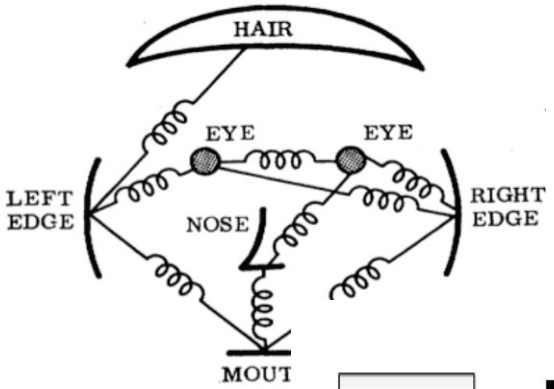
Lecture 8
Statistical Learning & Part Models

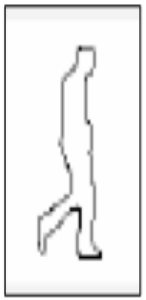
COS 429: Computer Vision

PRINCETON UNIVERSITY

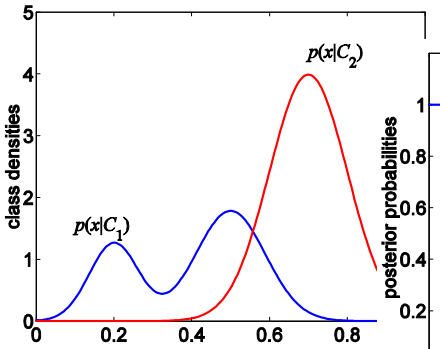
Review: Object Detection Typical Components

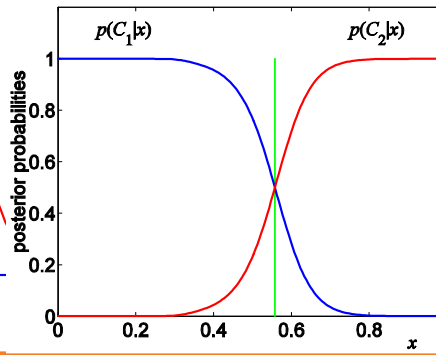
- **Hypothesis** generation
  - Sliding window, Segmentation, feature point detection, random, search
- **Encoding** of (local) image data
  - Colors, Edges, Corners, Histogram of Oriented Gradients, Wavelets, Convolution Filters
- **Relationship** of different parts to each other
  - Blur or histogram, Tree/Star, Pairwise/Covariance
- **Learning** from (labeled) examples
  - Selecting representative examples (templates), Clustering, Building a cascade
  - Classifiers: Bayes, Logistic regression, SVM, AdaBoost, Neural Network...
  - Generative vs. Discriminative
- **Verification -** removing redundant, overlaping, incompatible examples
    - Non-Max Suppression, context priors, geometry





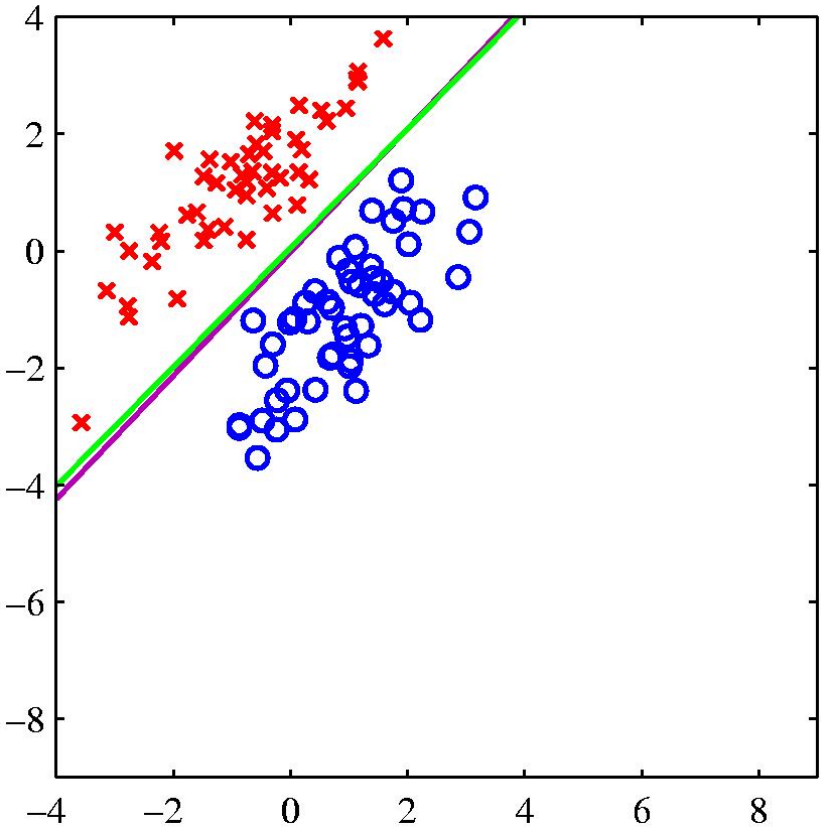
Exemplar Summary

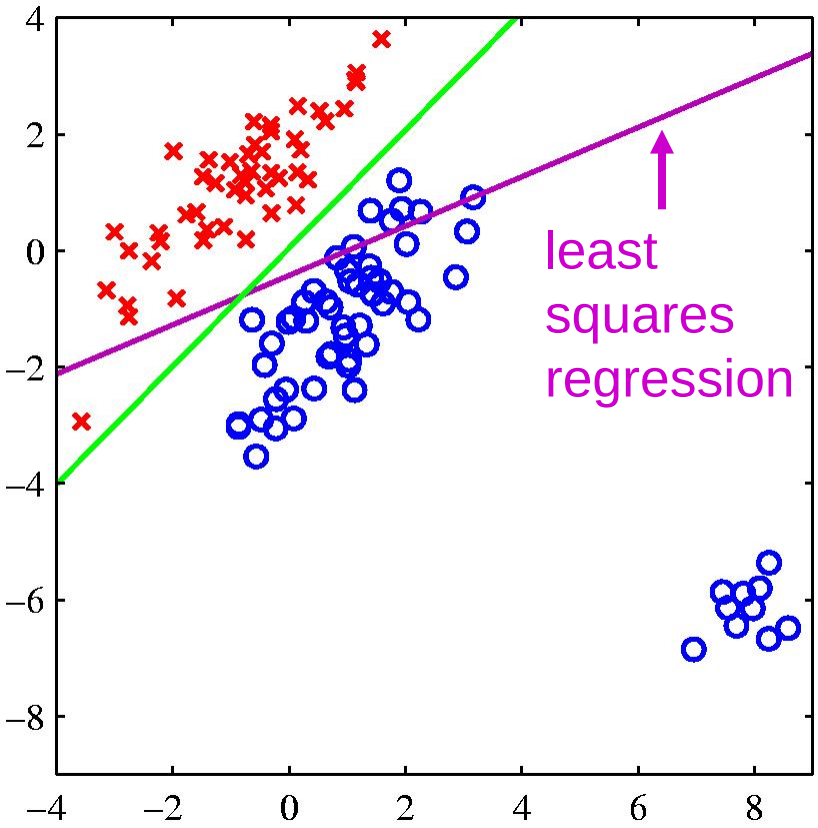




Slide Credit:

Using least squares for classification



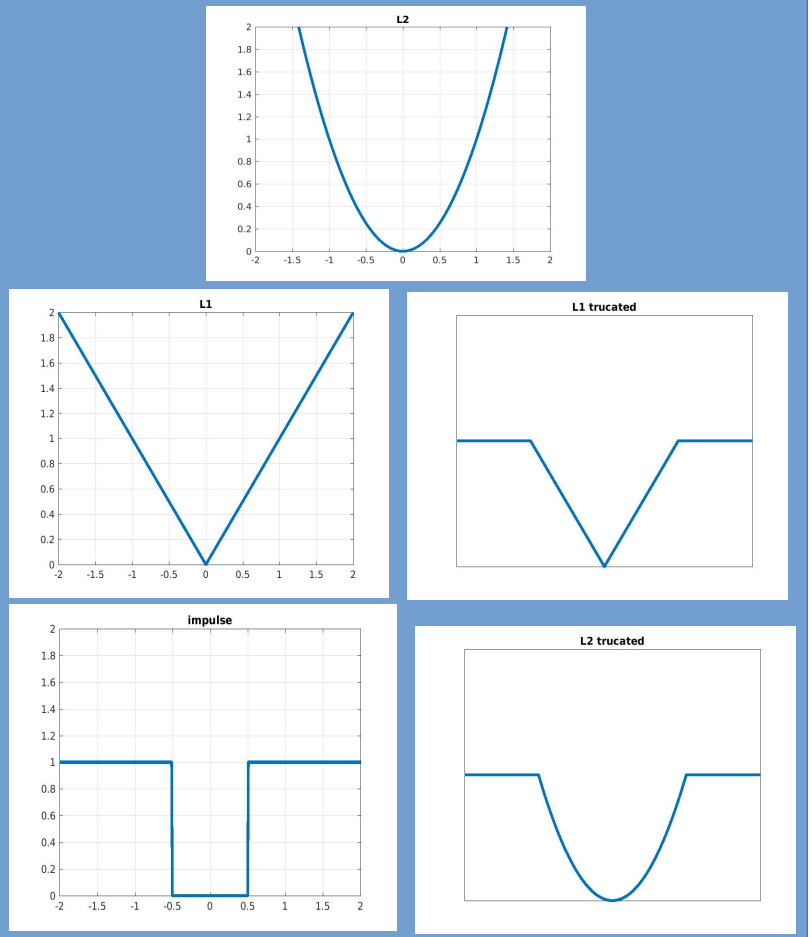


If the right answer is 1 and the model says 1.5, it loses, so it changes the boundary to avoid being “too correct”

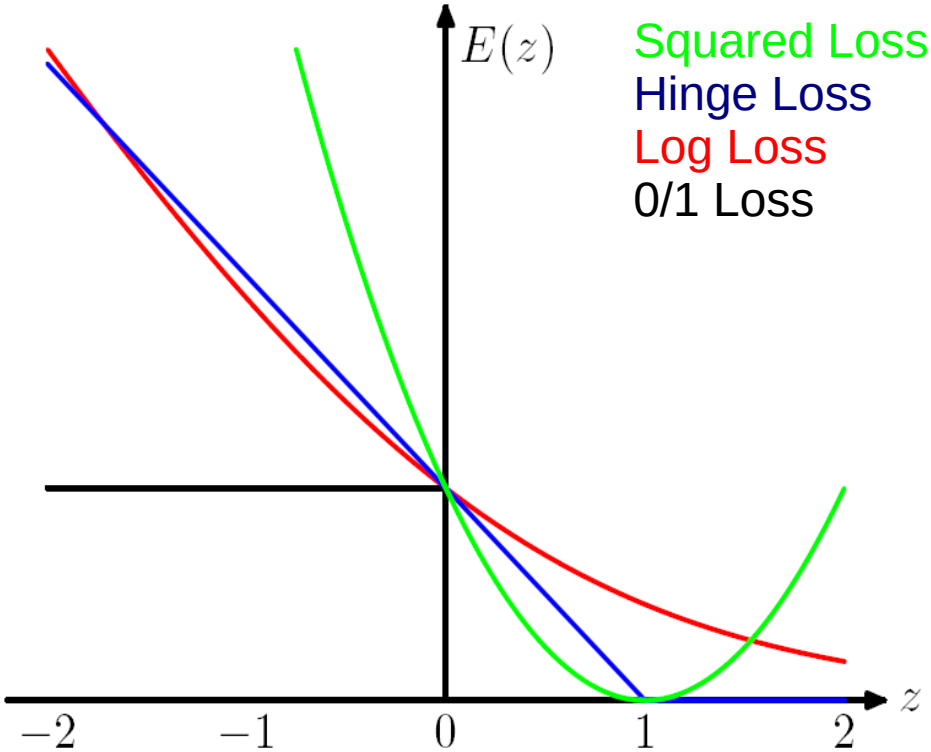
Slide Credit: J. Hinton

The Problem: Loss Function

Recall: Regression Loss Functions



Some Classification Loss Functions



Slide Credit:

Sigmoid

We model the probability of a label Y to be equal $y \in \{-1, 1\}$, given a data point $x \in \mathbf{R}^n$, as:

$$P(Y = y \mid x) = \frac{1}{1 + \exp(-y(w^T x + b))}.$$

This amounts to modeling the *log-odds ratio* as a linear function of X:

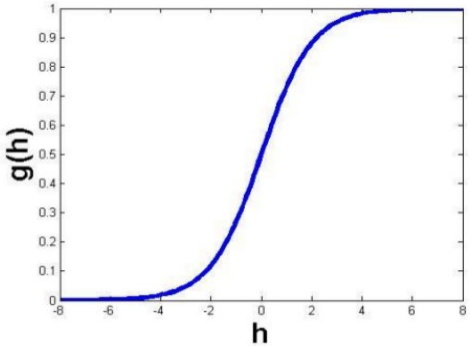
$$\log \frac{P(Y = 1 \mid x)}{P(Y = -1 \mid x)} = w^T x + b.$$

$\frac{}{e^{-h}}$

binary

$\frac{1}{+e^{w^T x}}$

$e^{w^T x}$



- The decision boundary $P(Y = 1 \mid x) = P(Y = -1 \mid x)$ is the hyperplane with equation $w^T x + b = 0$.
- The region $P(Y = 1 \mid x) \geq P(Y = -1 \mid x)$ (*i.e.*, $w^T x + b \geq 0$) corresponds to points with predicted label $\hat{y} = +1$.

