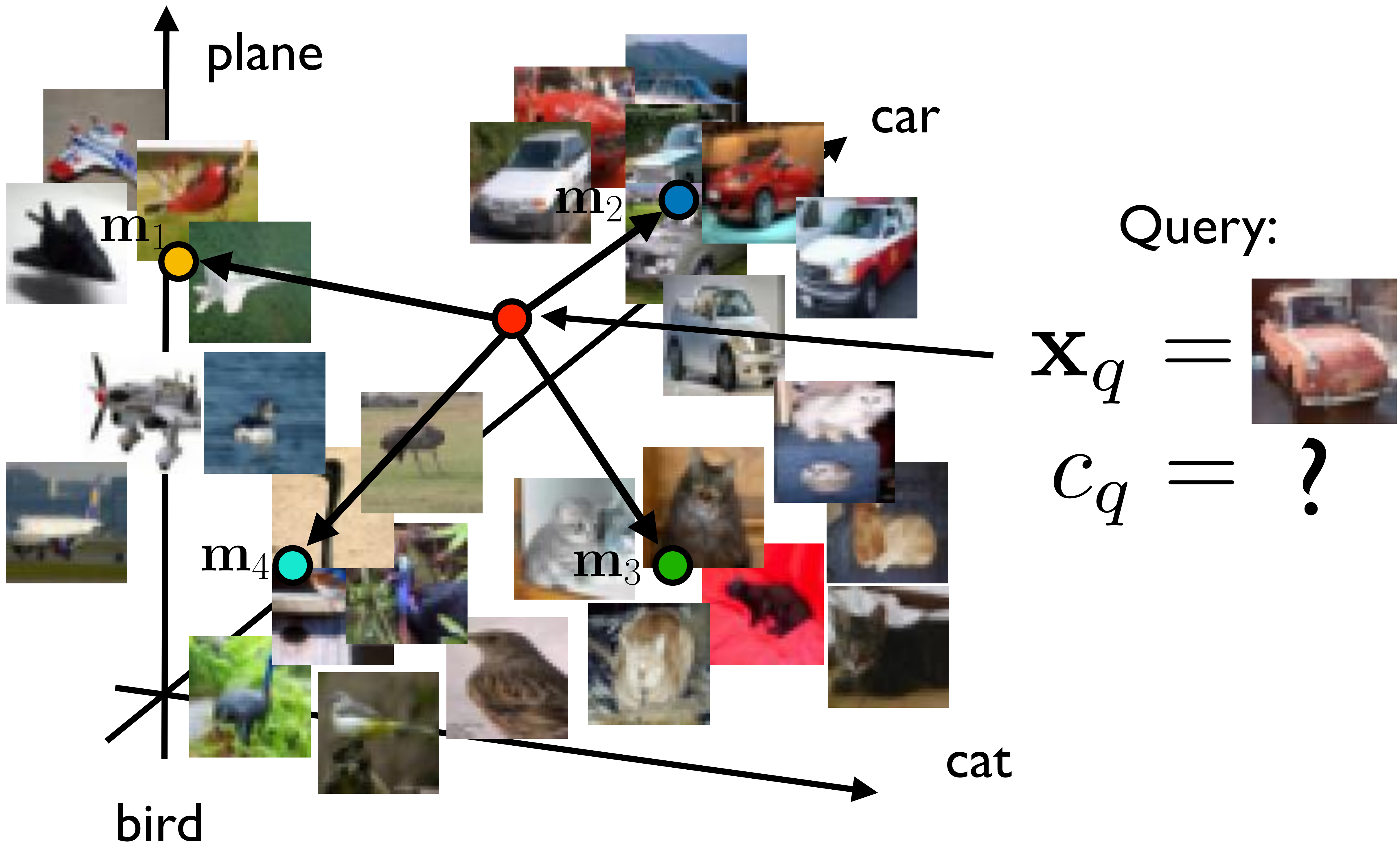
Dr. Matthew Brown

Visual Classification 2

- Fundamentals and Pre-Deep Learning
- Bayesian classifiers, Gaussian distributions, PCA, LDA
- Decision Forests, Visual words, SVMs

Nearest Mean Classification

- How about a single template per class



Nearest Mean Classification

- Find nearest mean and assign class

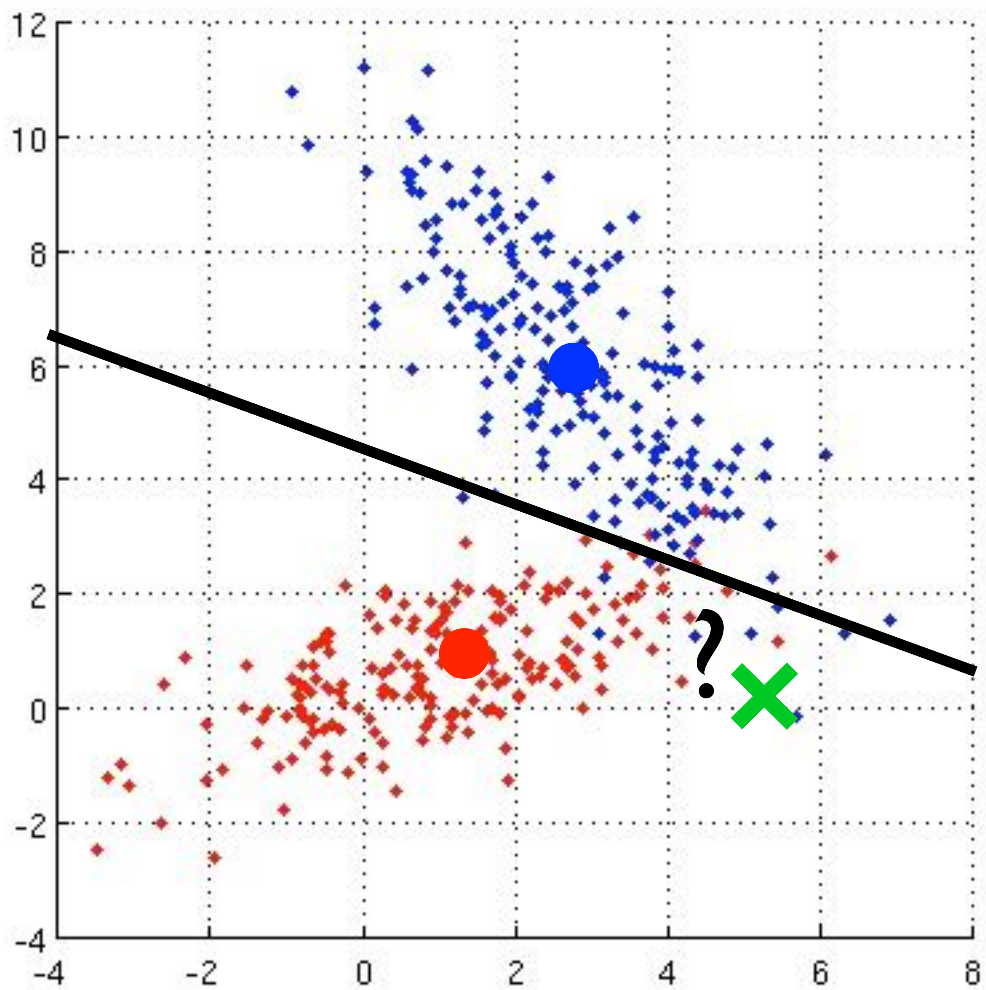
$$c_q = \arg\min_i |\mathbf{x}_q - \mathbf{m}_i|^2$$

- CIFAR 10 class means

- Can we do better?

Nearest Mean Classifier

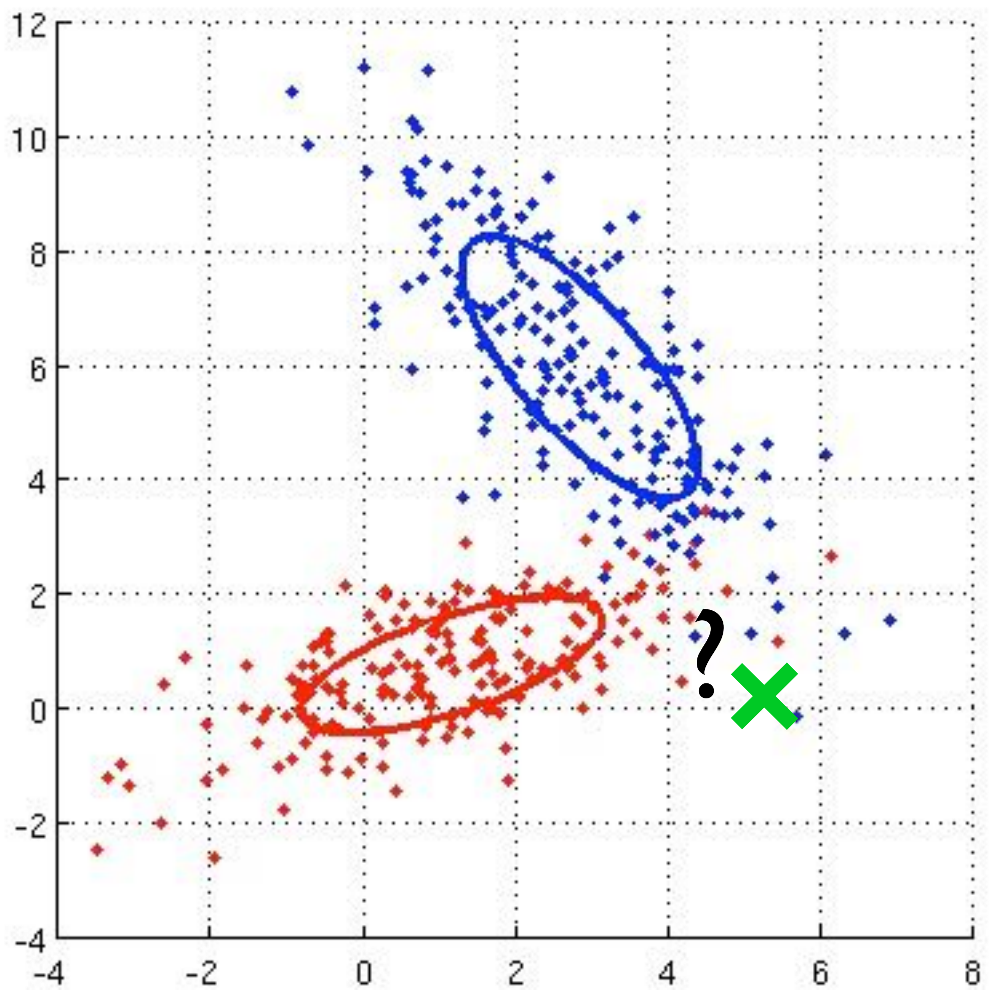
- Suppose we have 2 classes of 2-dimensional data that are not linearly separable



- A simple approach could be to assign to the class of the nearest mean
- Can we do better if we know about the data distribution?

Bayesian Classificaion

- A probabilistic view of classification models the likelihood of observing the data given a class/parameters

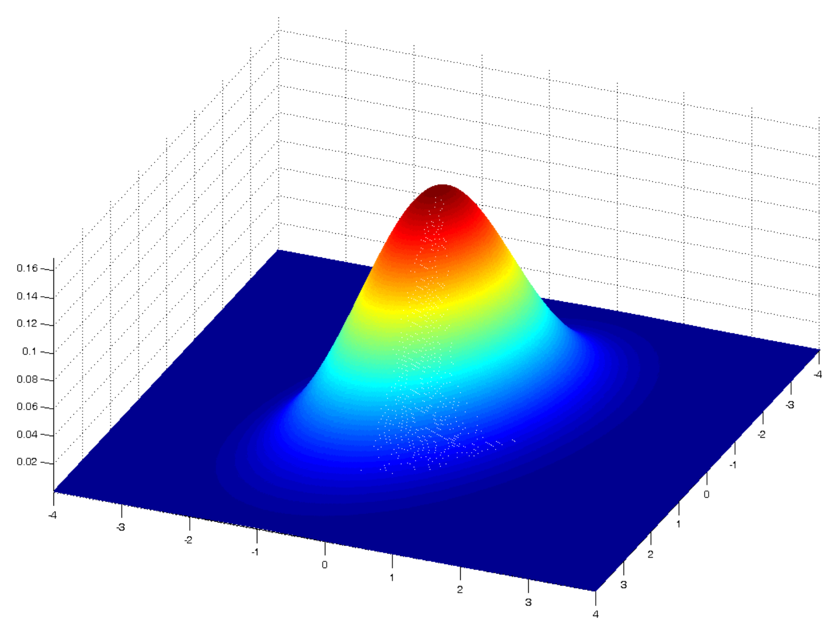


e.g., we might assume that the distribution of data given the class is Gaussian

Multi-dimensional Gaussian

- The Gaussian probability density is given by

$$p(\mathbf{x}|\mathbf{m}, \mathbf{\Sigma}) = \frac{1}{|2\pi\mathbf{\Sigma}|^{\frac{1}{2}}} \exp -\frac{1}{2}(\mathbf{x} - \mathbf{m})^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{m})$$