Slide Credit: Ziv-Bar Joseph

Log Loss

The likelihood function is

$$l(w,b) = \prod_{i=1}^{m} \frac{1}{1+e^{-y_i(w^T x_i + b)}}.$$

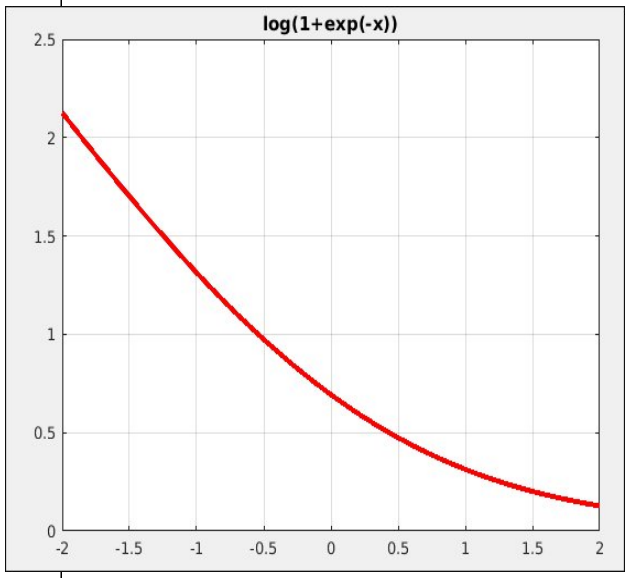
Now maximize the log-likelihood:

$$\max_{w,b} \; L(w,b) := -\sum_{i=1}^{m} \log(1+e^{-y_i(w^T x_i + b)})$$

In practice, we may consider adding a regularization term

$$\max_{w,b} \; L(w,b) + \lambda r(w),$$

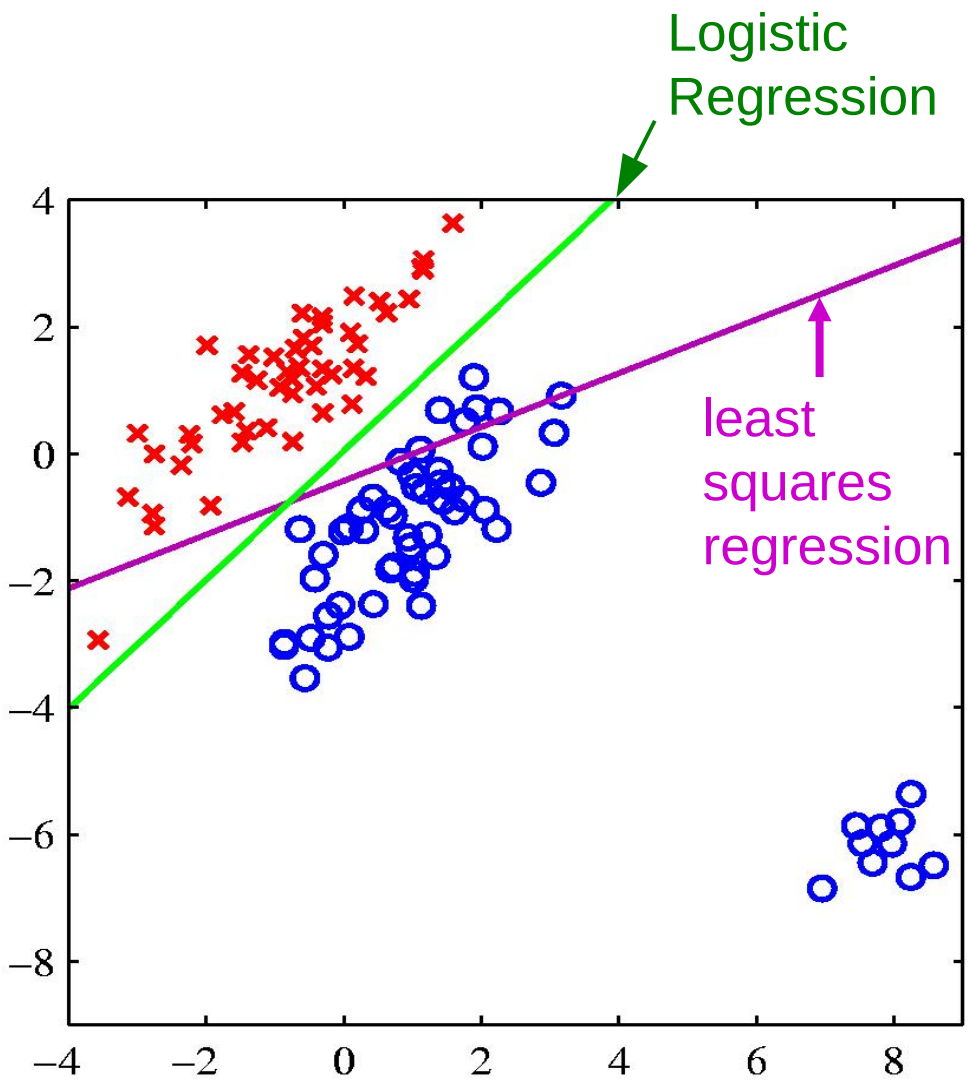
with $r(w) = \|w\|_2^2$ or $r(x) = \|w\|_1$.



The Logistic Loss:
Log(1+ exp(-z))
Where
$z = y*(w^T x+b)$

Slide Credit: Ziv-Bar Joseph

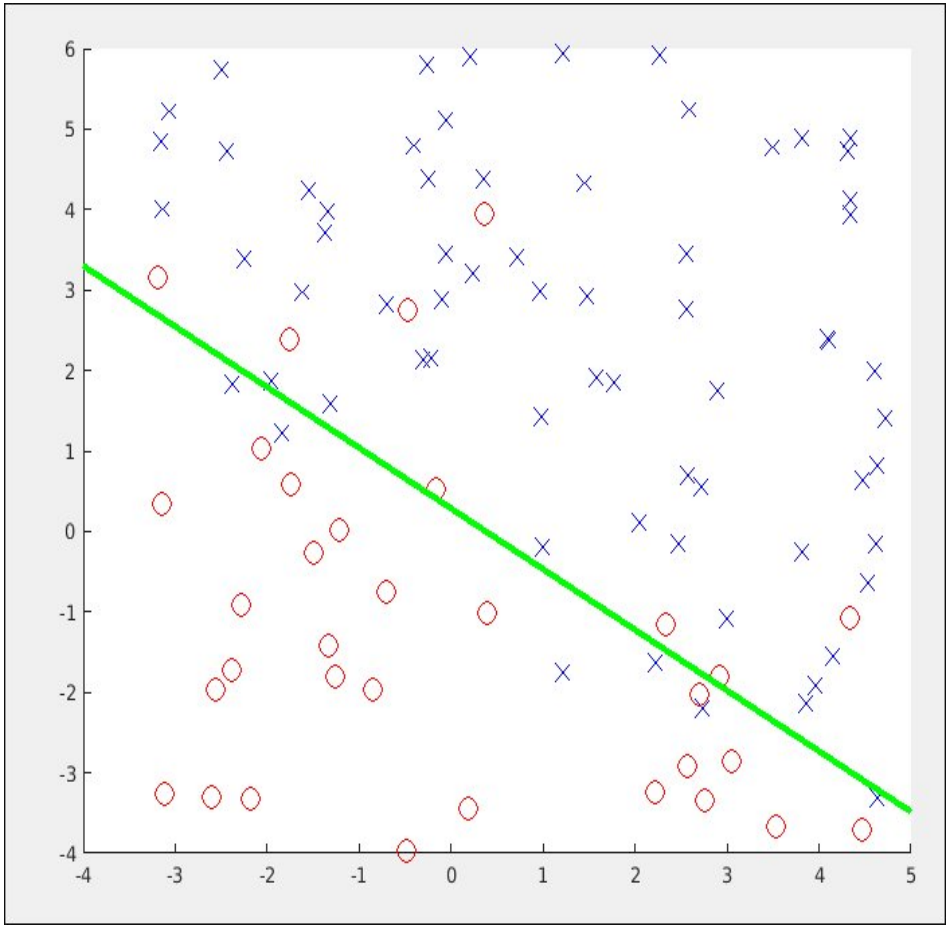
Logistic Result



Slide Credit:

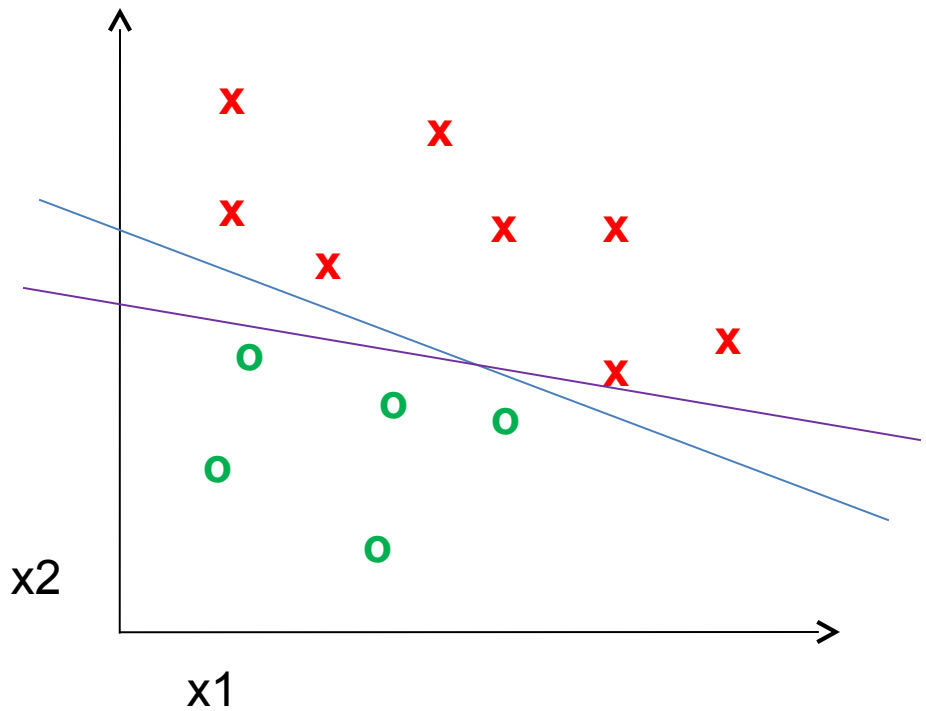
Using Logistic Regression

- Quick, simple classifier (try it first)
- Outputs a probabilistic label confidence
- Use L2 or L1 regularization
  - L1 does feature selection and is robust to irrelevant features but slower to train



Slide Credit:

Classifiers: Linear SVM

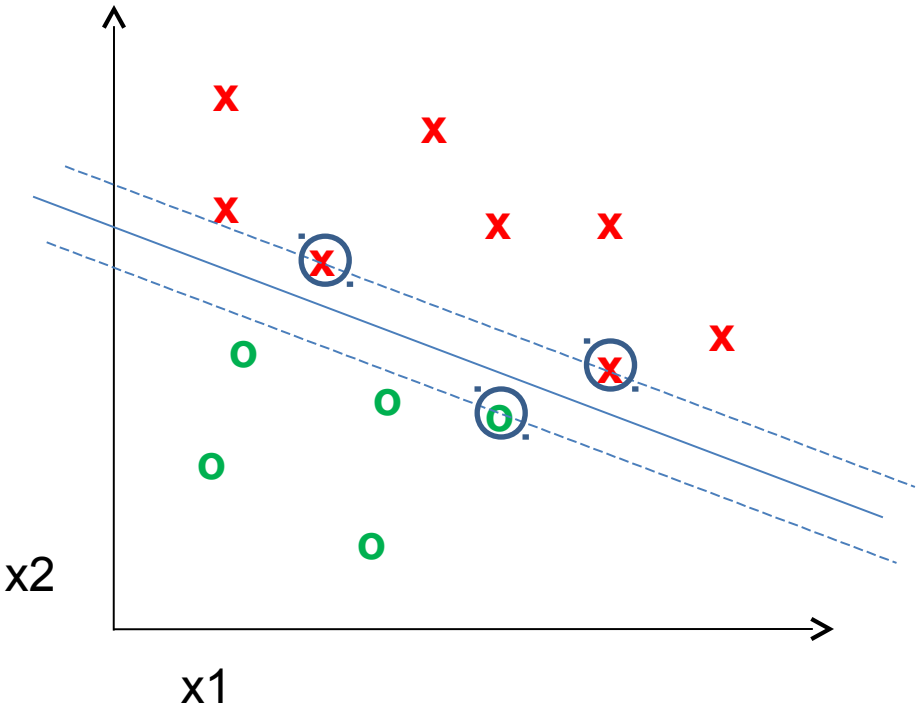


- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Slide Credit:

Classifiers: Linear SVM



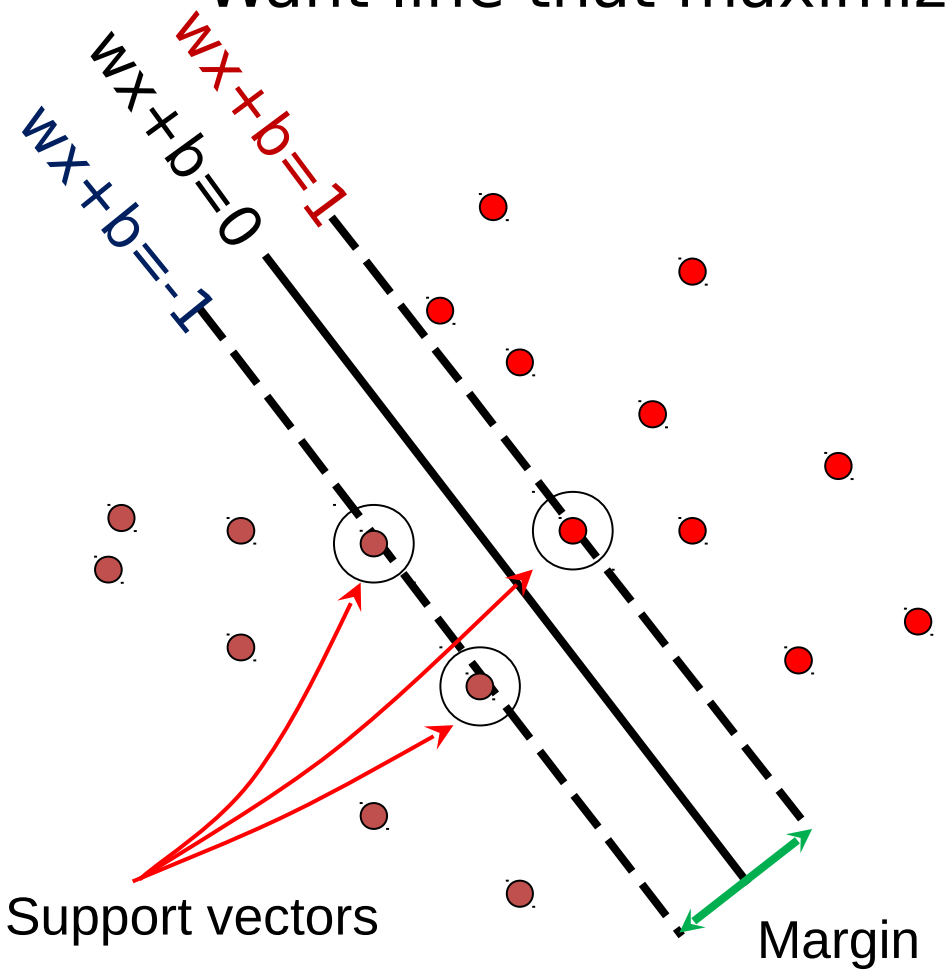
- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \mathrm{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Slide Credit:

Support vector machines: Margin

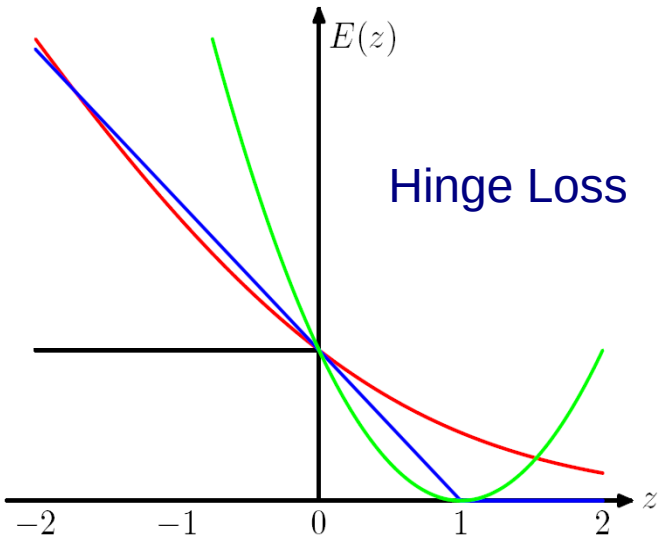
- Want line that maximizes the margin.



$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

For support, vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$



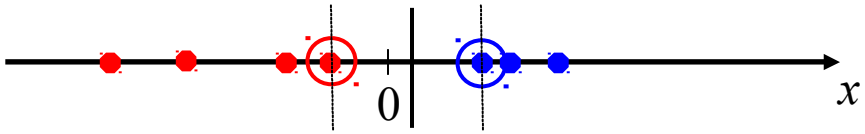
L(y,f(x)) = max(0,1−y·f(x))

C. Burges,
A Tutorial on Support Vector Machines for Pattern Recognition

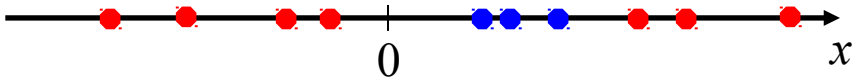
Slide Credit: Kristin Grauman

Nonlinear SVMs

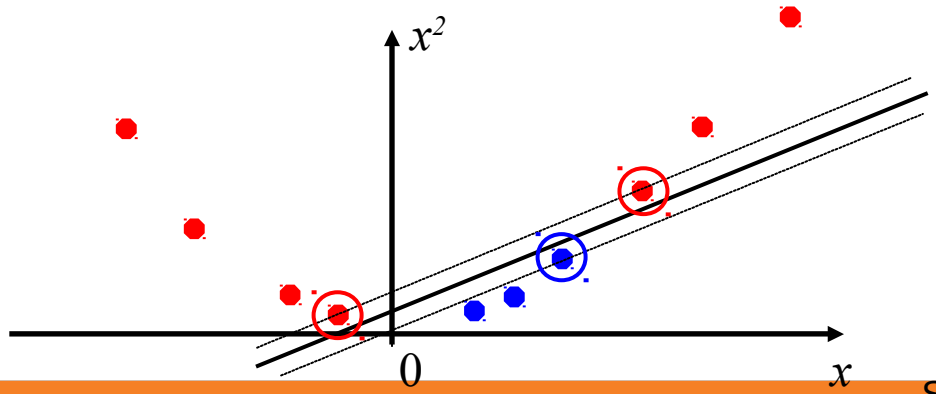
- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?



- We can map it to a higher-dimensional space:

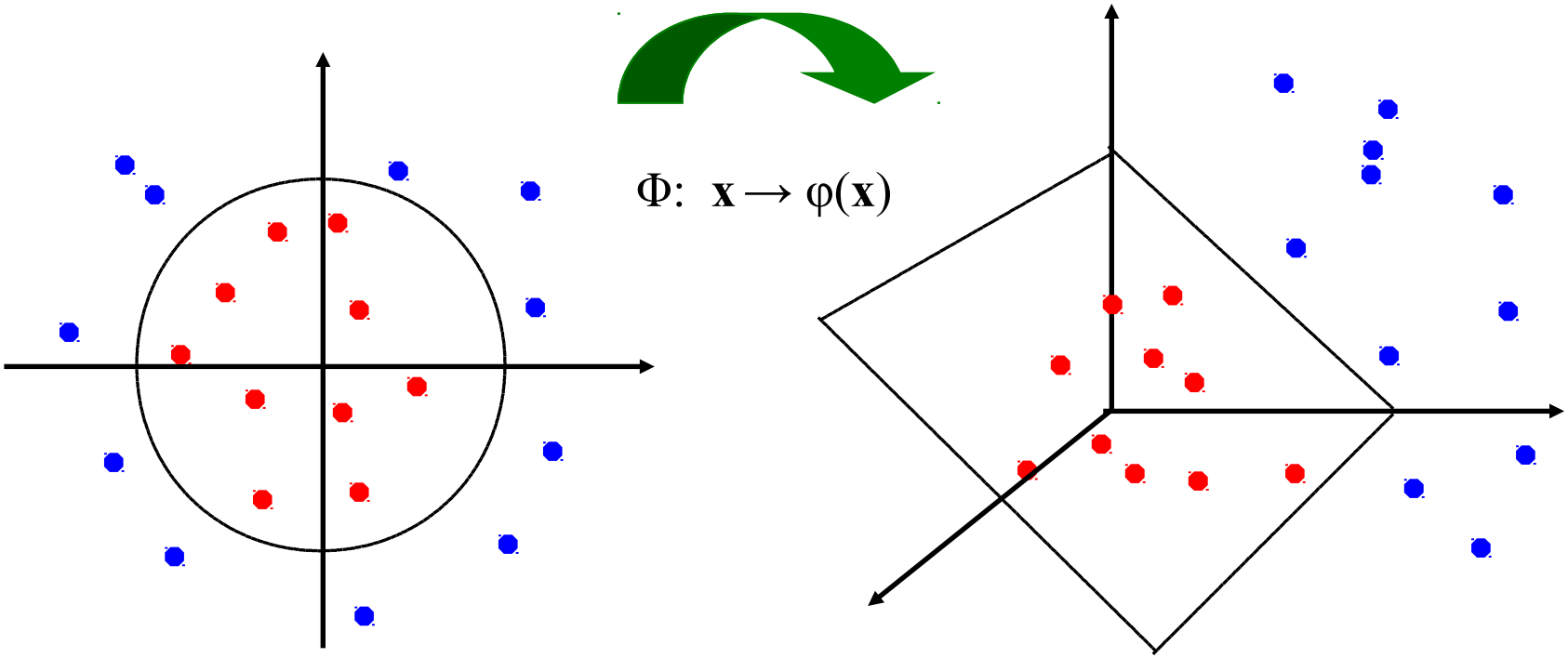


Slide credit: Andrew Moore

Slide Credit:

Nonlinear SVMs

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Slide credit: Andrew Moore

Slide Credit:

Nonlinear SVMs

- *The kernel trick*: instead of explicitly computing the lifting transformation $\boldsymbol{\varphi}(\mathbf{x})$, define a kernel function K such that

$$K(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\varphi}(\mathbf{x}_i) \cdot \boldsymbol{\varphi}(\mathbf{x}_j)$$

  (to be valid, the kernel function must satisfy *Mercer's condition*)

- This gives a nonlinear decision boundary in the original feature space:

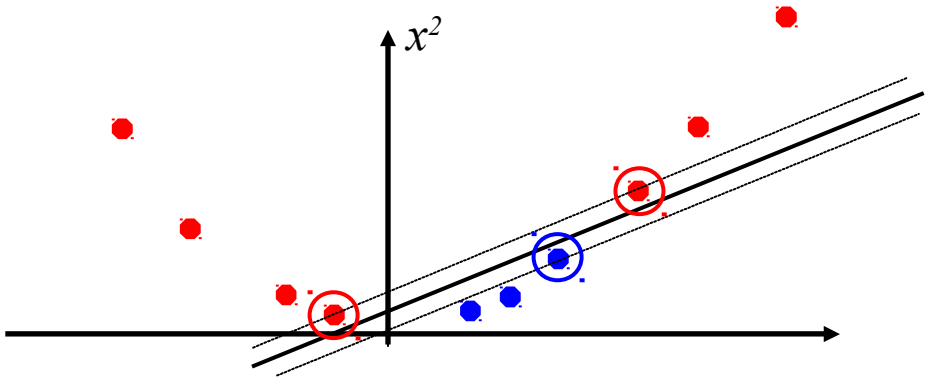
$$\sum_i \alpha_i y_i \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}) + b = \sum_i \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

Slide Credit:

Nonlinear kernel: Example

- Consider the mapping $\phi(x) = (x, x^2)$



$$\phi(x) \cdot \phi(y) = (x, x^2) \cdot (y, y^2) = xy + x^2 y^2$$

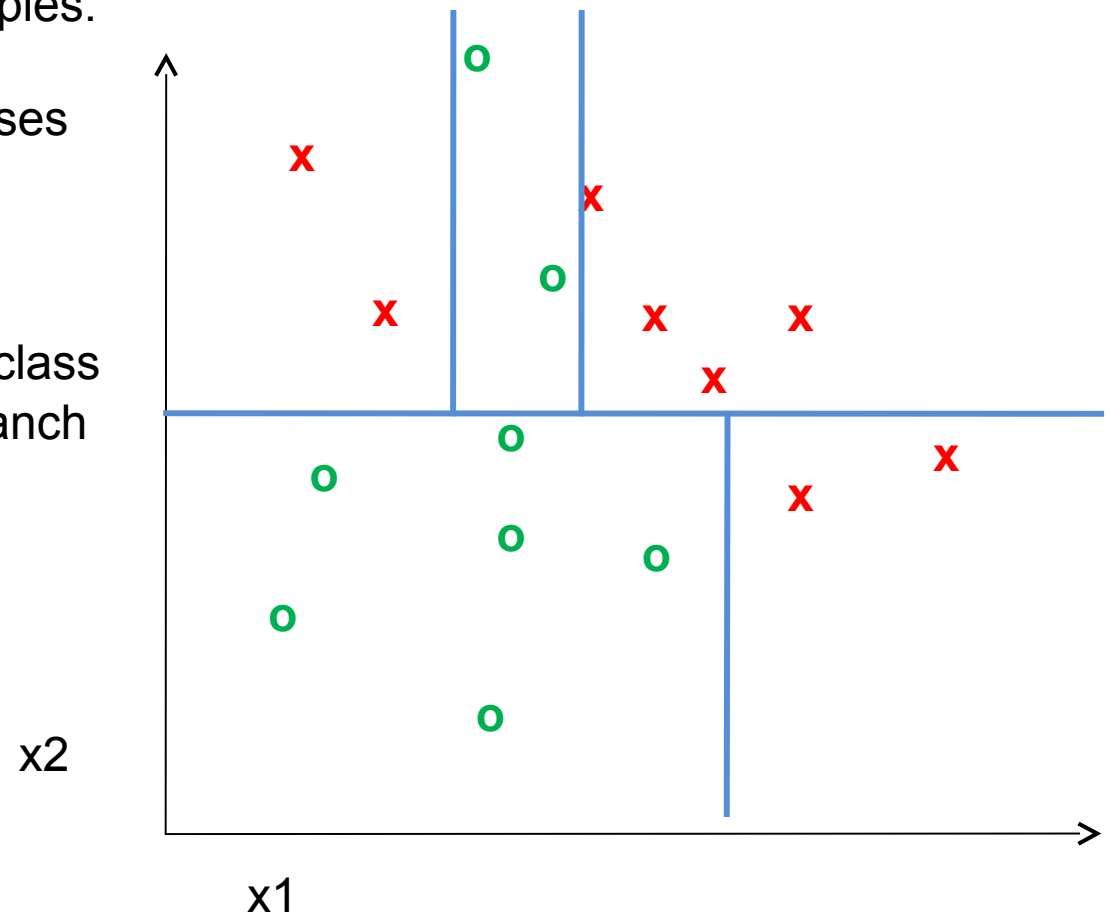
$$K(x, y) = xy + x^2 y^2$$

Slide Credit:

Classifiers: Decision Trees

Given (weighted) labeled examples:

- Select best single feature & threshold that separates classes
- For each branch, recurse
- Stop when
  - some depth is reached
  - Branch is (close to) single-class
  - too few examples left in branch



Slide Credit:

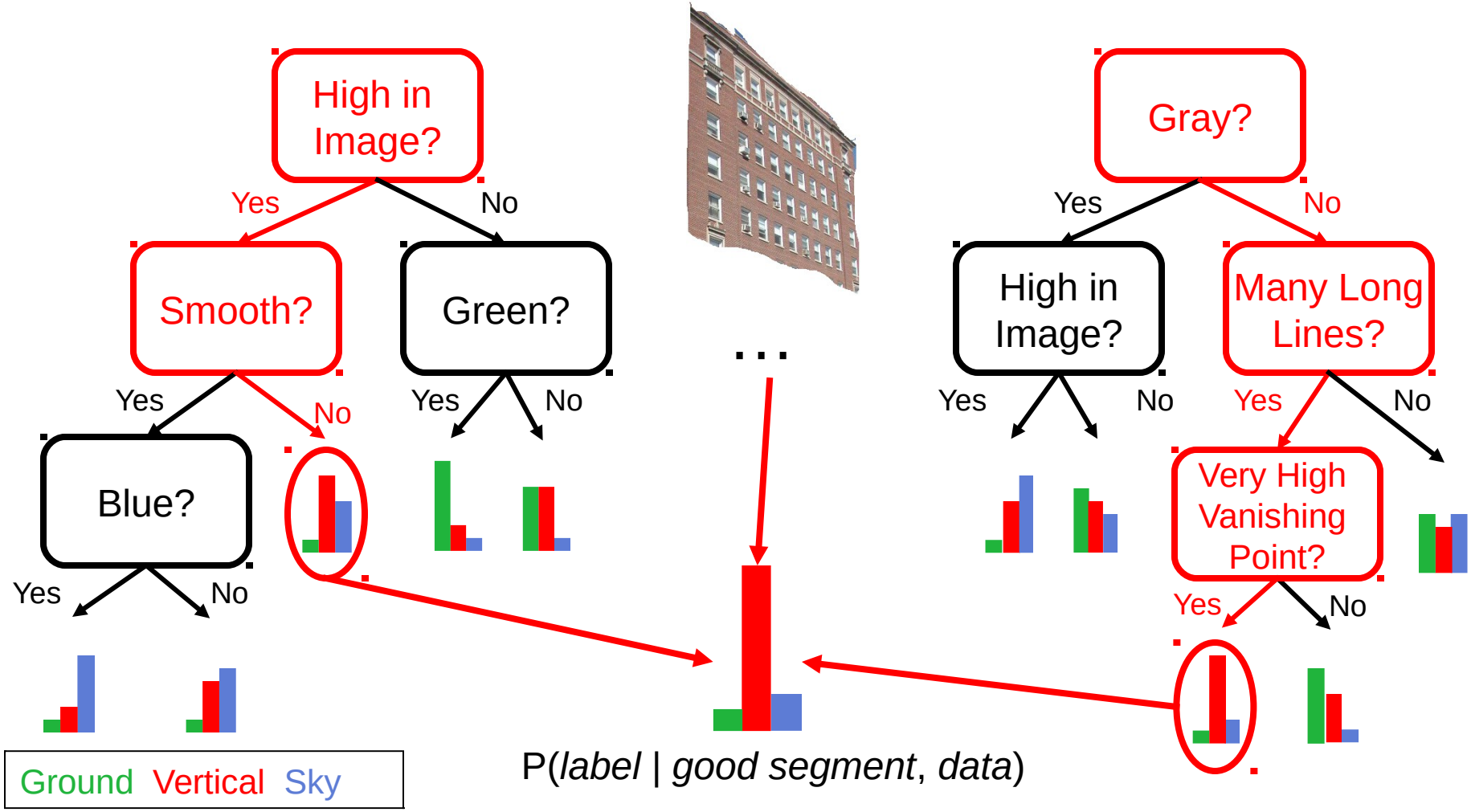
Ensemble Methods: Boosting

Discrete AdaBoost(Freund & Schapire 1996*b*)

1. Start with weights $w_i = 1/N$, $i = 1, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights w_i on the training data.
   (b) Compute $\mathrm{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m \cdot 1_{(y_i \neq f_m(x_i))}]$, $i = 1, 2, \ldots N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\mathrm{sign}[\sum_{m=1}^{M} c_m f_m(x)]$

Slide Credit: figure from Friedman et al. 2000

Boosted Decision Trees



P(*label* | *good segment*, *data*)

Slide Credit: [Collins et al. 2002]

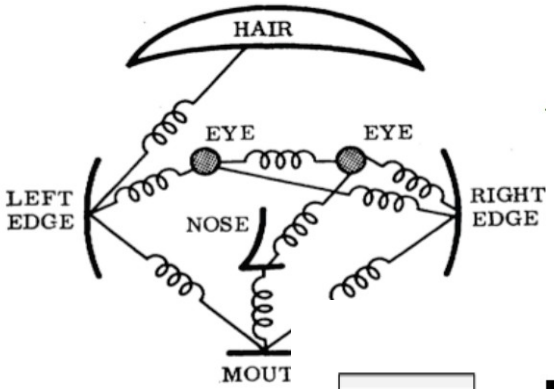
Using Boosted Decision Trees

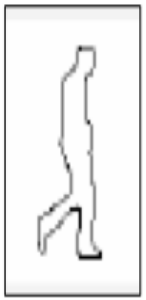
- Flexible: can deal with both continuous and categorical variables
- How to control bias/variance trade-off
  - Size of trees
  - Number of trees
- Boosting trees often works best with a small number of well-designed features
- Boosting “stubs” can give a fast classifier

Slide Credit:

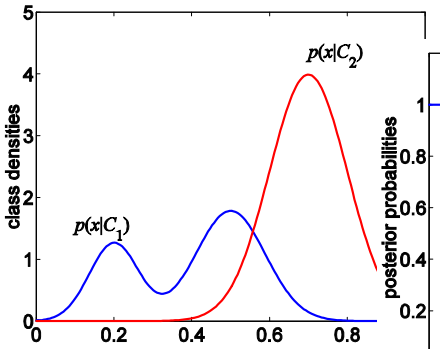
Review: Typical Components

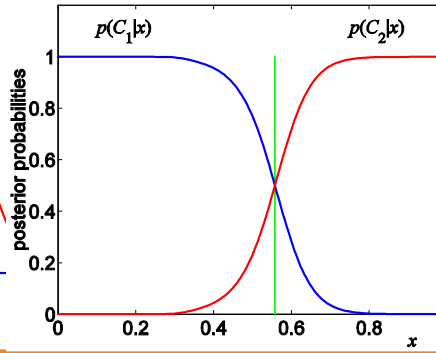
- **Hypothesis** generation
  - Sliding window, Segmentation, feature point detection, random, search
- **Encoding** of (local) image data
  - Colors, Edges, Corners, Histogram of Oriented Gradients, Wavelets, Convolution Filters
- **Relationship** of different parts to each other
  - Blur or histogram, Tree/Star, Pairwise/Covariance
- **Learning** from labeled examples
  - Selecting representative examples (templates), Clustering, Building a cascade
  - Classifiers: Bayes, Logistic regression, SVM, Decision Trees, AdaBoost, ...
  - Generative vs. Discriminative
- **Verification -** removing redundant, overlaping, incompatible examples
    - Non-Max Suppression, context priors, geometry





Exemplar Summary





Slide Credit:

Discriminative vs. Generative Classifiers

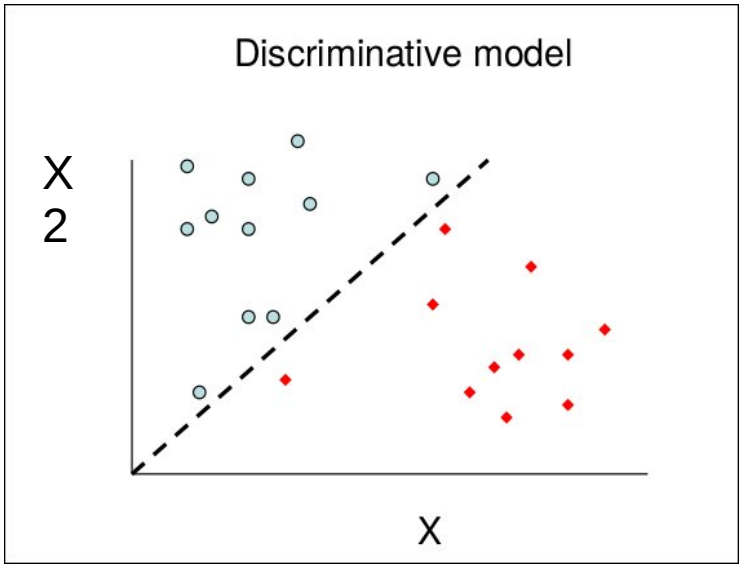
Training classifiers involves estimating f: X → Y, or P(Y|X) "Y given X"

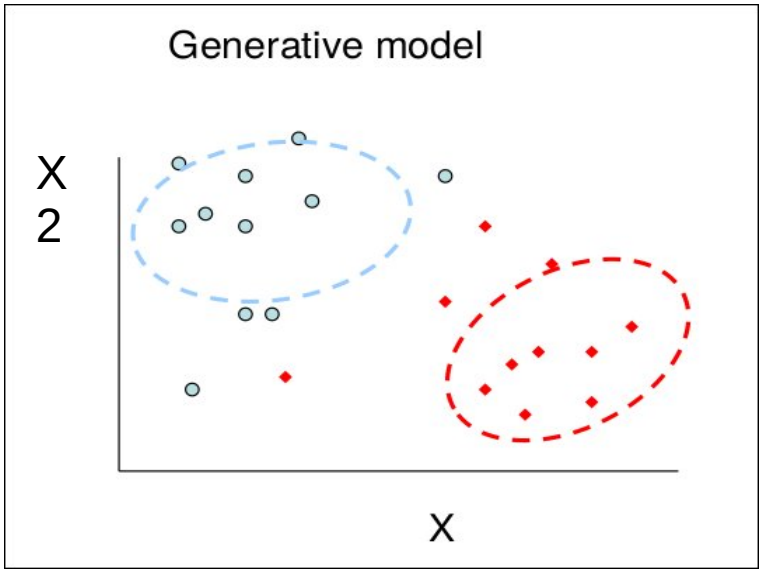
Discriminative Classification:
Find the boundary between classes

1. Assume some functional form for P(Y|X)
2. Estimate parameters of P(Y|X) directly from training data

Generative Classification:
Model each class & see which fits better

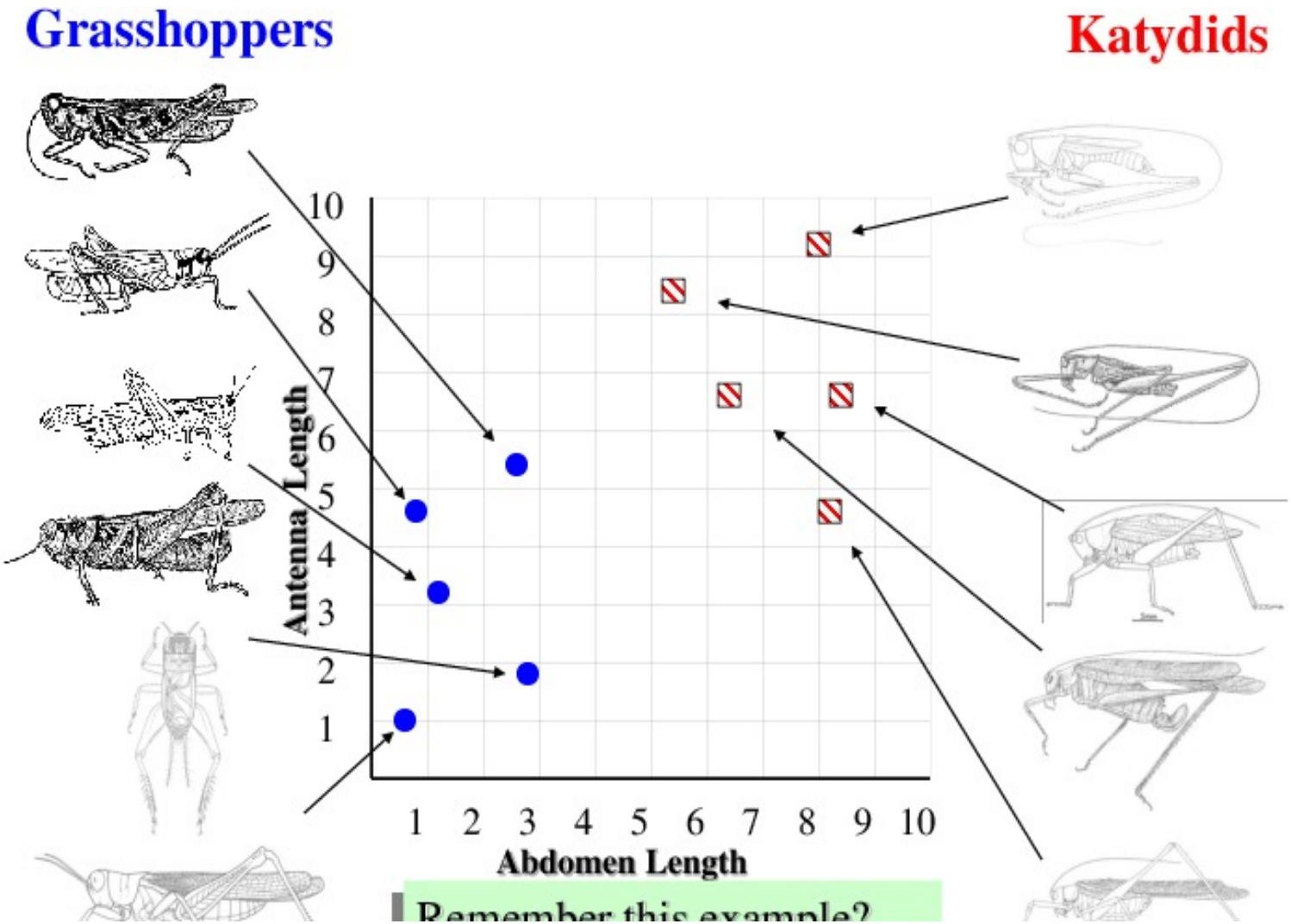
1. Assume some functional form for P(X|Y), P(X)
2. Estimate parameters of P(X|Y), P(X) directly from training data
3. Use Bayes rule to calculate P(Y|X= x_i)