- To estimate from data (x)

$$\hat{\mathbf{m}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i$$

$$\hat{\mathbf{\Sigma}} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i - \hat{\mathbf{m}})(\mathbf{x}_i - \hat{\mathbf{m}})^T$$

- These estimates maximise the probability of the data x given parameters m, Σ

2-Class Gaussian Classifier

- Simple classification rule: choose class #1 if

$$p(\mathbf{x}|c_1) > p(\mathbf{x}|c_2)$$

- taking -2 x ln of both sides (reverses sign)

$$-2\ln p(\mathbf{x}|c_1) < -2\ln p(\mathbf{x}|c_2)$$

- negative log of Gaussian density

$$-2\ln p(\mathbf{x}) = -2\ln \frac{1}{|2\pi\mathbf{\Sigma}|^{\frac{1}{2}}} \exp -\frac{1}{2}(\mathbf{x}-\mathbf{m})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{m})$$

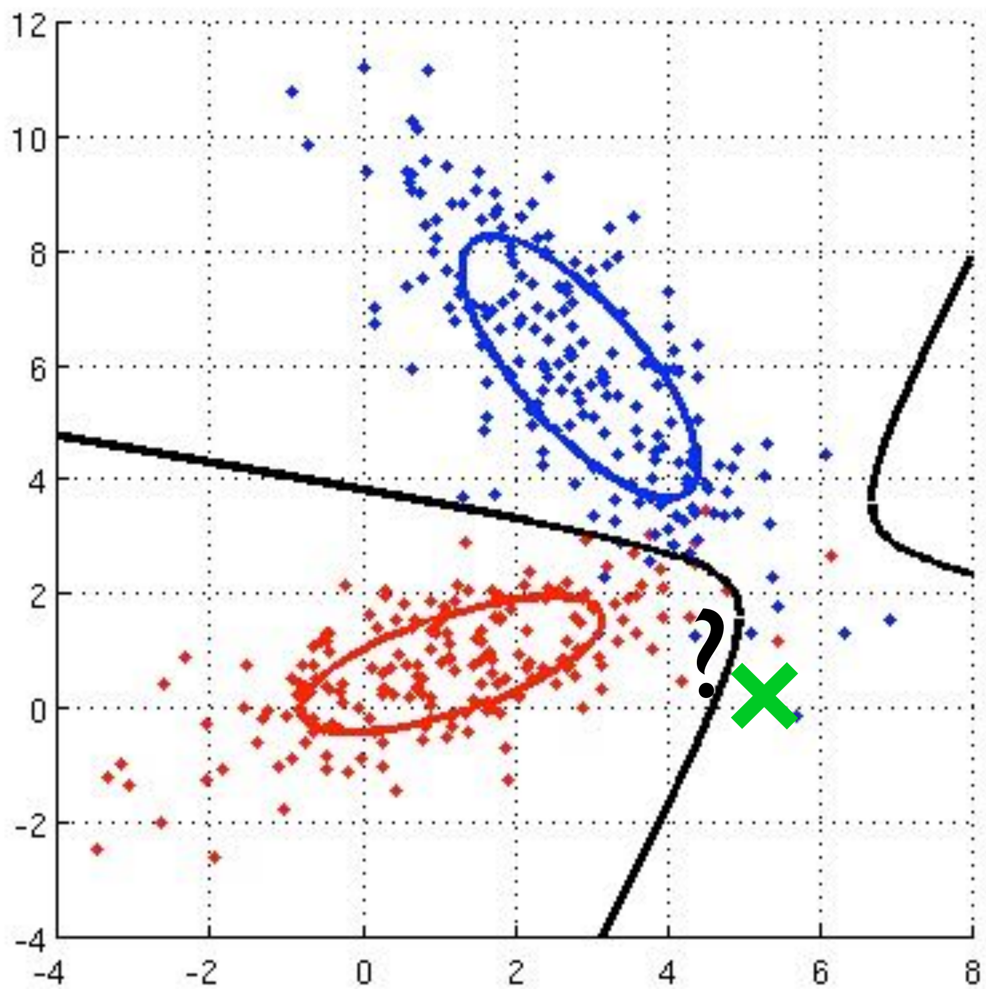
$$= \ln(2\pi^d) + \ln|\mathbf{\Sigma}| + (\mathbf{x}-\mathbf{m}^T)\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{m})$$

- decision rule becomes (class #1 if...)

$$\ln\mathbf{\Sigma}_1 + (\mathbf{x}-\mathbf{m}_1)^T\mathbf{\Sigma}_1^{-1}(\mathbf{x}-\mathbf{m}_1) < \ln\mathbf{\Sigma}_2 + (\mathbf{x}-\mathbf{m}_2)^T\mathbf{\Sigma}_2^{-1}(\mathbf{x}-\mathbf{m}_2)$$

2-Class Gaussian Classifier

- Suppose we've modelled our 2 classes with Gaussian distributions



$$p(\mathbf{x}|c_1) = N(\mathbf{x}; \mathbf{m}_1, \mathbf{\Sigma}_1)$$
$$p(\mathbf{x}|c_2) = N(\mathbf{x}; \mathbf{m}_2, \mathbf{\Sigma}_2)$$

- Our decision rule, class #1 if

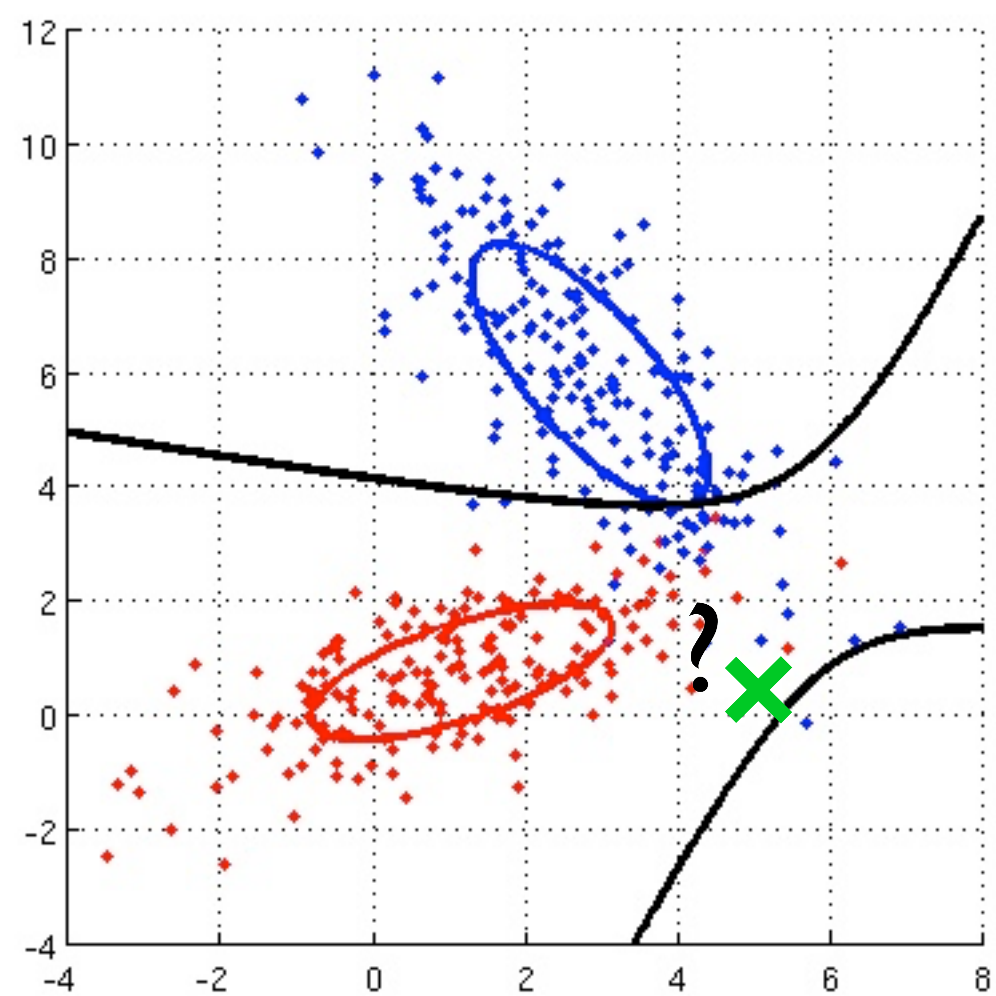
$$p(\mathbf{x}|c_1) > p(\mathbf{x}|c_2)$$

is called a maximum likelihood classifier

Incorporating Prior Knowledge

- What if red is more common than blue?
- Weight each likelihood by prior probabilities $p(c_1), p(c_2)$
- Decision rule (MAP classifier) choose class #1 if:

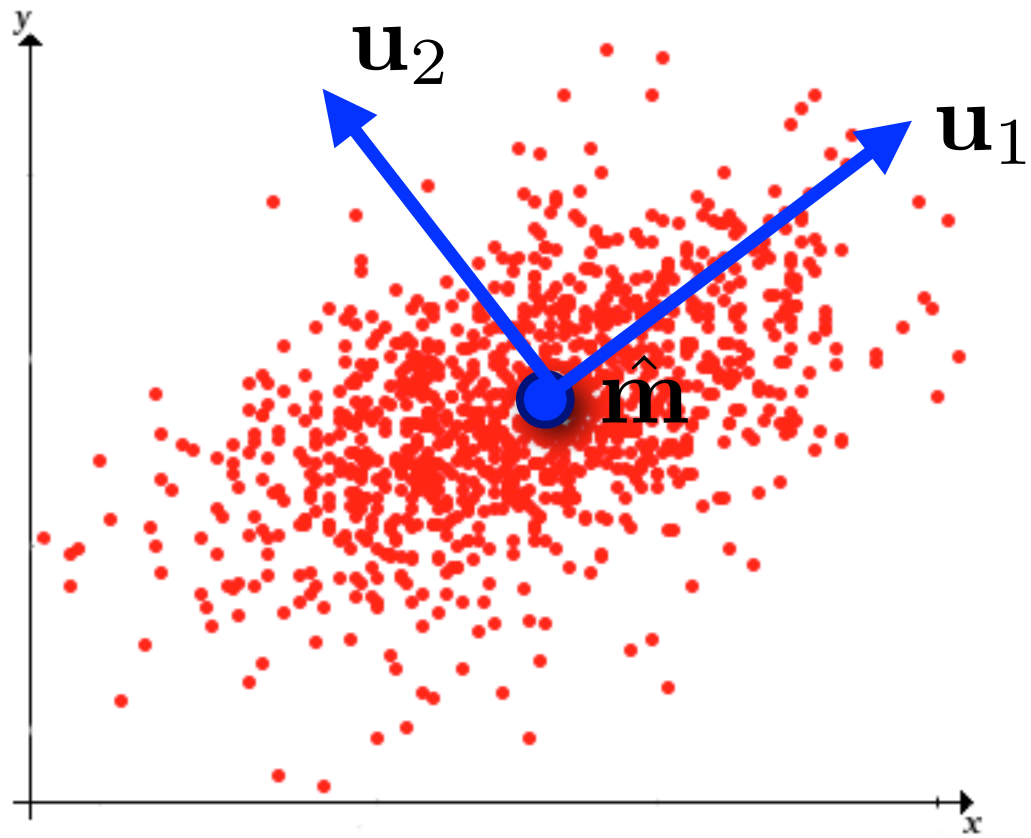
$$p(\mathbf{x}|c_1)p(c_1) > p(\mathbf{x}|c_2)p(c_2)$$