Slide Credit:

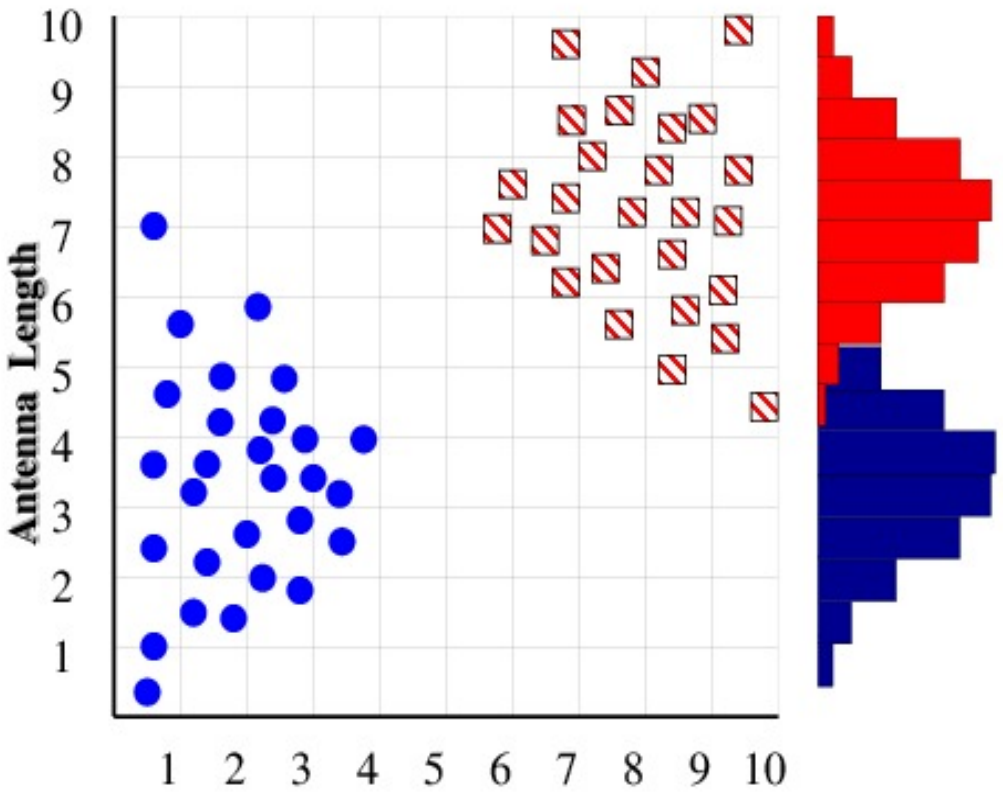
Bayesian Classification (Generative Model)



Slide Credit: Eamonn Keogh

Bayesian Classification

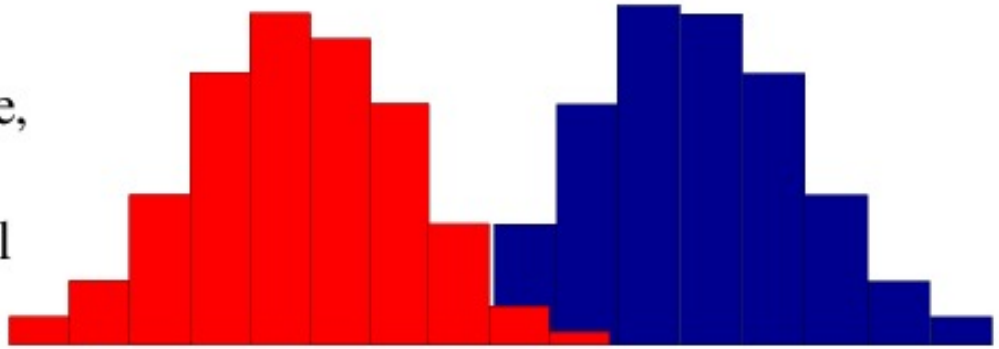
With a lot of data, we can build a histogram. Let us just build one for "Antenna Length" for now....



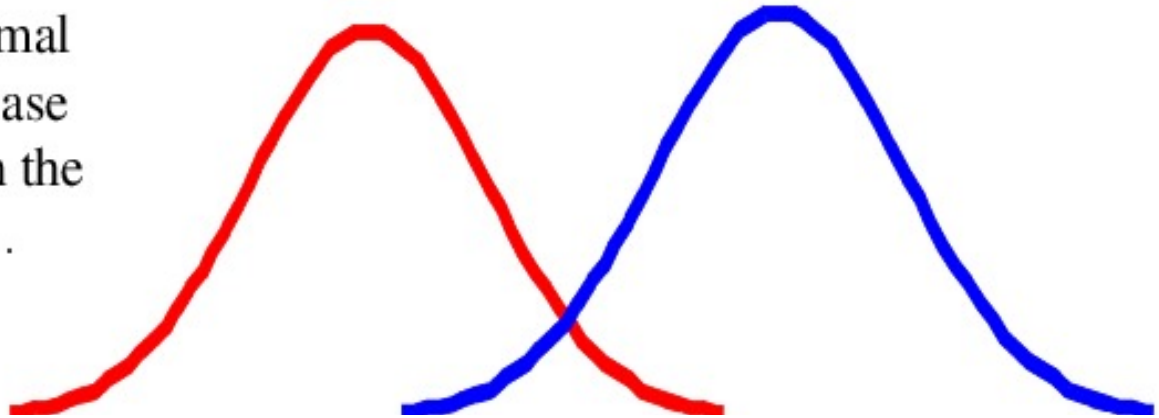
Slide Credit: Eamonn Keogh

Bayesian Classification

We can leave the histograms as they are, or we can summarize them with two normal distributions.



Let us us two normal distributions for ease of visualization in the following slides…



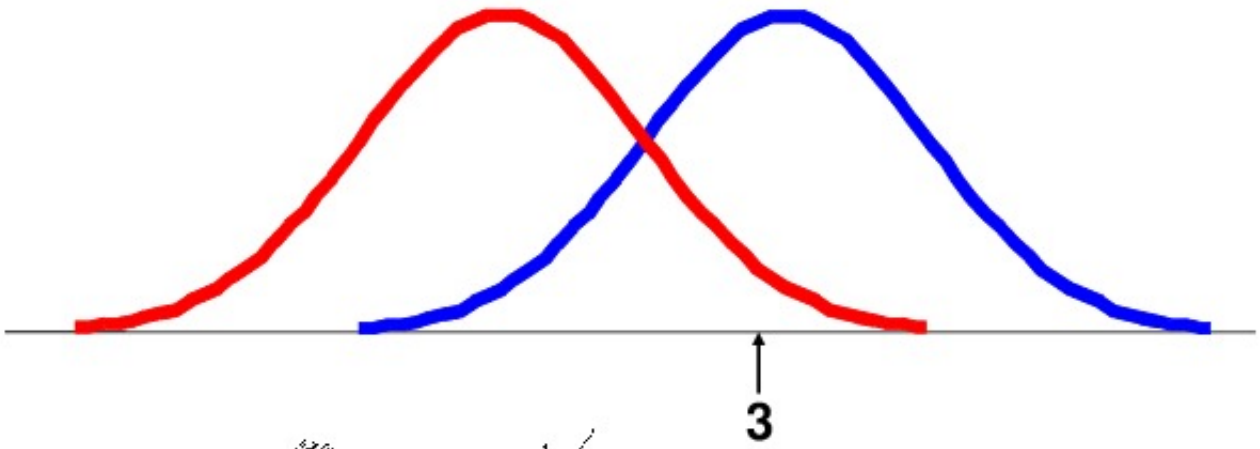
Slide Credit: Eamonn Keogh

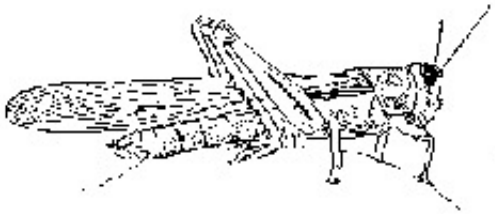
Bayesian Classification

- We want to classify an insect we have found. Its antennae are 3 units long. How can we classify it?

- We can just ask ourselves, give the distributions of antennae lengths we have seen, is it more probable that our insect is a **Grasshopper** or a **Katydid**.
- There is a formal way to discuss the most probable classification…

$p(c_j \mid d)$ = probability of class c_j, given that we have observed d





Antennae length is **3**

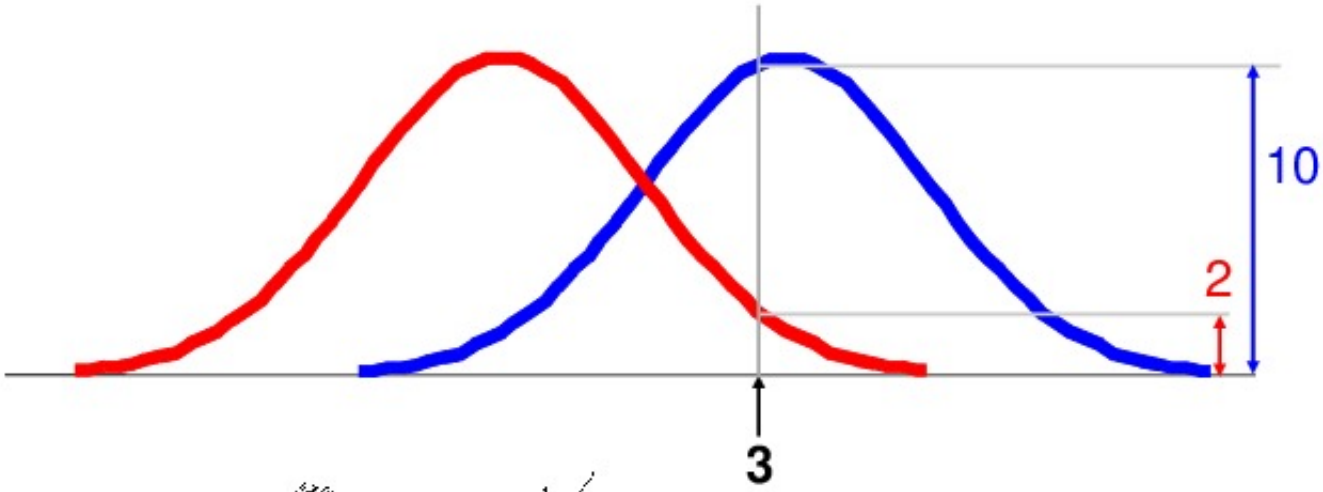
Slide Credit: Eamonn Keogh

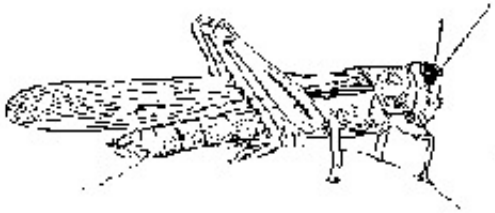
Bayesian Classification

$p(c_j \mid d)$ = probability of class c_j, given that we have observed d

P(**Grasshopper** | **3**) = 10 / (10 + 2) = 0.833

P(**Katydid** | **3**) = 2 / (10 + 2) = 0.166





Antennae length is **3**

Slide Credit: Eamonn Keogh

Bayesian Classification

- Bayesian classifiers use **Bayes theorem**, which says

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

- $p(c_j \mid d)$ = probability of instance d being in class c_j,

  This is what we are trying to compute

- $p(d \mid c_j)$ = probability of generating instance d given class c_j,

  We can imagine that being in class c_j, causes you to have feature d with some probability

- $p(c_j)$ = probability of occurrence of class c_j,

  This is just how frequent the class c_j, is in our database

- $p(d)$ = probability of instance d occurring

  This can actually be ignored, since it is the same for all classes

Slide Credit: Eamonn Keogh

Naïve Bayes Classification

- To simplify the task, **naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$p(d|c_j) = p(d_1|c_j) * p(d_2|c_j) * \ldots * p(d_n|c_j)$$

The probability of class c_j generating instance d, equals….

The probability of class c_j generating the observed value for feature 1, multiplied by..

The probability of class c_j generating the observed value for feature 2, multiplied by..

Slide Credit: Eamonn Keogh

Naïve Bayes, Odds ratio, and Logit

Assuming independence of features d1 and d2, we can classify between 2 classes {C1, C2}, by compute the ratio:

$$\frac{P(C_1|d_1,d_2)}{P(C_2|d_1,d_2)} = \frac{P(d_1|C_1)P(d_2|C_1)P(C_1)}{P(d_1|C_2)P(d_2|C_2)P(C_2)} < 1$$

This is called the Odds ratio.

It is often easier to take the log of this, called the Log Odds or Logit:

$$\log \frac{P(C_1|d_1,d_2)}{P(C_2|d_1,d_2)}$$

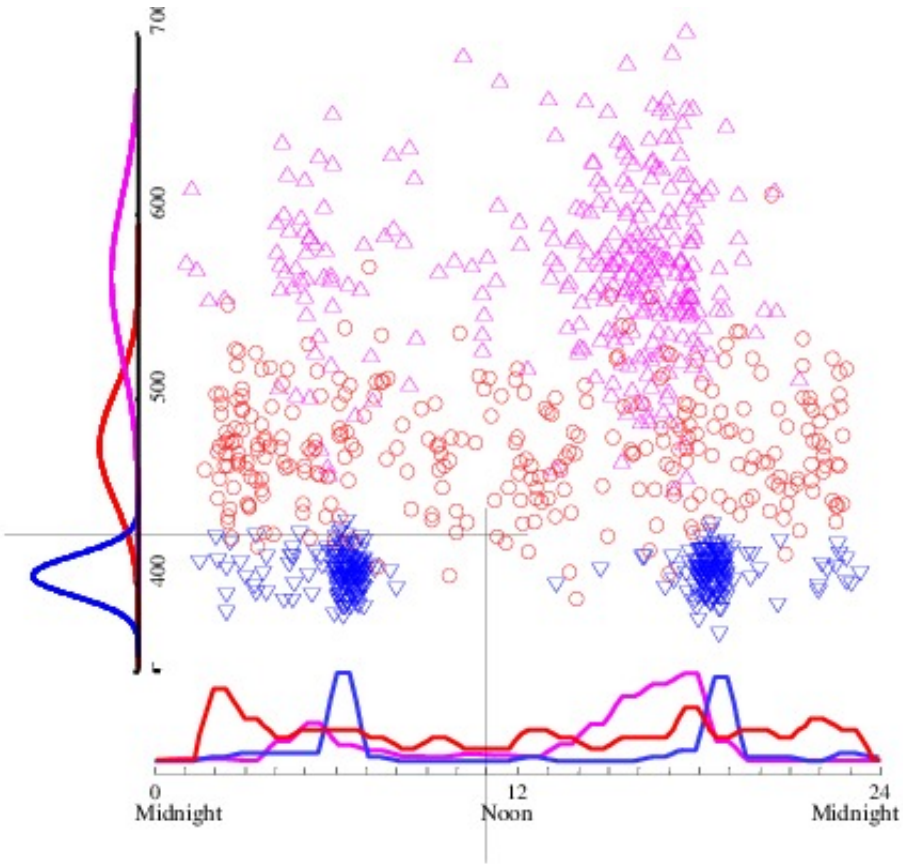
$$= \log \frac{P(d_1|C_1)}{P(d_1|C_2)} + \log \frac{P(d_2|C_1)}{P(d_2|C_2)} + \log \frac{P(C_1)}{P(C_2)} < 0$$

Slide Credit:

Naïve Bayes

Suppose I observe an insect with a wingbeat frequency of 420 at 11:00am

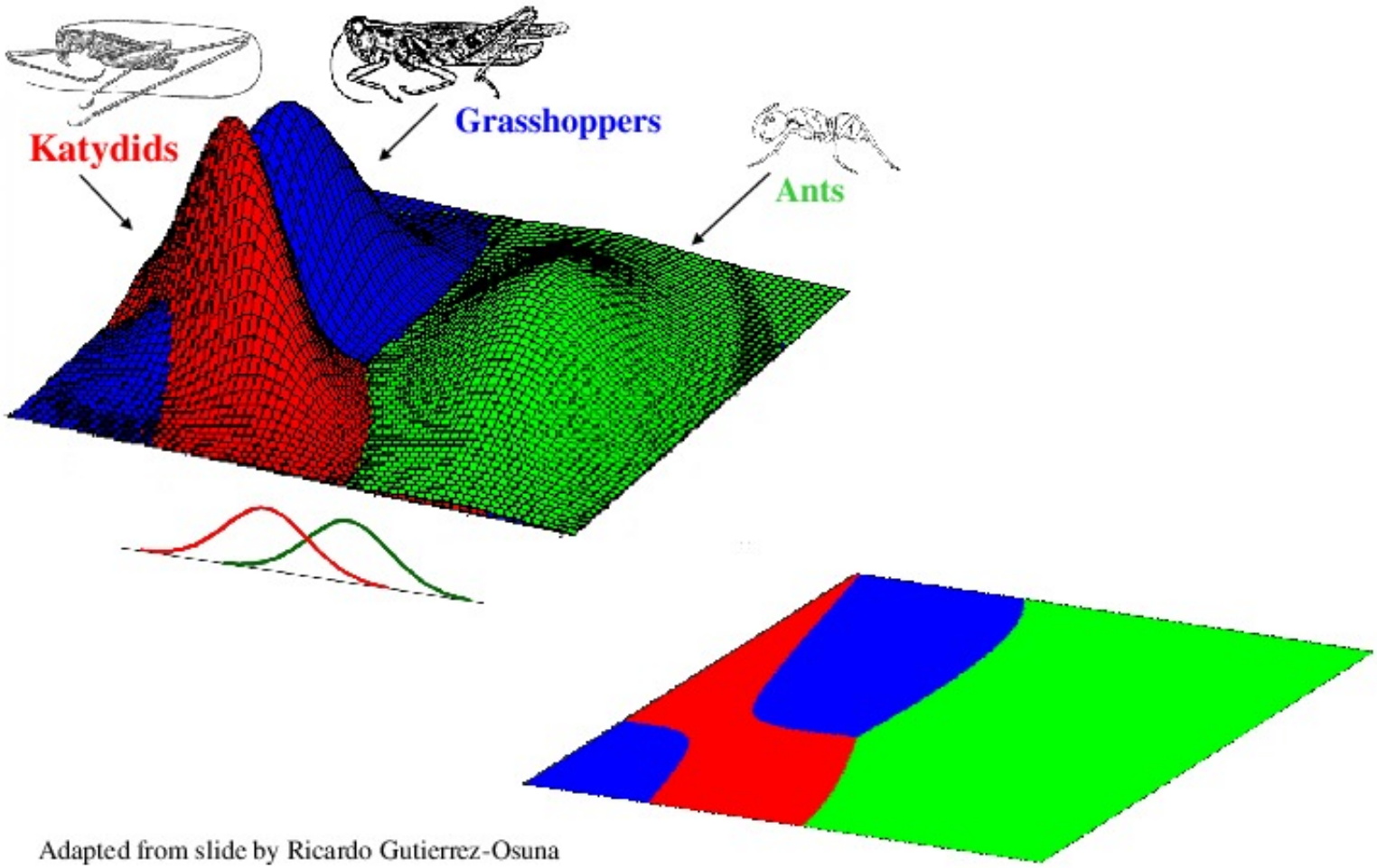
What is it?



Slide Credit: Eamonn Keogh

Naïve Bayes

Naive Bayes with Gaussian densities have piecewise quadratic decision boundary.



Adapted from slide by Ricardo Gutierrez-Osuna

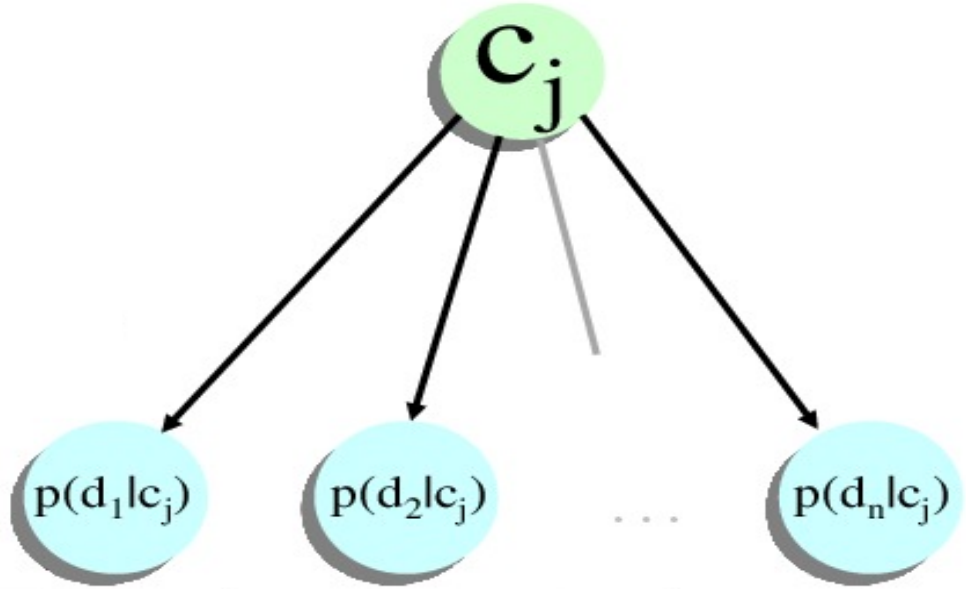
Slide Credit: Eamonn Keogh

Graphical Models

Independence assumption of Naive Bayes is the simplest assumption. Often much more complicated relationships needs to be represented.

A good way to do this is a Graphical Model
(aka. Byesian Network)

Naive Bayes assumes independence of d1,d2,... conditioned on the class c



Animal	Mass >10kg	
Cat	Yes	0.15
	No	0.85
Dog	Yes	0.91
	No	0.09
Pig	Yes	0.99
	No	0.01

Animal	Color	
Cat	Black	0.33
	White	0.23
	Brown	0.44
Dog	Black	0.97
	White	0.03
	Brown	0.90
Pig	Black	0.04
	White	0.01

Animal
Cat
Dog
Pig

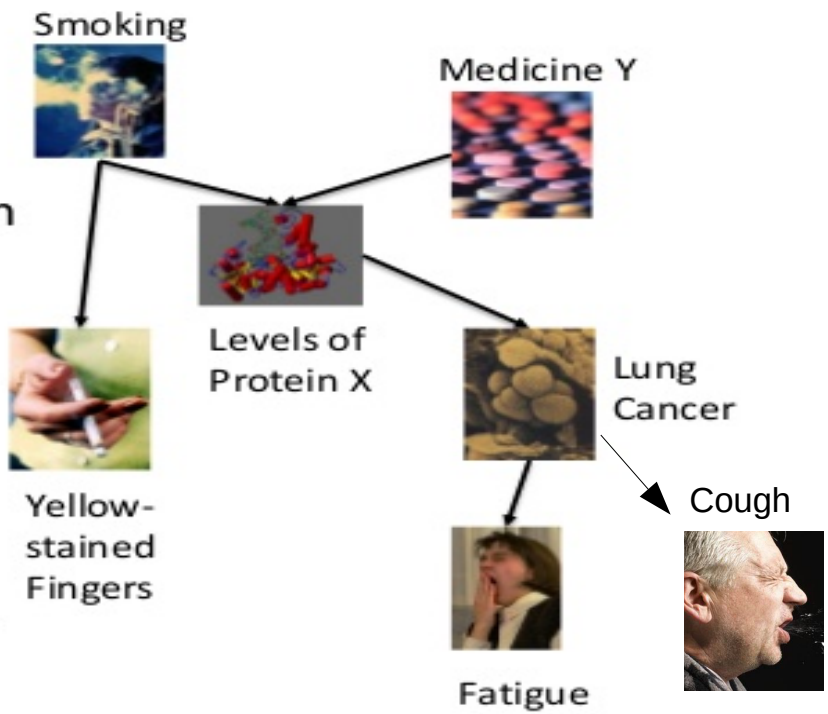
Slide Credit: Eamonn Keogh

Graphical Models

More complicated models consider the dependence between different variables.

Bayesian Network

1. Factorize the jpd
2. Answer questions like:
   1. P(Lung Cancer| Levels of Protein X) = ?
   2. Ind(Smoking, Fatigue| Levels of Protein X)?
   3. What will happen if I design a drug that blocks the function of protein X (**predict effect of interventions**)?