$p(c_1) = 0.59$

$p(c_2) = 0.45$

Principal Components



$$\hat{\mathbf{m}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

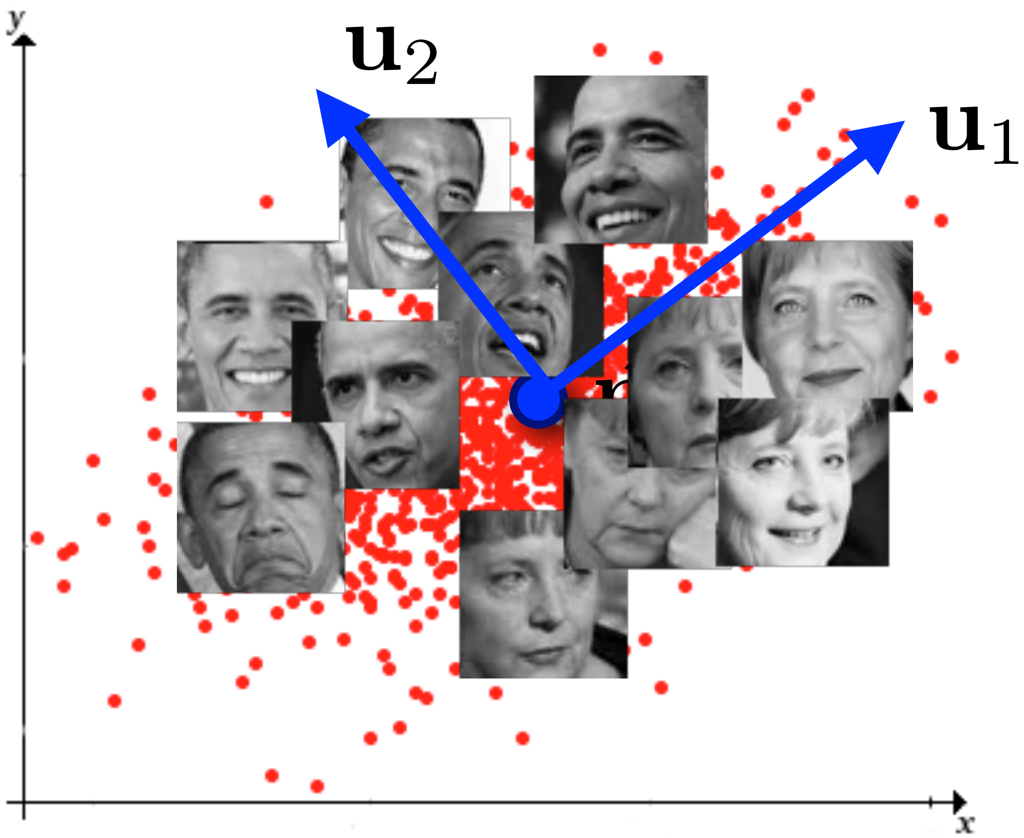
$$\hat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \hat{\mathbf{m}})(\mathbf{x}_i - \hat{\mathbf{m}})^T$$

- We can visualise the major modes of variation in data by looking at the eigenvectors of the covariance matrix

$$\hat{\mathbf{\Sigma}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$$

- The eigenvectors $\mathbf{U} = [\mathbf{u}_1 \mathbf{u}_2 \ldots]$ are directions of max variance, they are mutually orthogonal

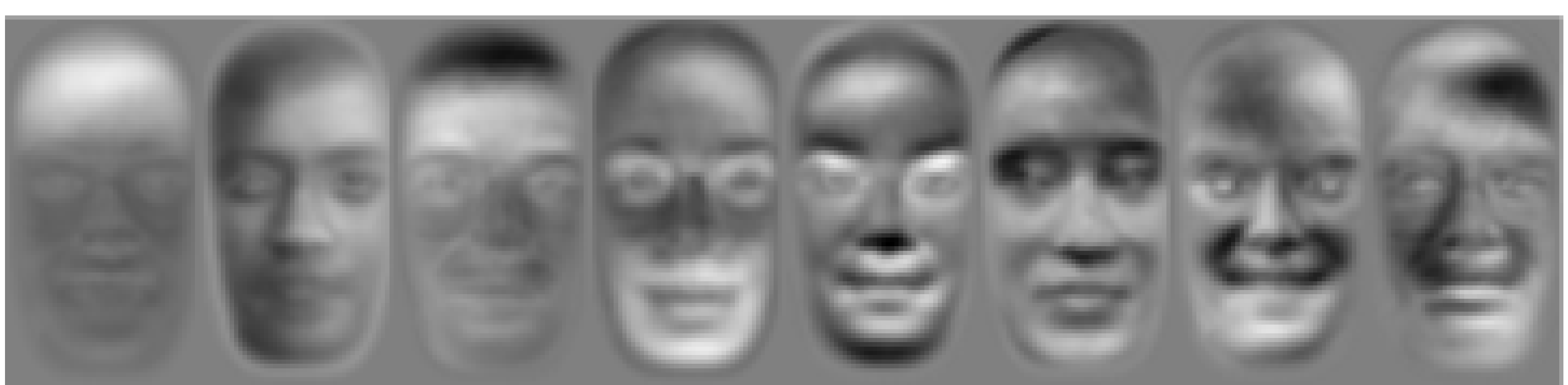
Principal Components



$$\hat{\mathbf{m}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

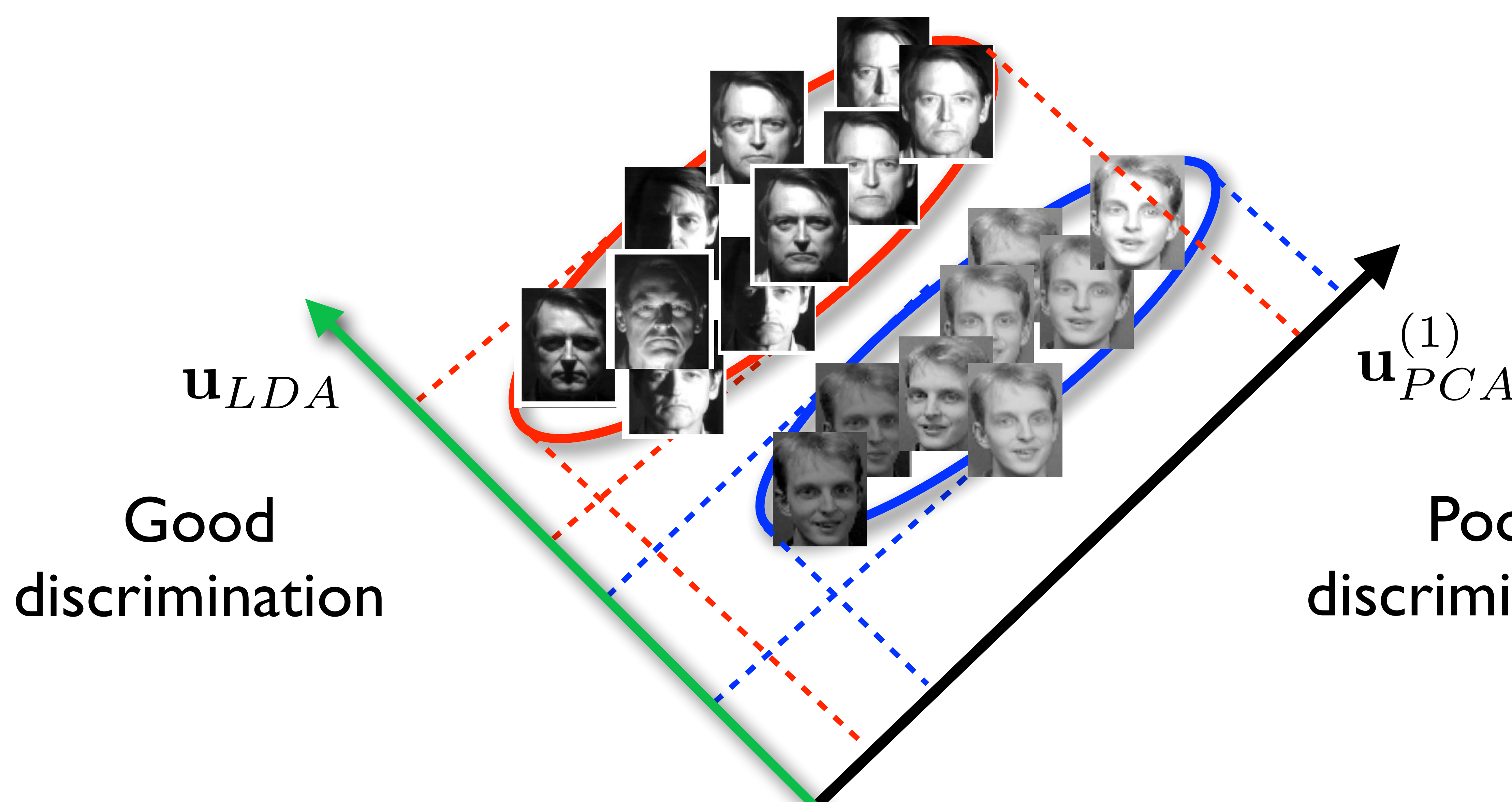
$$\hat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \hat{\mathbf{m}})(\mathbf{x}_i - \hat{\mathbf{m}})^T$$

- e.g., the principal components (covariance eigenvectors) of a set of faces can be visualised as images [Moghaddam et al 2000]

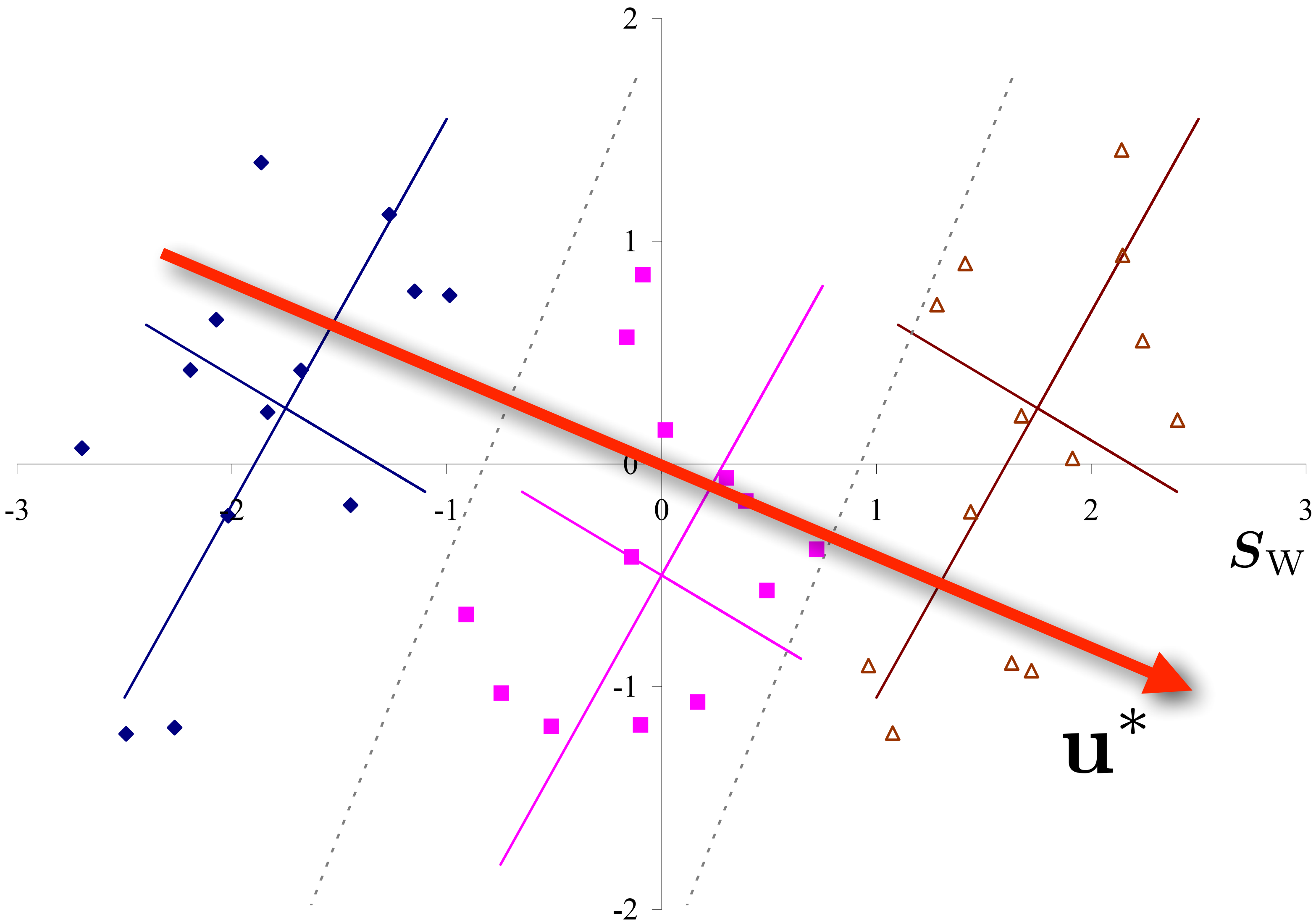


Discriminative Projection

- PCA directions are not generally discriminative
- Intuitively, we'd like to project to a direction that separates the classes without too much overlap



Fisher's Linear Discriminant



$$\mathbf{u}^* = \arg\max_{\mathbf{u}} \; J(\mathbf{u}) = \frac{\mathbf{u}^T \mathbf{S}_B \mathbf{u}}{\mathbf{u}^T \mathbf{S}_W \mathbf{u}}$$

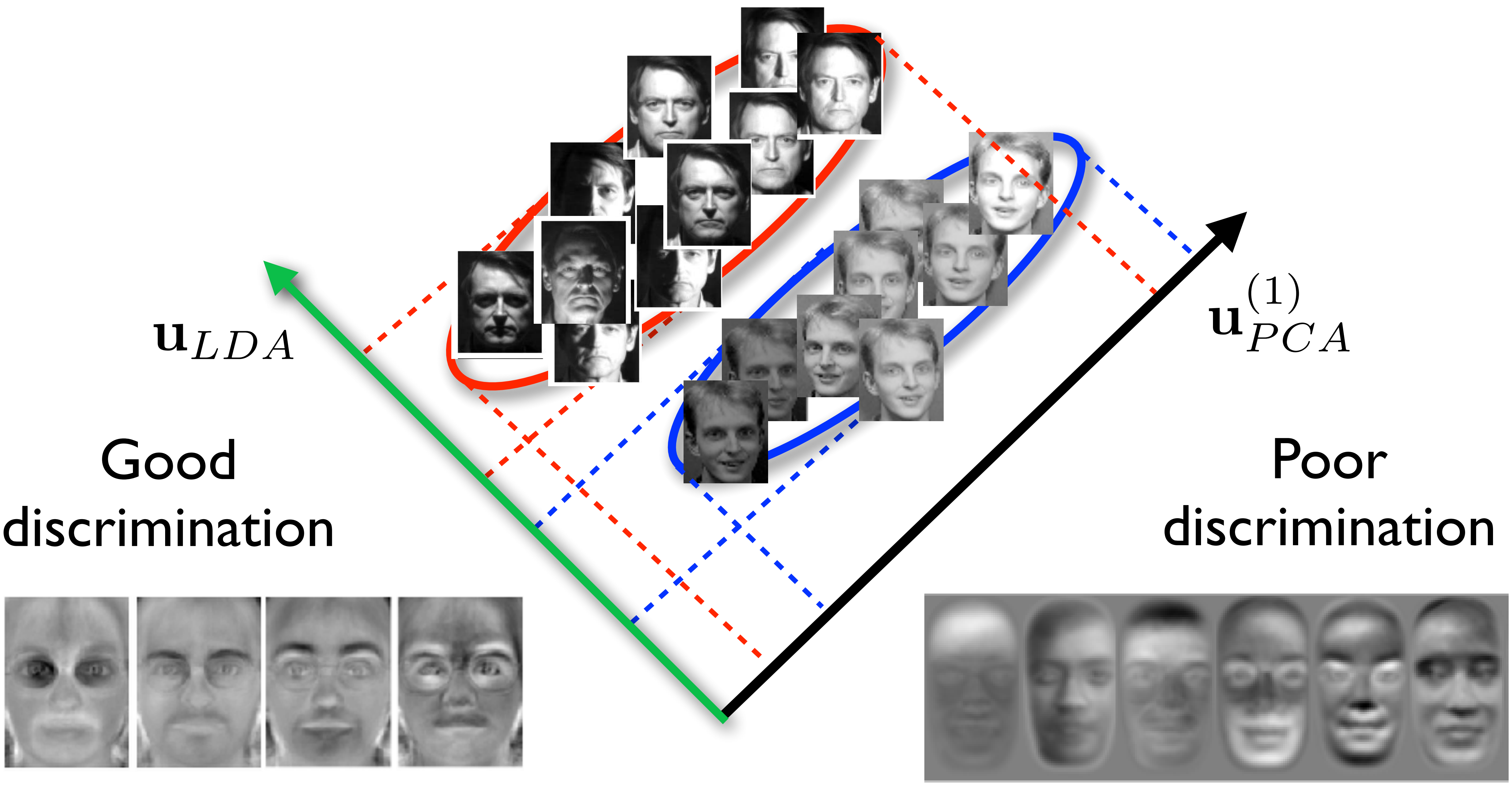
$$\boldsymbol{S}_{\mathrm{W}} = \sum_{k=0}^{K-1} \boldsymbol{S}_k = \sum_{k=0}^{K-1} \sum_{i \in C_k} (\boldsymbol{x}_i - \boldsymbol{m}_k)(\boldsymbol{x}_i - \boldsymbol{m}_k)^T,$$

$$\boldsymbol{S}_{\mathrm{B}} = \sum_{k=0}^{K-1} N_k (\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^T,$$

- Maximise the ratio of between class variance to within class variance, in the projected direction u
- Can be generalised to multi-dimensions, e.g., $J(\mathbf{U}) = \dfrac{|\mathbf{U}^T \mathbf{S}_B \mathbf{U}|}{|\mathbf{U}^T \mathbf{S}_W \mathbf{U}|}$
- An example of Linear Discriminant Analysis (LDA)

PCA vs LDA

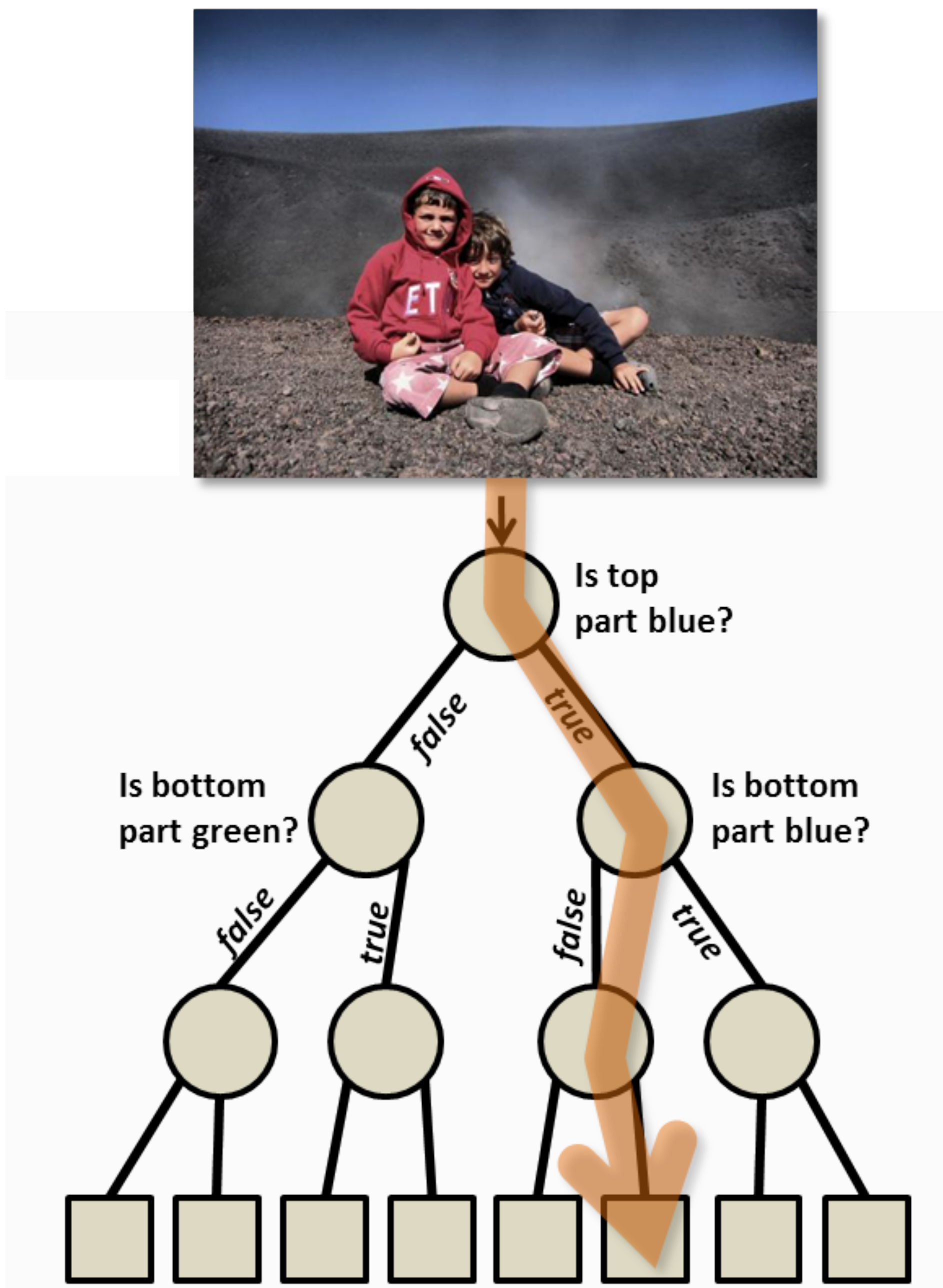
- PCA : maximise projected variance
- LDA : maximise between class, minimise within class variance



[A. Martinez]

Decision Forests

- A decision tree organises a hierarchical set of feature splits



Nodes in the tree split the data based on parametrized, typically simple features (weak learners):

$$h(\theta, \tau) = [\tau_1 < \theta^T[\mathbf{x}, 1] < \tau_2]$$

$h(\theta, \tau)$ = binary split function

$\mathbf{x}$ = input data

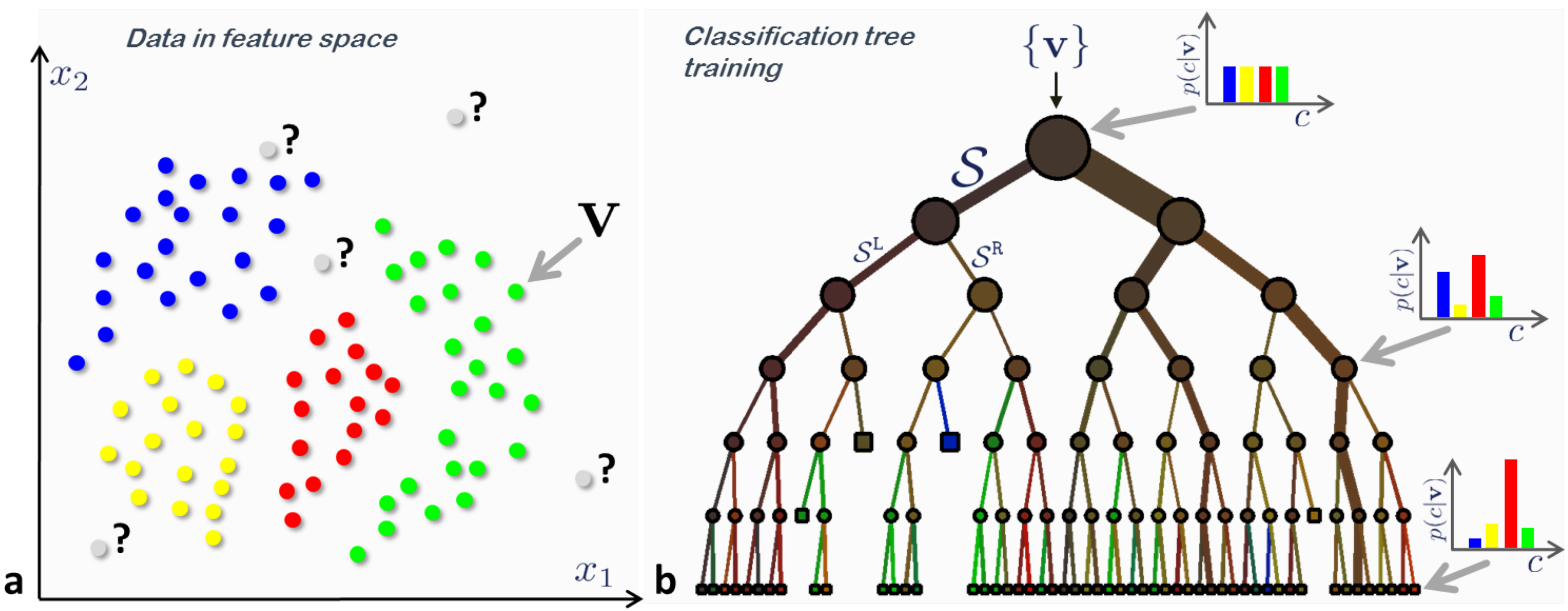
θ, τ = trainable parameters

[Criminisi et al 2011]

Classification Tree Training

- To train a tree for classification, parameters for the split nodes are optimised based on an information gain criterion, e.g.,