Slide Credit: Ioannis Tsamardinos

Discriminative vs. Generative Classifiers

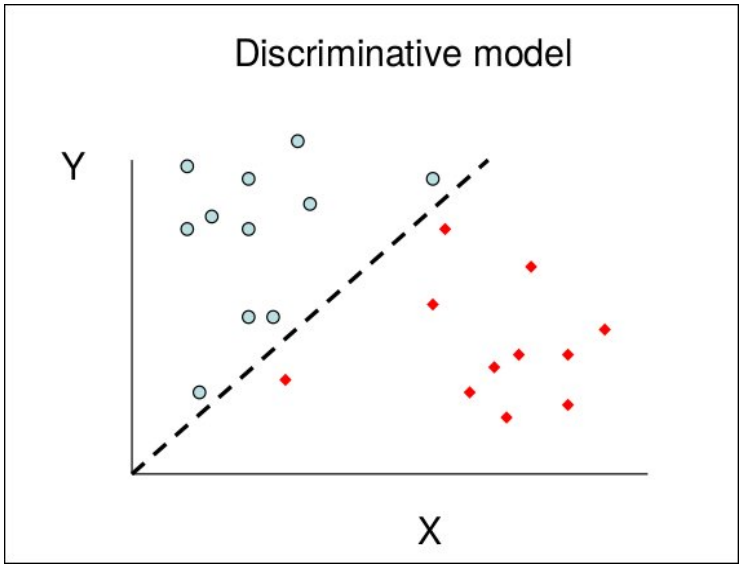
Training classifiers involves estimating f: X → Y, or P(Y|X) "Y given X"

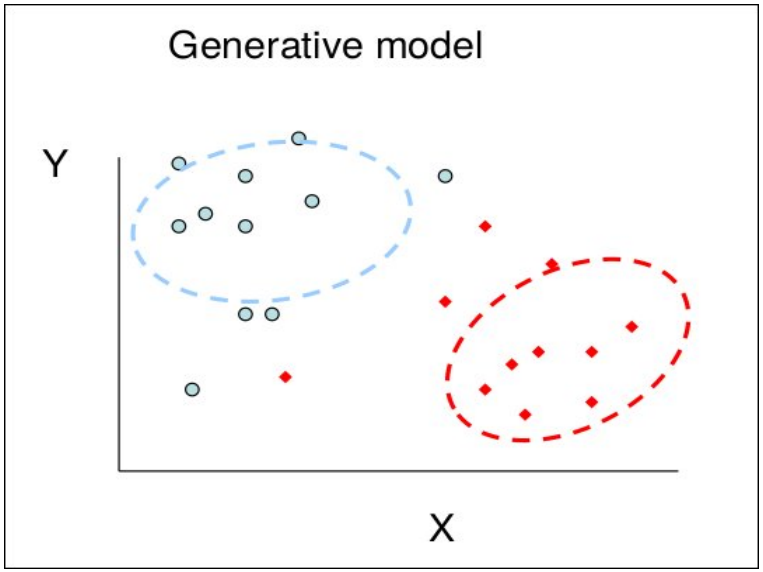
Discriminative Classification:
Find the boundary between classes

1. Assume some functional form for P(Y|X)
2. Estimate parameters of P(Y|X) directly from training data

Generative Classification:
Model each class & see which fits better

1. Assume some functional form for P(X|Y), P(X)
2. Estimate parameters of P(X|Y), P(X) directly from training data
3. Use Bayes rule to calculate P(Y|X= x_i)

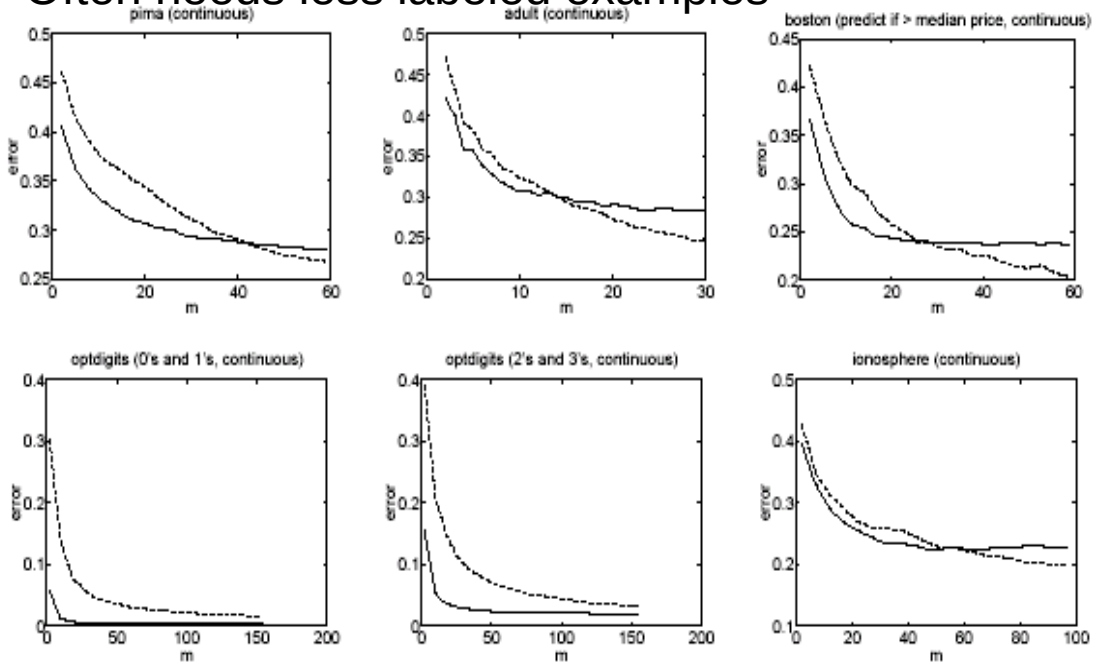




Slide Credit:

Discriminative vs. Generative Classifiers

Discriminative:
+ Model directly what you care about
+ With many examples usually more accurate
+ Often faster to evaluate, can scale well to many examples & classes
Generative:
+ Allows more flexibility to model relationships between variables
+ Can handle compositionality, missing & occluded parts (more “object oriented”)
+ Often needs less labeled examples



“On Discriminative vs. Generative classifiers: A comparison of logistic regression and naïve Bayes,” A. Ng and M. Jordan, NIPS 2002.

Slide Credit:

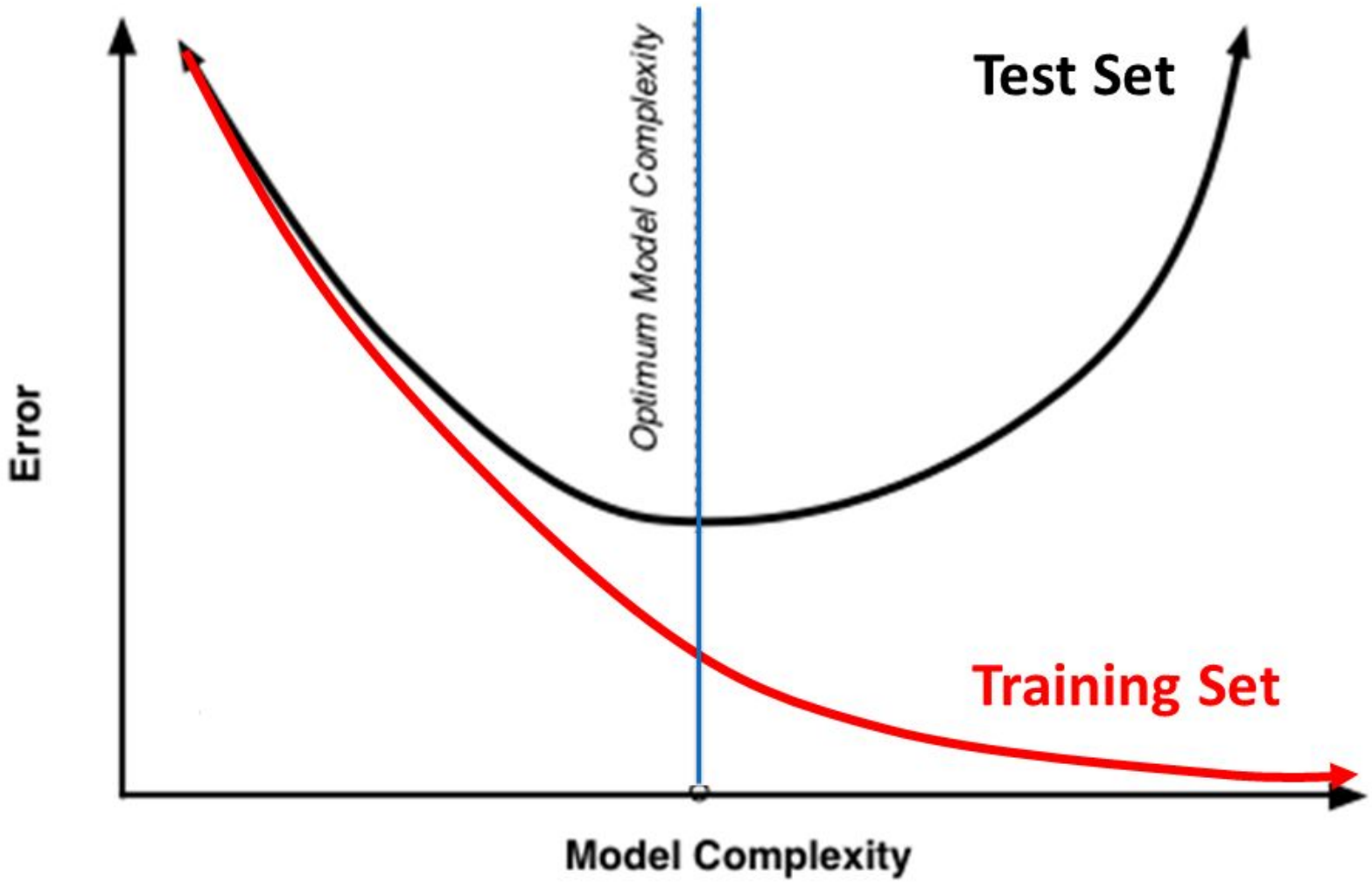
Comparison

assuming x in {0 1}

	Learning Objective	Training	Inference
Naïve Bayes	$\text{maximize} \sum_i \left[\sum_j \log P(x_{ij} \mid y_i; \theta_j) + \log P(y_i; \theta_0) \right]$	$\theta_{kj} = \frac{\sum_i \delta(x_{ij} = 1 \wedge y_i = k) + r}{\sum_i \delta(y_i = k) + Kr}$	$\boldsymbol{\theta}_1^T \mathbf{x} + \boldsymbol{\theta}_0^T (1 - \mathbf{x}) > 0$ where $\theta_{1j} = \log \frac{P(x_j = 1 \mid y = 1)}{P(x_j = 1 \mid y = 0)}$, $\theta_{0j} = \log \frac{P(x_j = 0 \mid y = 1)}{P(x_j = 0 \mid y = 0)}$
Logistic Regression	$\text{maximize} \sum_i \log(P(y_i \mid \mathbf{x}, \boldsymbol{\theta})) + \lambda \lVert \boldsymbol{\theta} \rVert$ where $P(y_i \mid \mathbf{x}, \boldsymbol{\theta}) = 1/(1 + \exp(-y_i \boldsymbol{\theta}^T \mathbf{x}))$	Gradient ascent	$\boldsymbol{\theta}^T \mathbf{x} > 0$
Linear SVM	$\text{minimize } \lambda \sum_i \xi_i + \frac{1}{2} \lVert \boldsymbol{\theta} \rVert$ such that $y_i \boldsymbol{\theta}^T \mathbf{x} \geq 1 - \xi_i \quad \forall i$	Linear programming	$\boldsymbol{\theta}^T \mathbf{x} > 0$
Kernelized SVM	complicated to write	Quadratic programming	$\sum_i y_i \alpha_i K(\hat{\mathbf{x}}_i, \mathbf{x}) > 0$
Nearest Neighbor	most similar features → same label	Record data	y_i where $i = \text{argmin}_i \; K(\hat{\mathbf{x}}_i, \mathbf{x})$

Slide Credit: Slide credit: D. Hoiem

Train vs. Test Accuracy



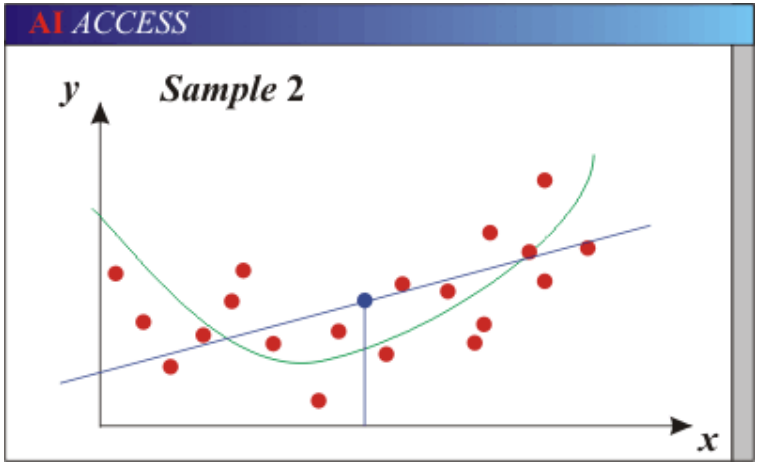
Slide Credit:

Generalization

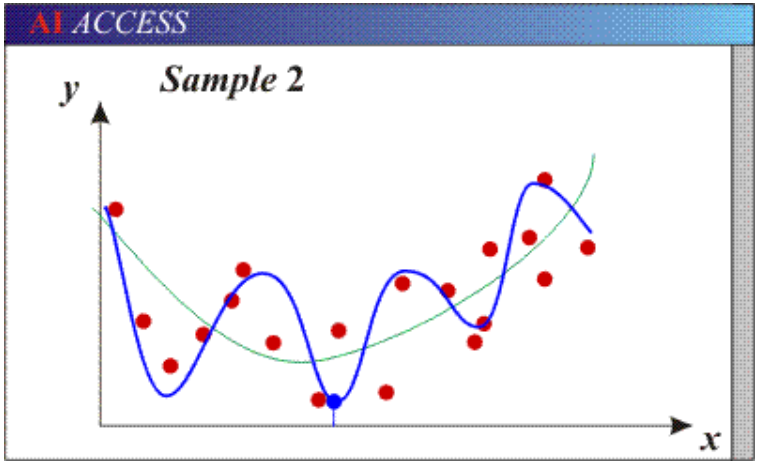
- Components of generalization error
  - **Bias:** how much the average model over all training sets differ from the true model?
    - Error due to inaccurate assumptions/simplifications made by the model
  - **Variance:** how much models estimated from different training sets differ from each other
- **Underfitting:** model is too “simple” to represent all the relevant class characteristics
  - High bias and low variance
  - High training error and high test error
- **Overfitting:** model is too “complex” and fits irrelevant characteristics (noise) in the data
  - Low bias and high variance
  - Low training error and high test error

Slide Credit: Slide credit: L. Lazebnik

Bias-Variance Trade-off



- Models with too few parameters are inaccurate because of a large bias (not enough flexibility).