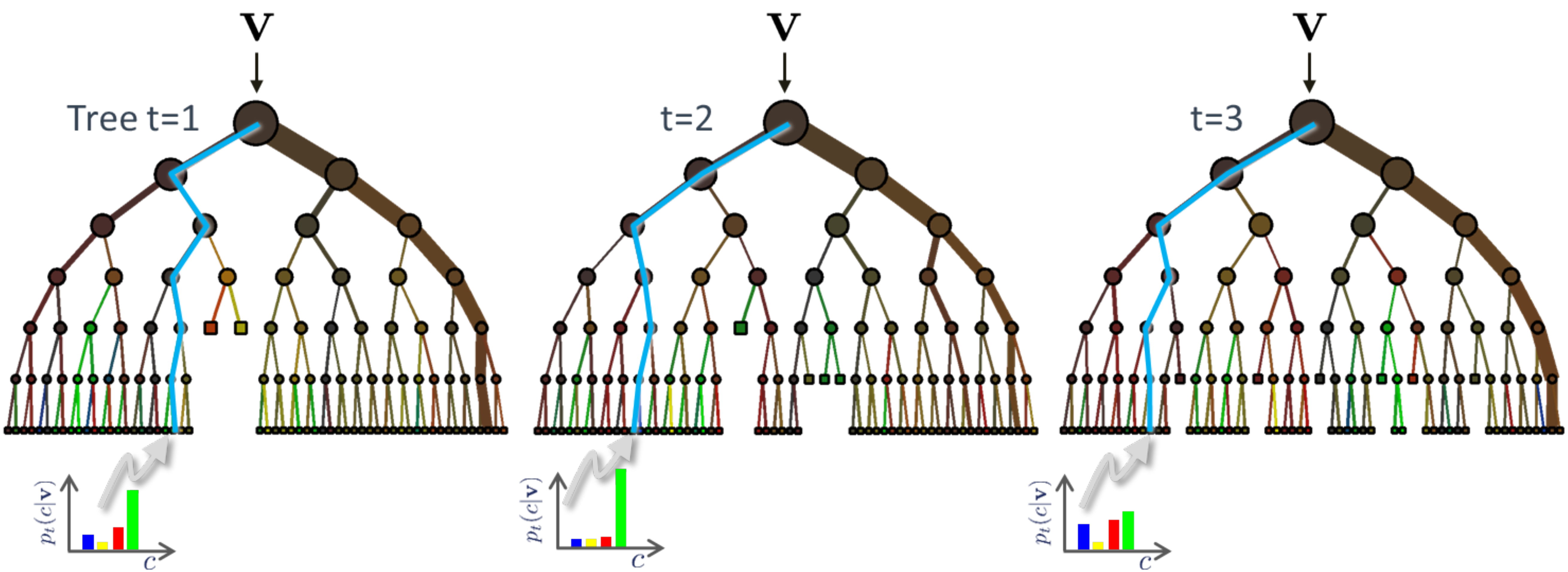
$$\boldsymbol{\theta}_j^* = \arg\max_{\boldsymbol{\theta}_j \in \mathcal{T}_j} I_j = H(\mathcal{S}_j) - \sum_{i \in \{L,R\}} \frac{|\mathcal{S}_j^i|}{|\mathcal{S}_j|} H(\mathcal{S}_j^i) \qquad H(\mathcal{S}) = -\sum_{c \in \mathcal{C}} p(c) \log p(c)$$



Leaves store a probability distribution over class c

Classification Forest

- A set of trees (forest) is trained with different random features
- At test time the query v is put through all trees and the class probability distributions at the leaves are averaged:



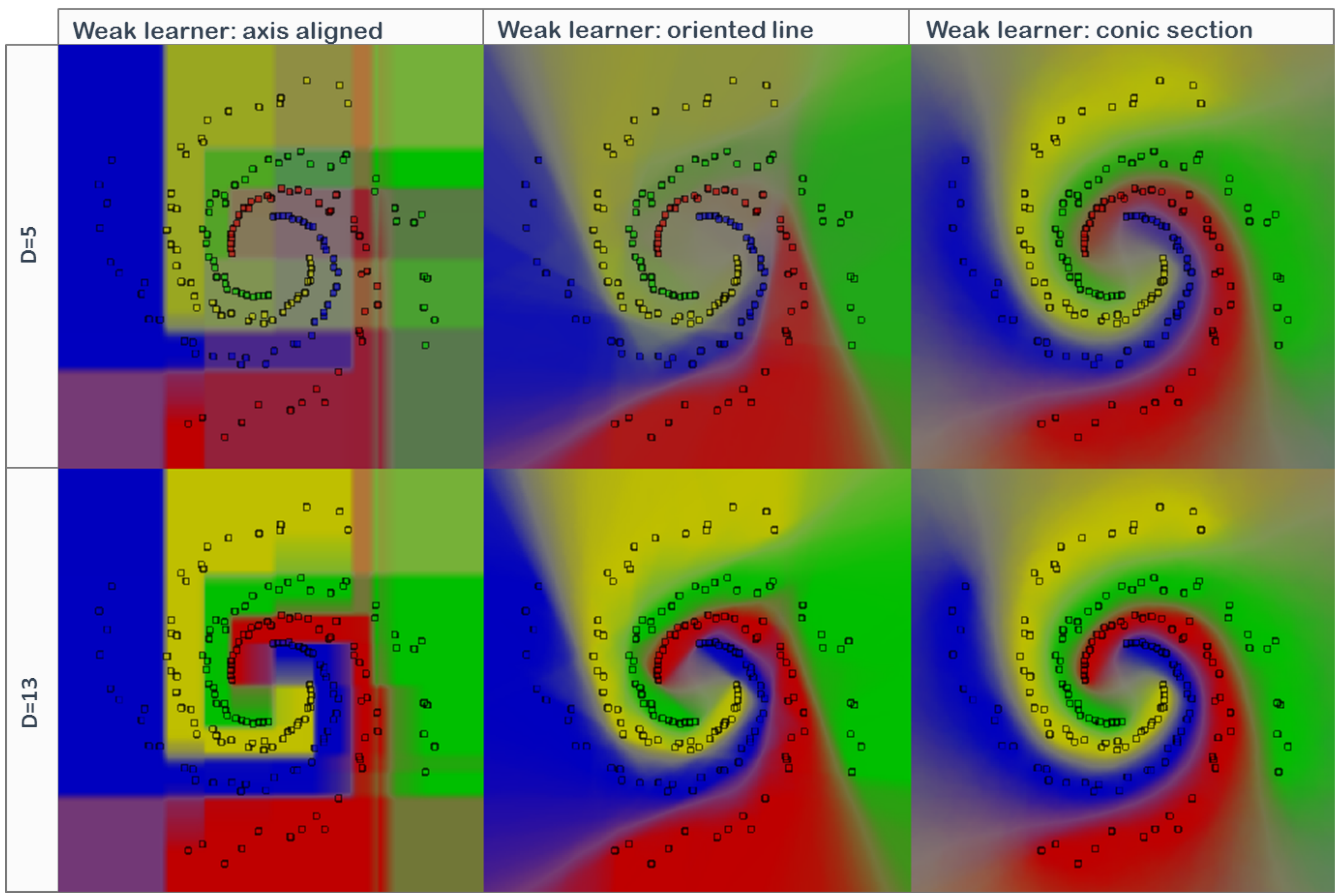
$$p(c|\mathbf{v}) = \frac{1}{T}\sum_{t}^{T} p_t(c|\mathbf{v})$$

Classification Forests

- By ensembling a large collection of weak features we can model complex decision boundaries, e.g., 400 trees

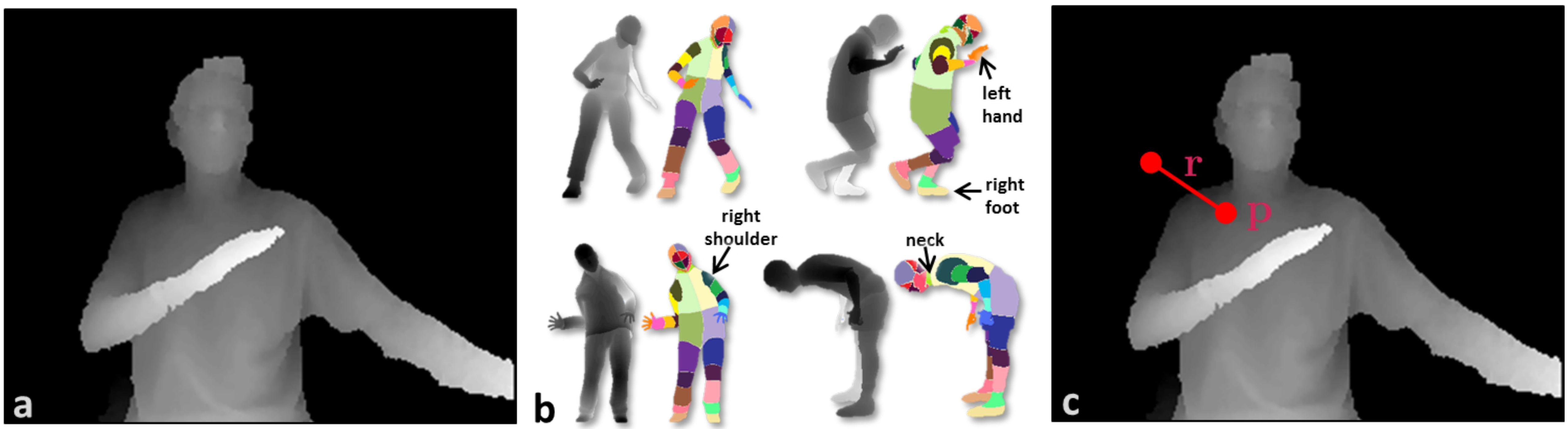
depth = 5

depth = 13



Application: Body Pose Estimation

- Classification Forests have been used for body pose estimation using the Kinect depth scanner

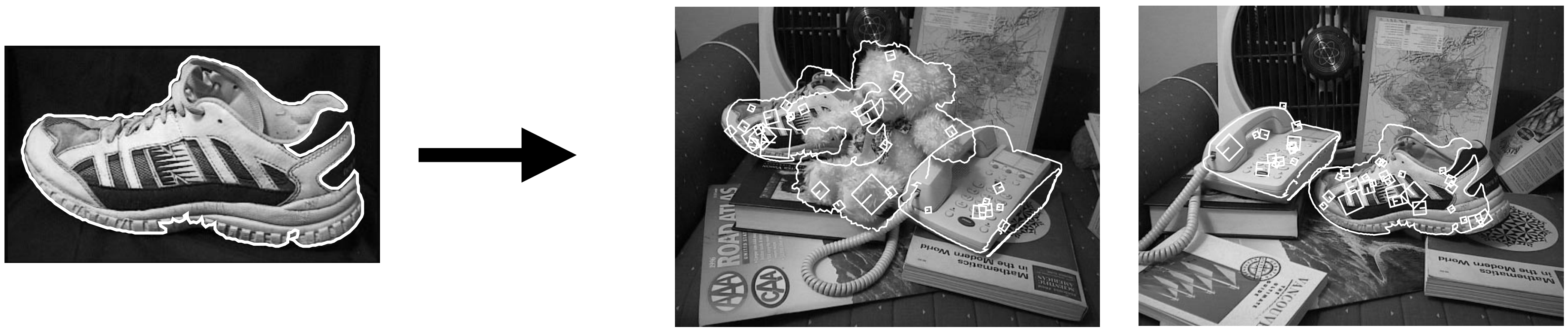


- Features (weak learners) are simple depth differences, parametrized by an offset and threshold $\boldsymbol{\theta}_j = (\mathbf{r}_j, \tau_j)$
- The model was trained using a large dataset of CG generated human poses
- At test time, every pixel is classified into 1 of 31 body parts

[Shotton et al 2011]

Recognition using Local Features

- Feature-based object instance recognition is similar to image registration (2D) or camera pose estimation (3D):

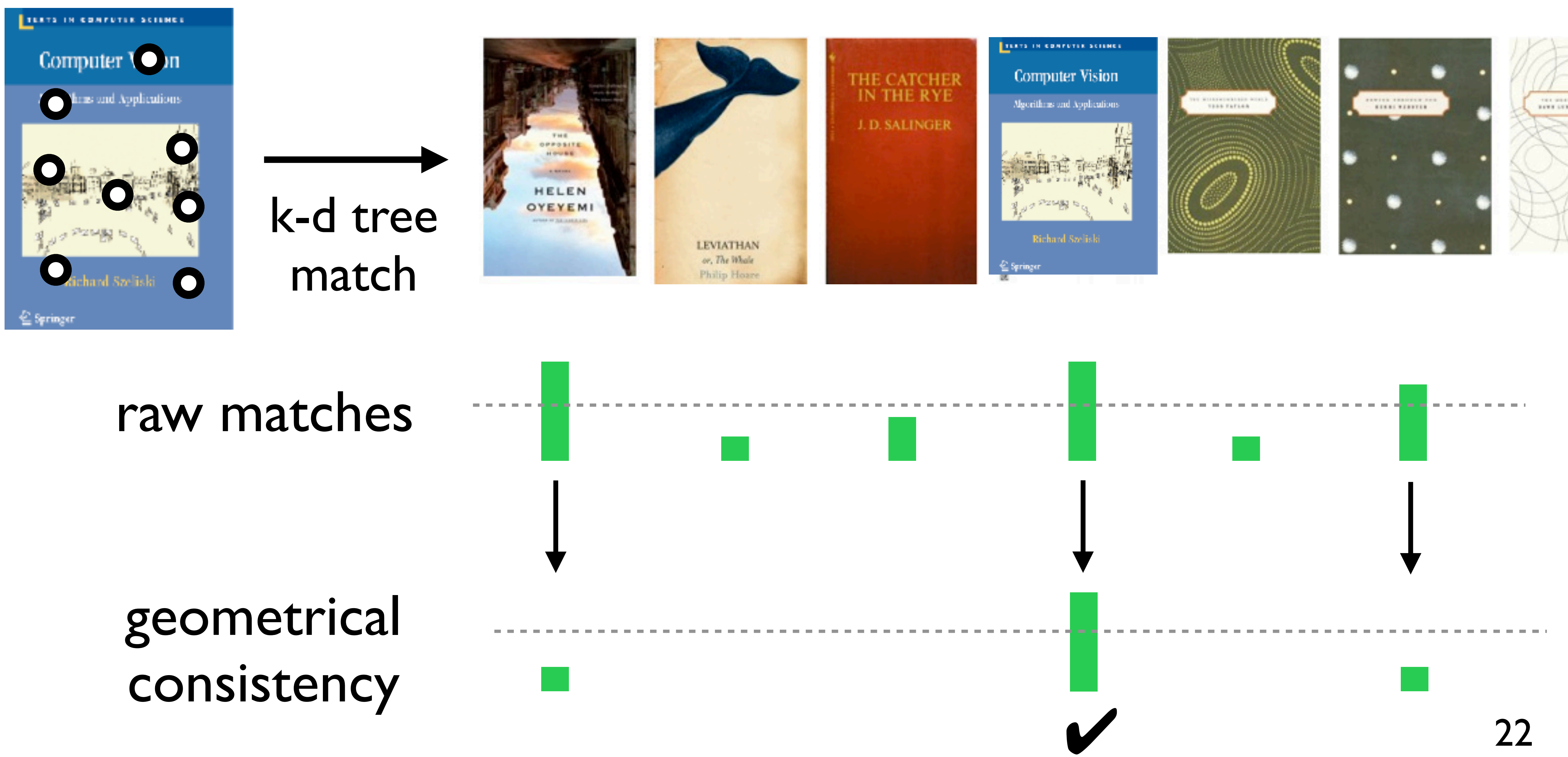


1. Detect Local Features (e.g., SIFT) in all images
2. Match Features using Nearest Neighbours
3. Find geometrically consistent matches using RANSAC (with Affine/Homography or Fundamental matrix)

- The final stage is to verify the match, e.g., require that # consistent matches > threshold

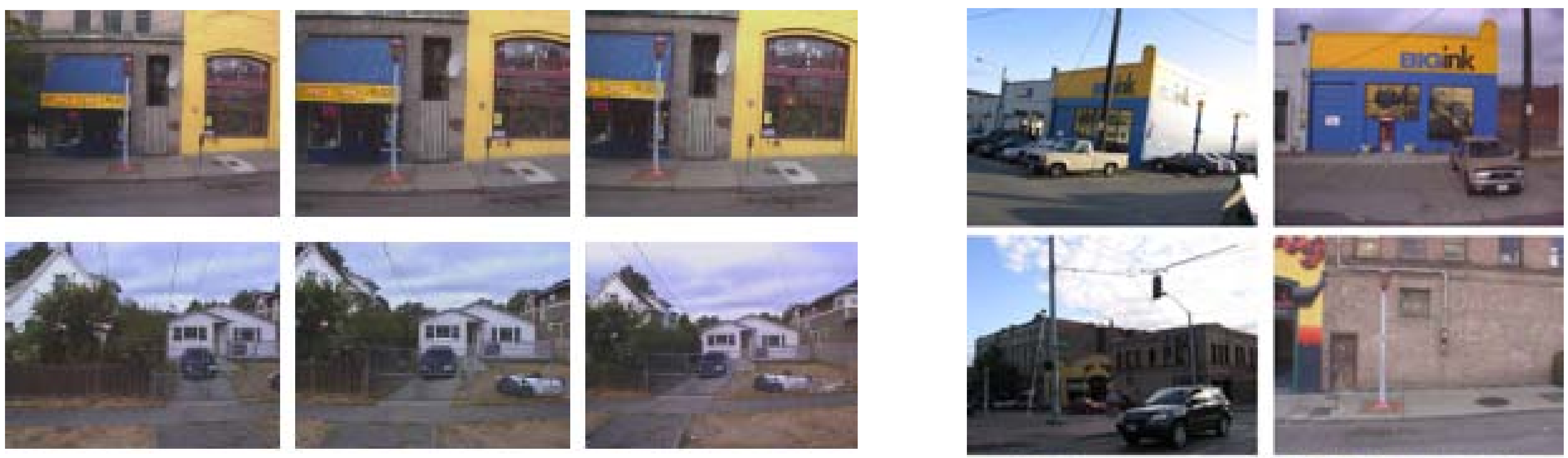
Scaling Local Feature Recognition

- To avoid performing all pairwise comparisons $O(n^2)$:
- Match query descriptors to entire database using k-d tree
- Select subset with max # raw matches and check geometry

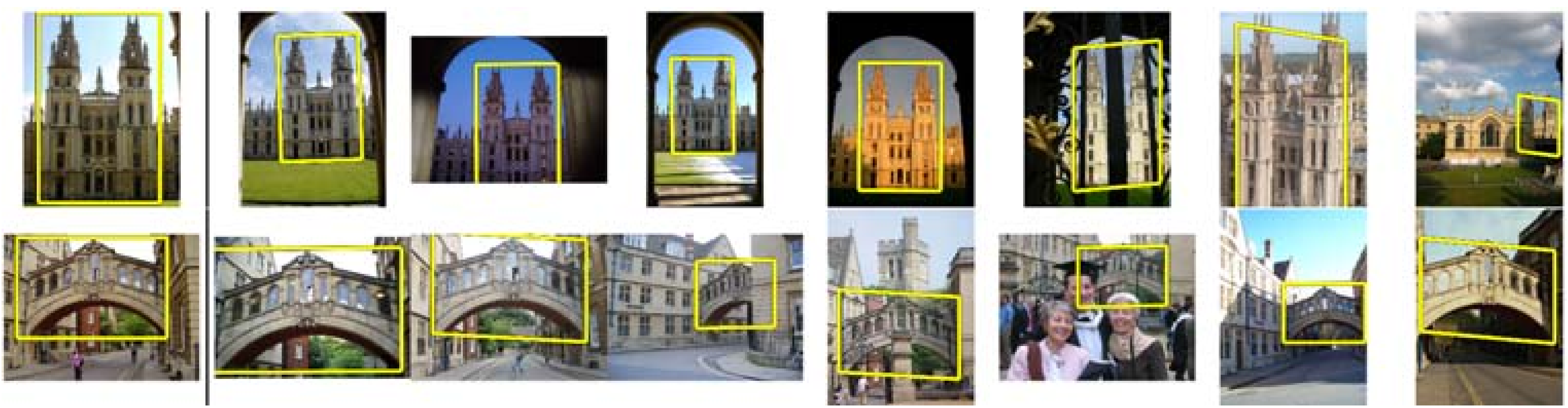


Application: Location Recognition

- Find photo in streetside imagery



[Schindler Brown Szeliski 2007]



[Philbin et al 2007]

Local Feature Recognition Failures

- Features + RANSAC fails with large appearance variation, e.g., most object categories and some instance problems

Few correct matches

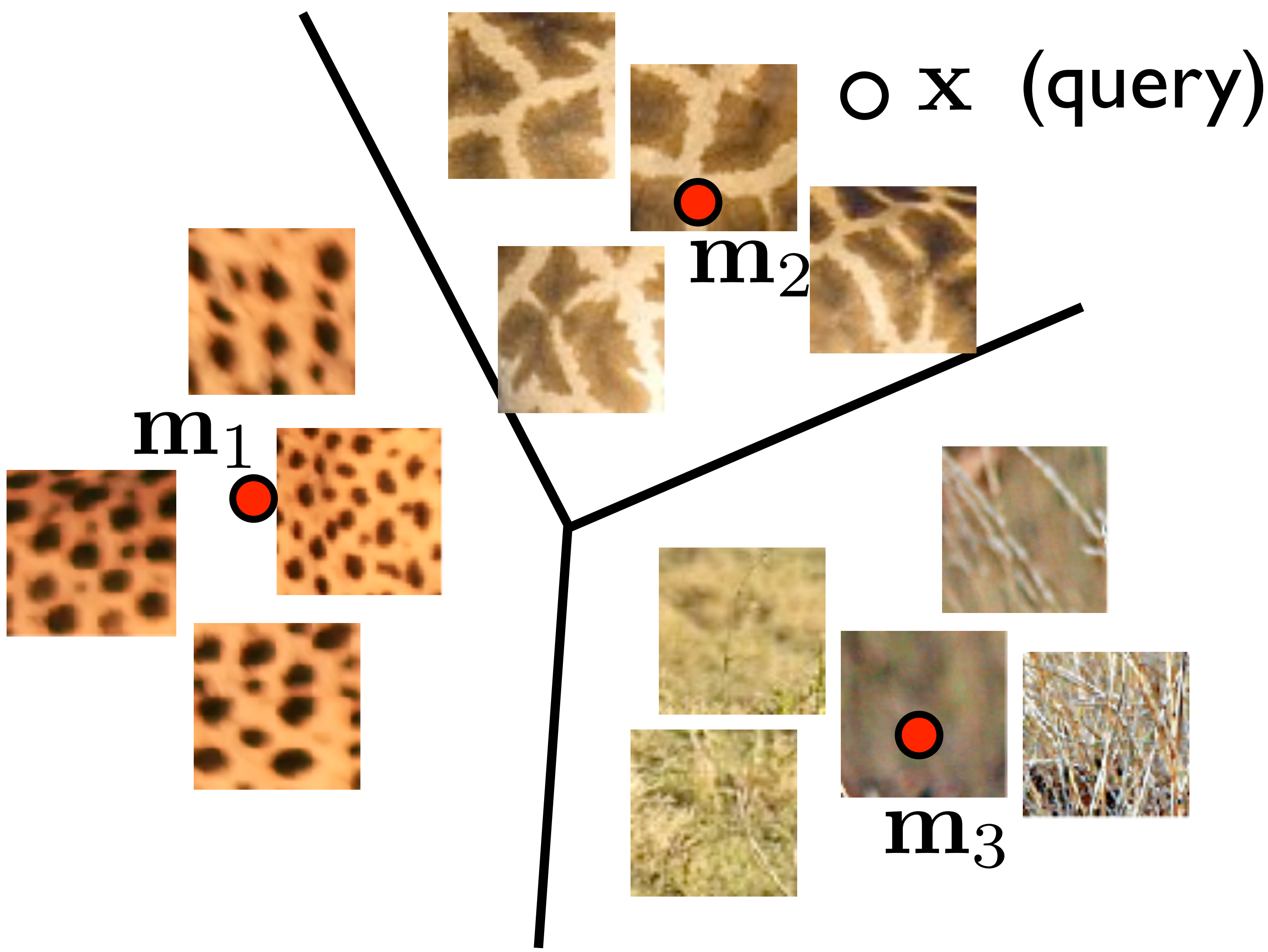
Local Feature Recognition Failures

- Features + RANSAC fails with large appearance variation, e.g., most object categories and some instance problems

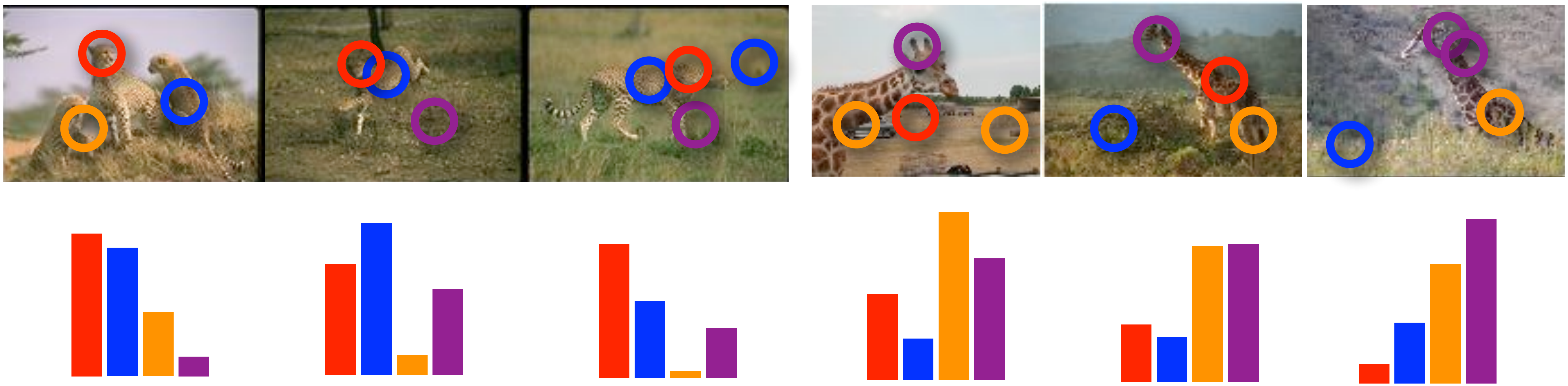
No correct matches

Visual Words

- The amorphous appearance of visual categories can be modelled using regions of feature space
- A common method is to quantise feature descriptors to a codebook of "visual words" using k-means clustering



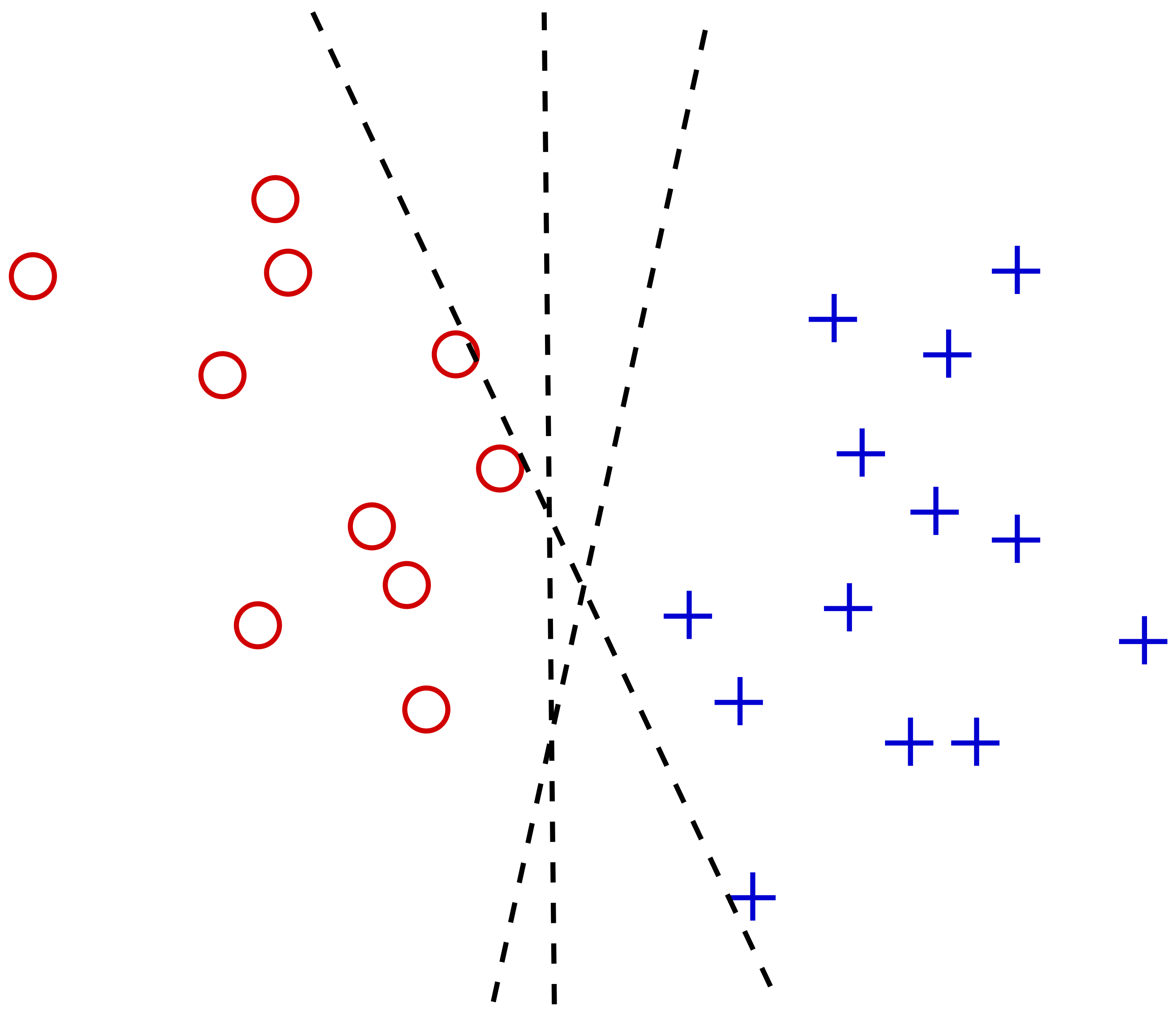
Visual Word Histogram + SVM



- A popular category recognition method was to use histograms of visual word frequencies to represent each image
- Given a labelled image dataset, a Support Vector Machine (SVM) could be trained to perform image classification, with per-image visual word histograms as input
- Variants on this theme were state-of-the-art for image classification up to around 2011 (deep learning + AlexNet)

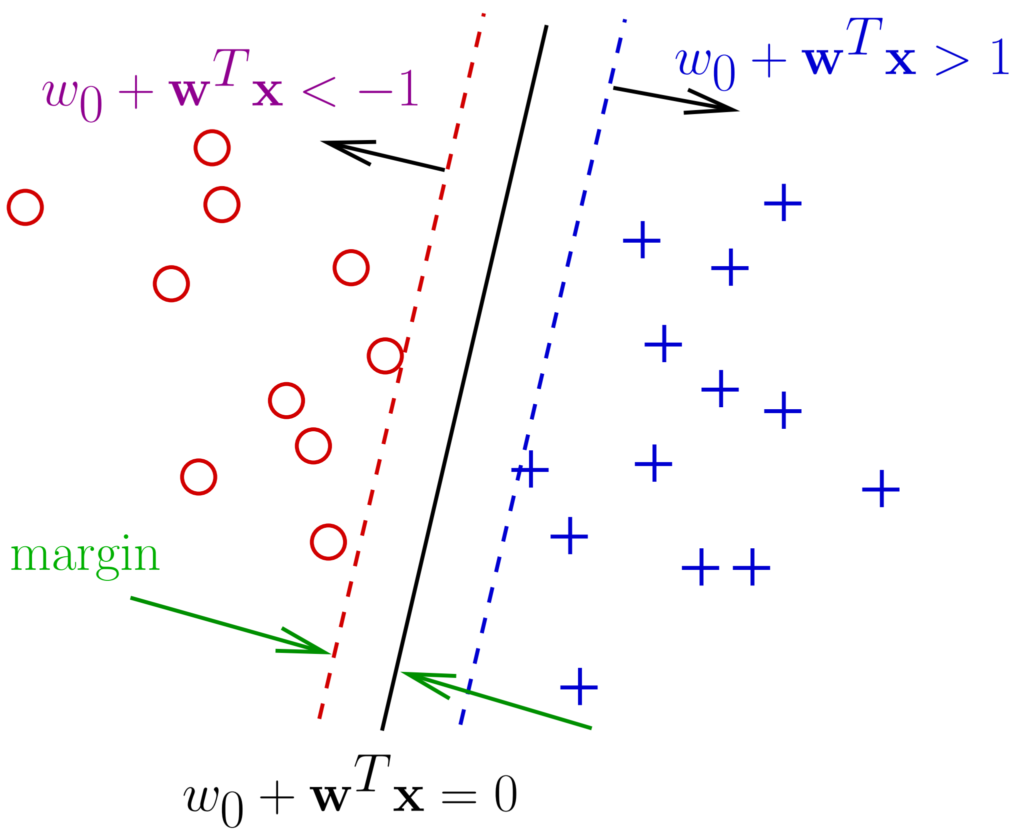
Support Vector Machines

- Which decision boundary is best?



Max-Margin Classifier

- Separation between classes is called the **margin**



- Distance from boundary

$$d_i = y_i(\mathbf{w}^T\mathbf{x}_i + w_0)$$

Note that d_i could be arbitrarily large for large w

- Maximise the minimum distance for fixed |w|

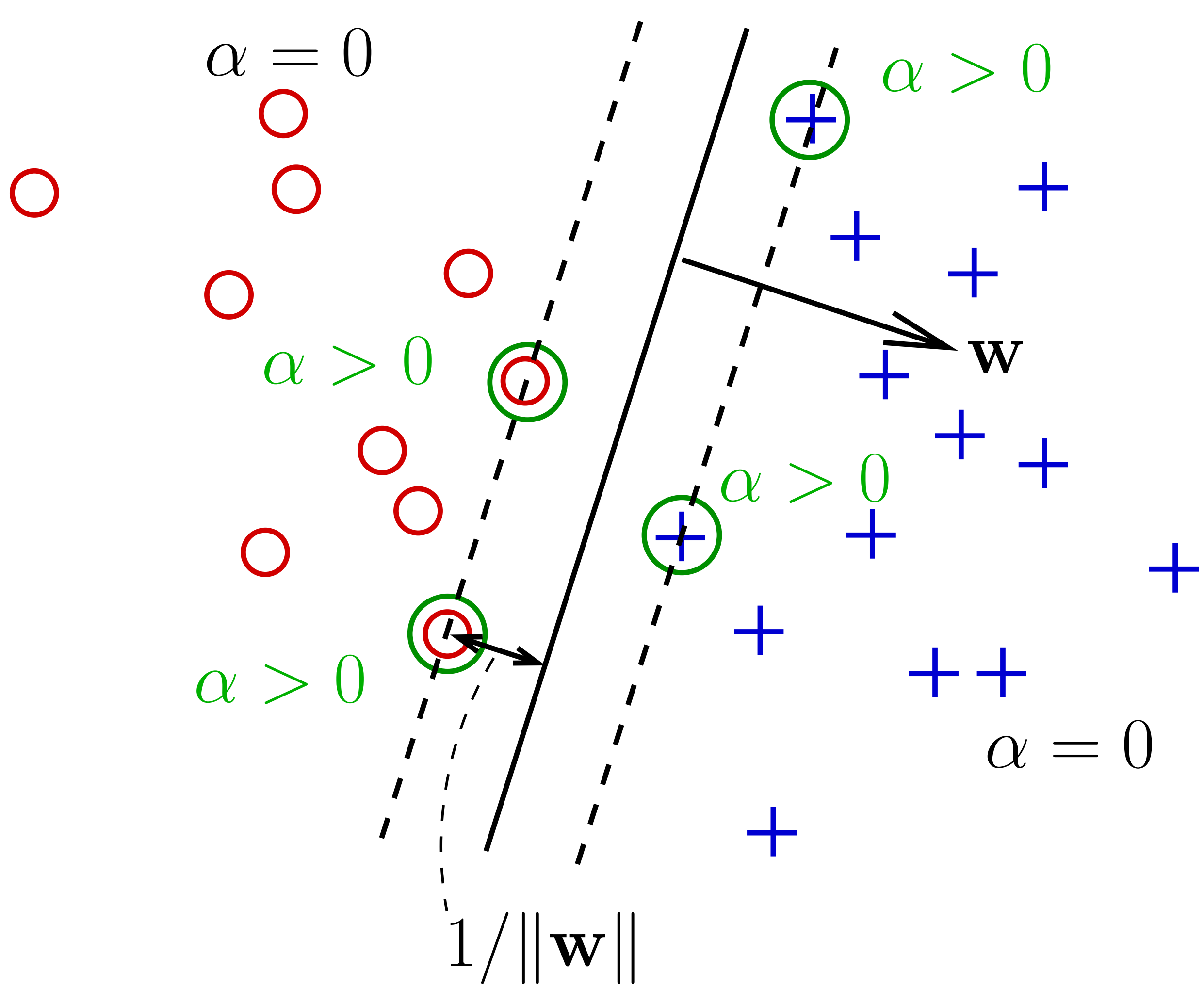
$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \min_i y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \quad s.t. \ |\mathbf{w}|^2 = 1$$

(quadaratic programming)

[Figure credits: G. Shakhnarovich]

Support Vectors

- The active constraints are due to the data that define the classification boundary, these are called **support vectors**



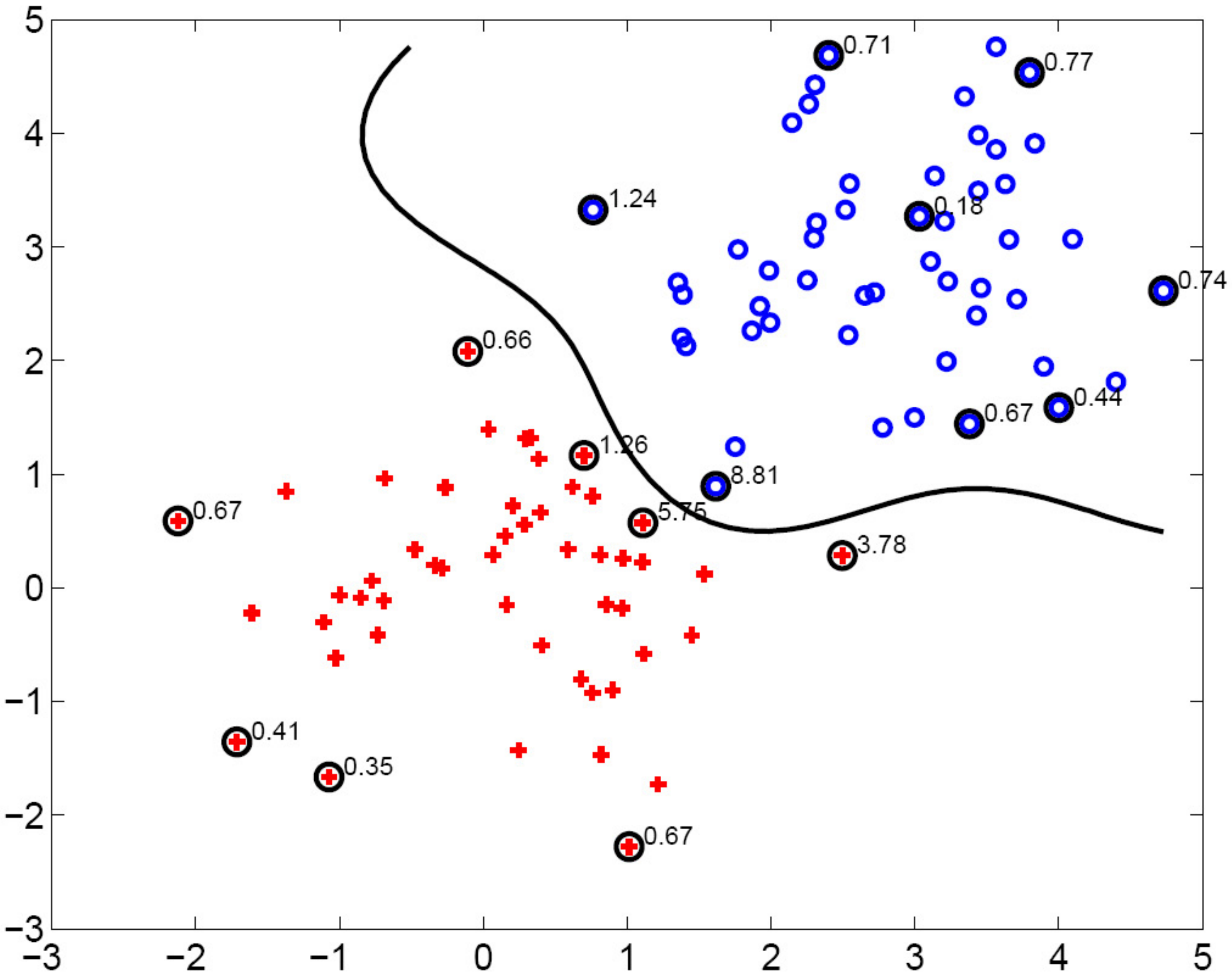
Final classifier can be written in terms of the support vectors:

$$\hat{y} = \operatorname{sign}\left(\hat{w}_0 + \sum_{\alpha_i > 0} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}\right)$$

Non-Linear SVM

- Replace inner product with kernel

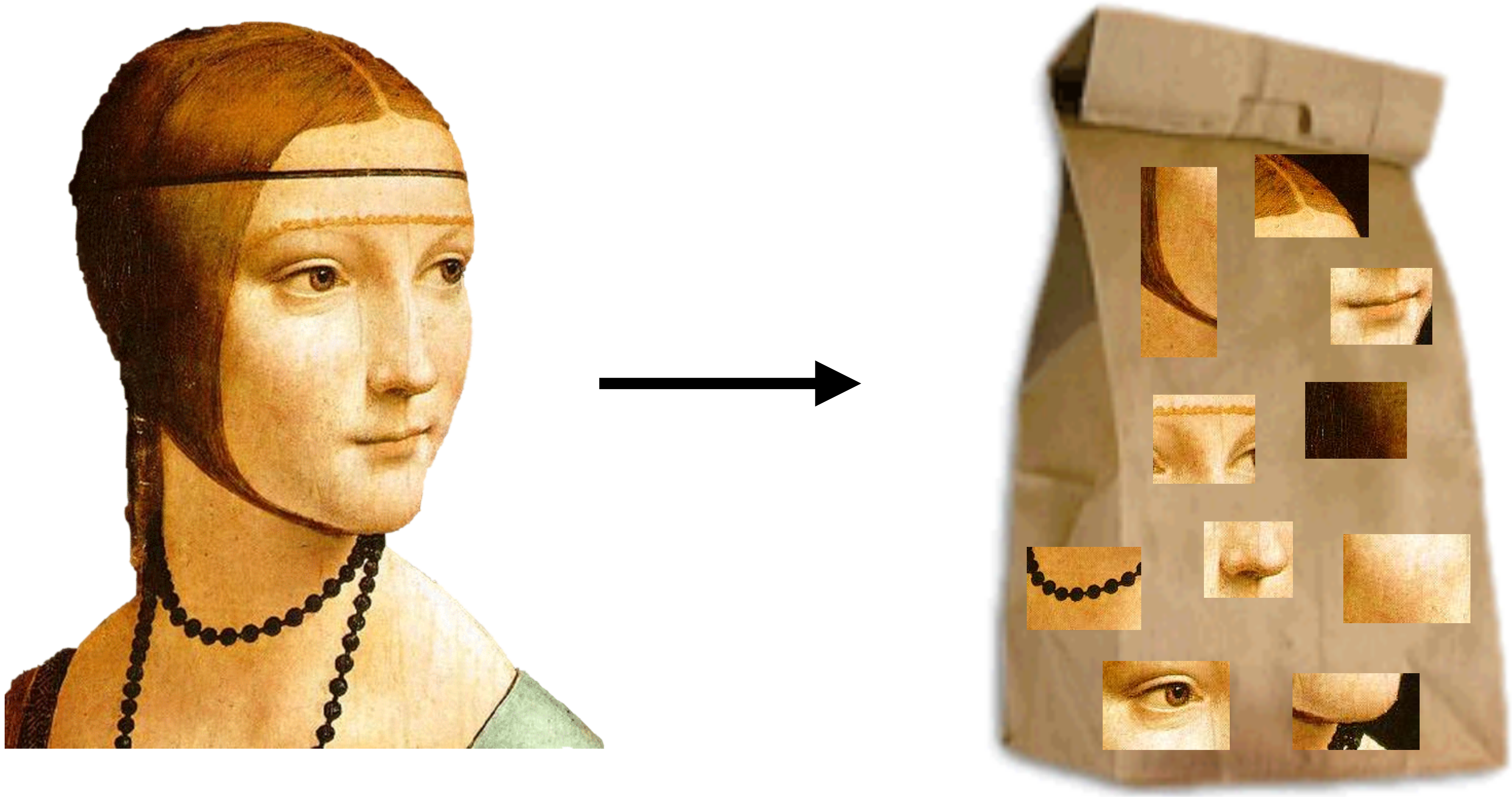
$$\mathbf{x}_i^T \mathbf{x} \rightarrow \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) \rightarrow k(\mathbf{x}_i, \mathbf{x})$$



- Data are (ideally) linearly separable in ϕ(x)
- But we don't need to know ϕ(x), we just specify k(x,y)
- Points with α>0 (circled) are support vectors
- Other data can be removed without affecting classifier

Bag of Words

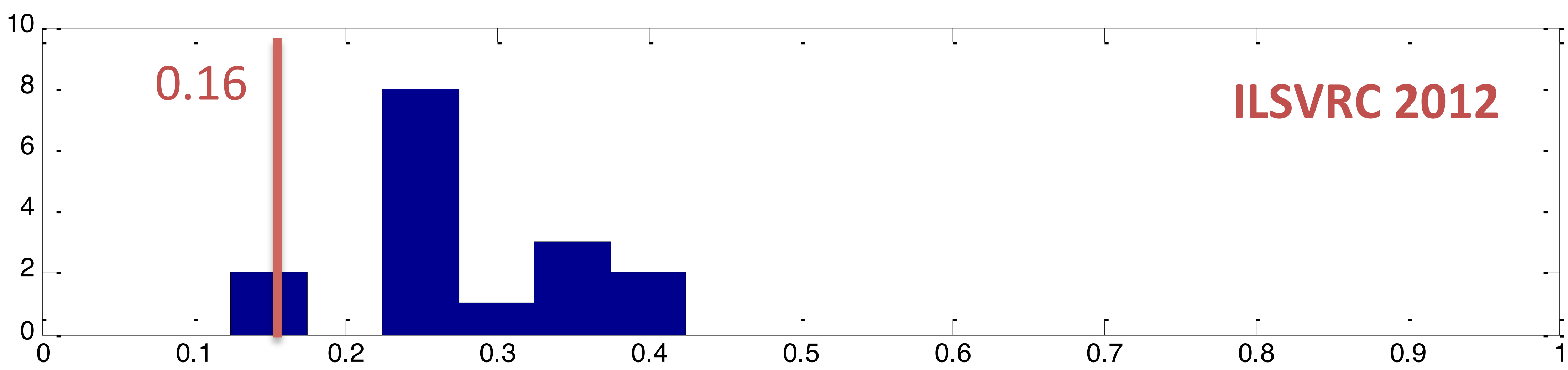
- Images are represented as collections of local visual words (discarding spatial relationships), before SVM classification



- There is some evidence that similar features are effective in CNNs, e.g., pooling of learned small receptive field (17x17) features gives good performance on ImageNet [“BagNets” Brendel Bethge 19]

ILSVRC 2012

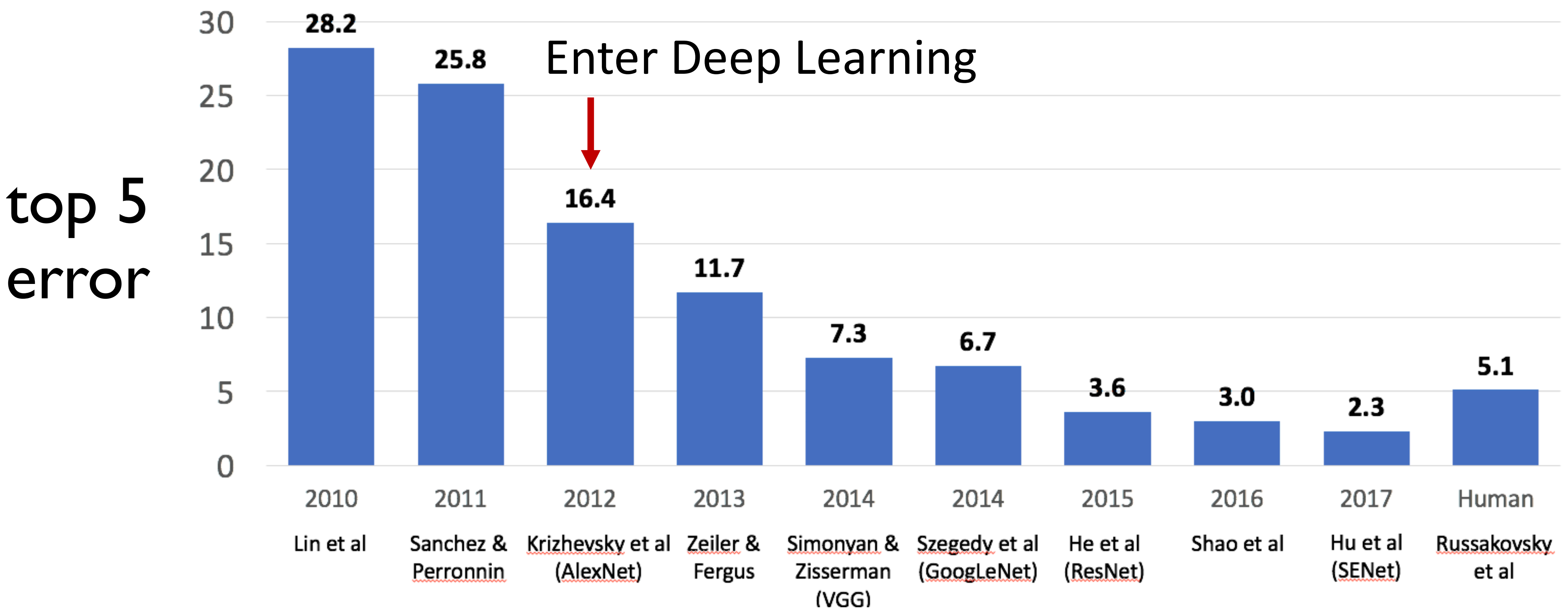
- For the ImageNet Large Scale Visual Recognition Challenge 2012 competitors had to classify 100K unseen test images into one of 1000 categories
- Alex Krizhevsky and Geoff Hinton used an 8-layer, 60M parameter convolutional neural network trained on two GPUs for 1 week
- This beat all other approaches (visual word/SVM based) by a large margin:



Top 5 error rate

Alexnet

- Won the Imagenet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 by a large margin
- Some ingredients: Deep neural net (Alexnet), Large dataset (Imagenet), Lots of compute (2 GPU weeks), non-saturating activation functions (ReLU)



[J. Johnson]

Next Lecture

- Neural Nets