- Models with too many parameters are inaccurate because of a large variance (too much sensitivity to the sample).

Slide Credit: Slide credit: D. Hoiem

Bias-Variance Trade-off

$$E(MSE) = \text{noise}^2 + \text{bias}^2 + \text{variance}$$

Unavoidable error

Error due to incorrect assumptions

Error due to variance of training samples

See the following for explanations of bias-variance (also Bishop's "Neural Networks" book):

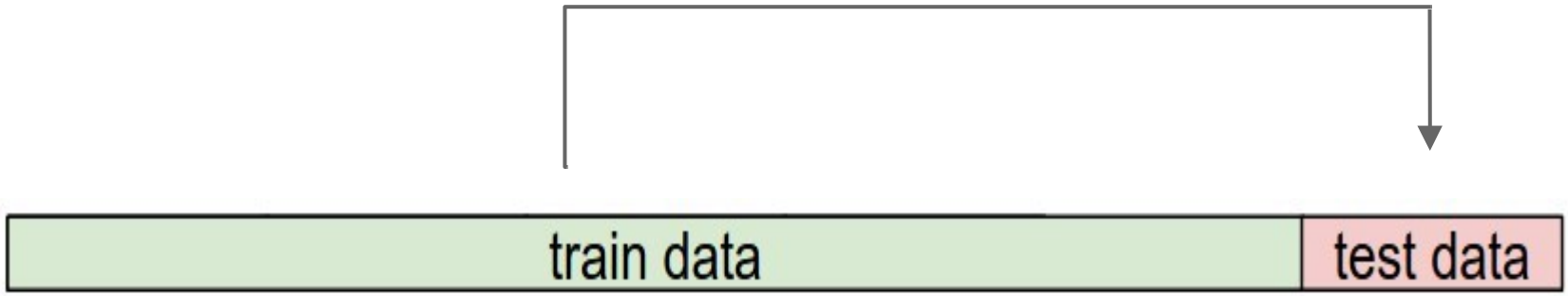
- http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf

Slide Credit:

Slide credit: D. Hoiem

Cross Validation

Try out what hyperparameters work best on test set.



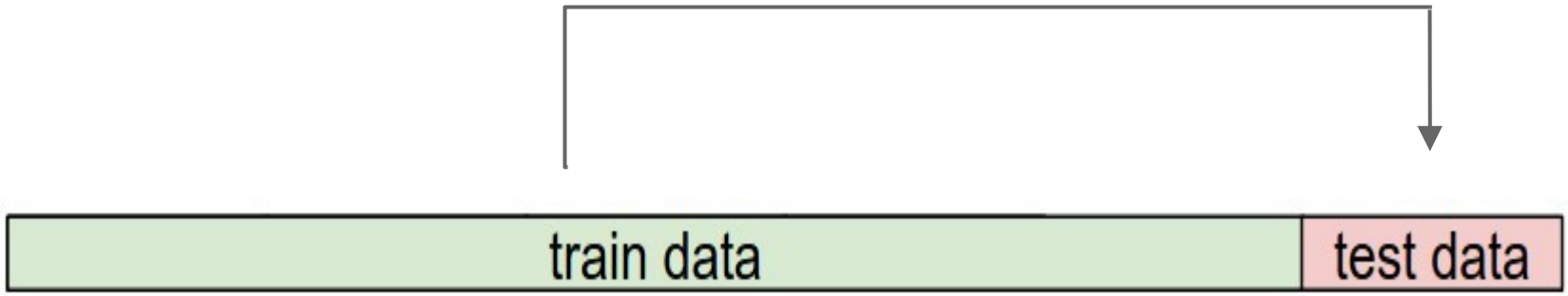
Slide Credit:

Cross Validation

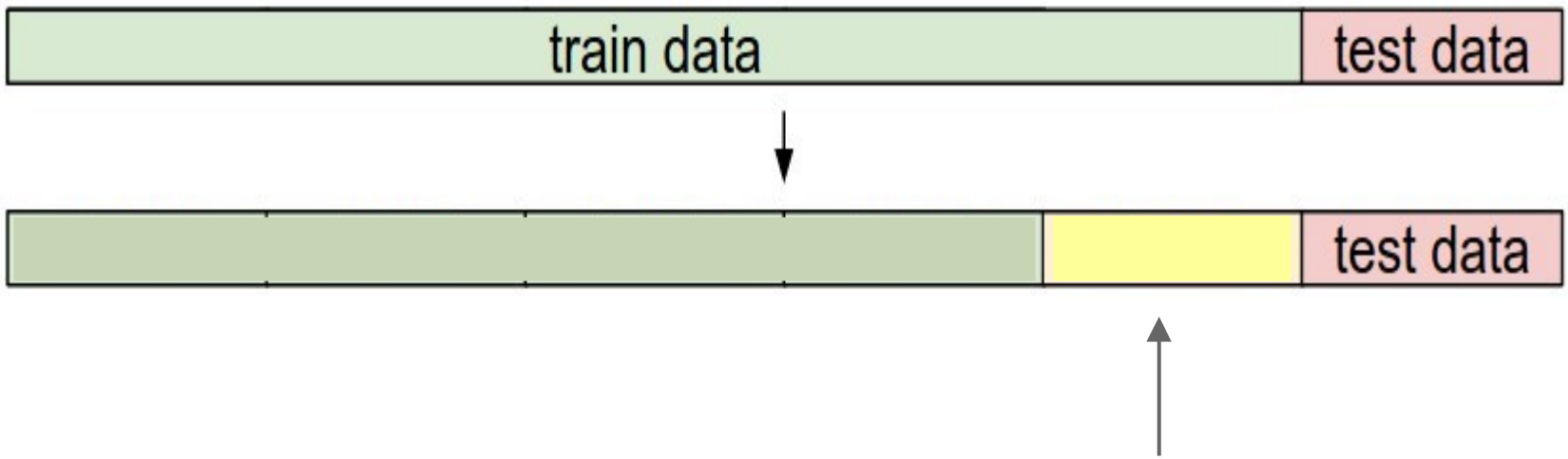
Trying out what hyperparameters work best on test set:

Very bad idea. The test set is a proxy for the generalization performance!

Use only **VERY SPARINGLY,** at the end.



Validation

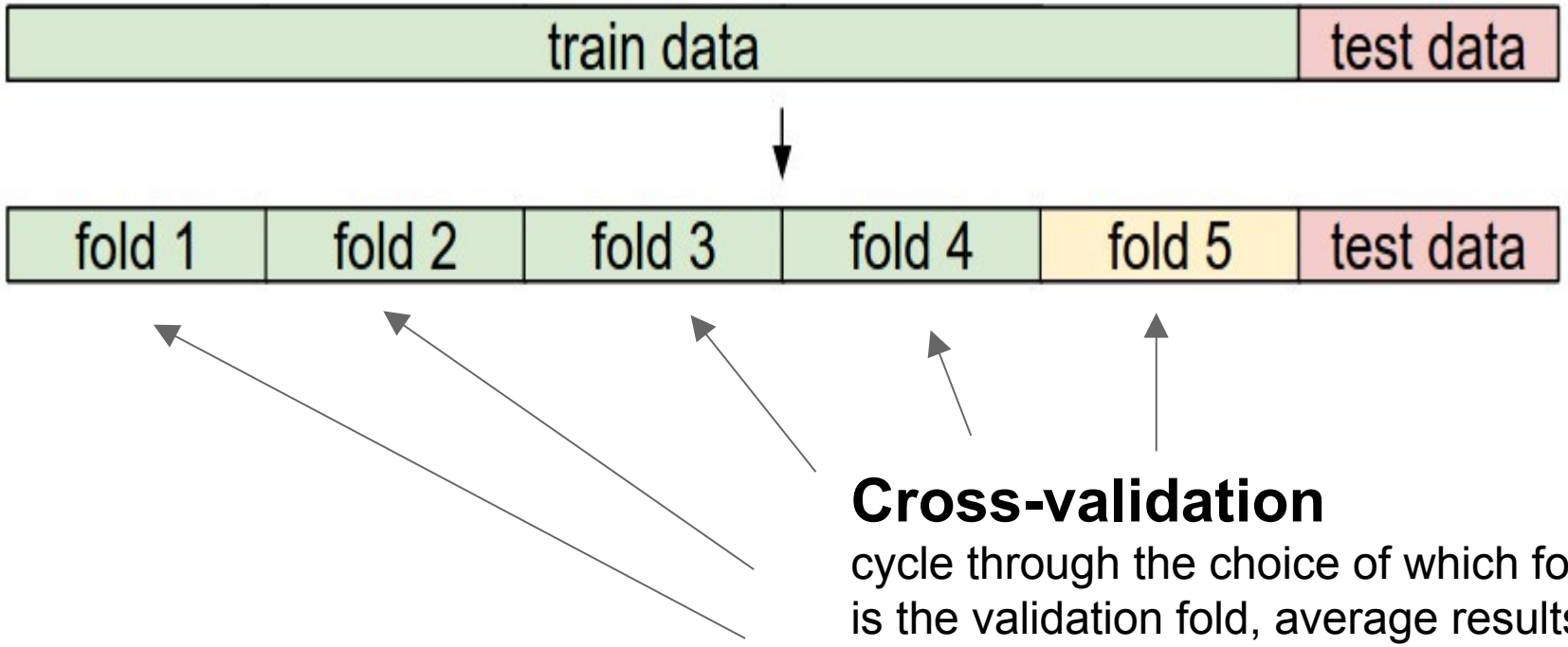


Validation data

use to tune hyperparameters

Slide Credit:

Cross Validation



Cross-validation

cycle through the choice of which fold is the validation fold, average results.

Slide Credit:

So…

- No classifier is inherently better than any other: you need to make assumptions to generalize

- Three kinds of error
  - Inherent: unavoidable
  - Bias: due to over-simplifications
  - Variance: due to inability to perfectly estimate parameters from limited data

Slide Credit: Sebastian D. Dolan

What to remember about classifiers

- Machine learning algorithms are tools, not dogmas
- Try simple classifiers first
- Better to have smart features and simple classifiers than simple features and smart classifiers
- Use increasingly powerful classifiers with more training data (bias-variance tradeoff)

Slide Credit: Slide credit: D. Hoiem

How to reduce variance?

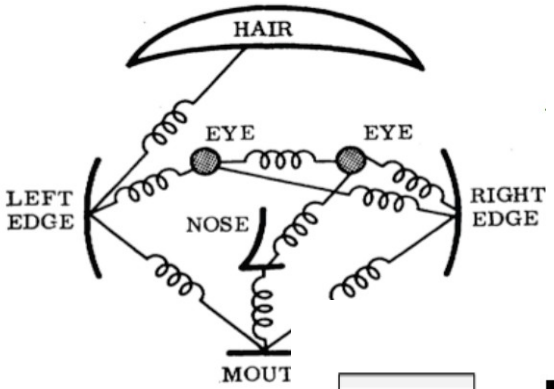
- Choose a simpler classifier
- Regularize the parameters
- Get more training data

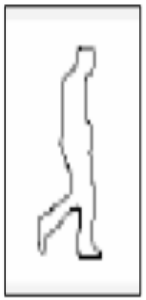
Slide Credit:

Slide credit: D. Hoiem

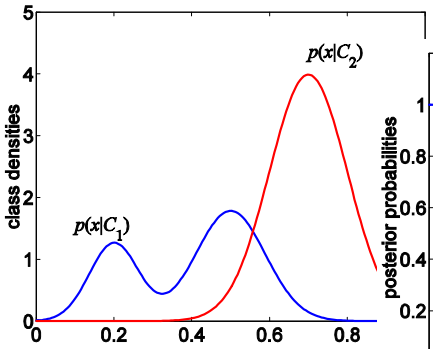
Review: Typical Components

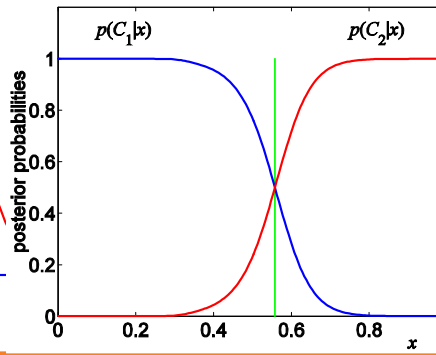
- **Hypothesis** generation
  - Sliding window, Segmentation, feature point detection, random, search
- **Encoding** of (local) image data
  - Colors, Edges, Corners, Histogram of Oriented Gradients, Wavelets, Convolution Filters
- **Relationships** of different parts to each other
  - Blur or histogram, Tree/Star, Pairwise/Covariance
- **Learning** from labeled examples
  - Selecting representative examples (templates), Clustering, Building a cascade
  - Classifiers: Bayes, Logistic regression, SVM, Decision Trees, AdaBoost, ...
  - Generative vs. Discriminative
- **Verification -** removing redundant, overlaping, incompatible examples
    - Non-Max Suppression, context priors, geometry





Exemplar Summary

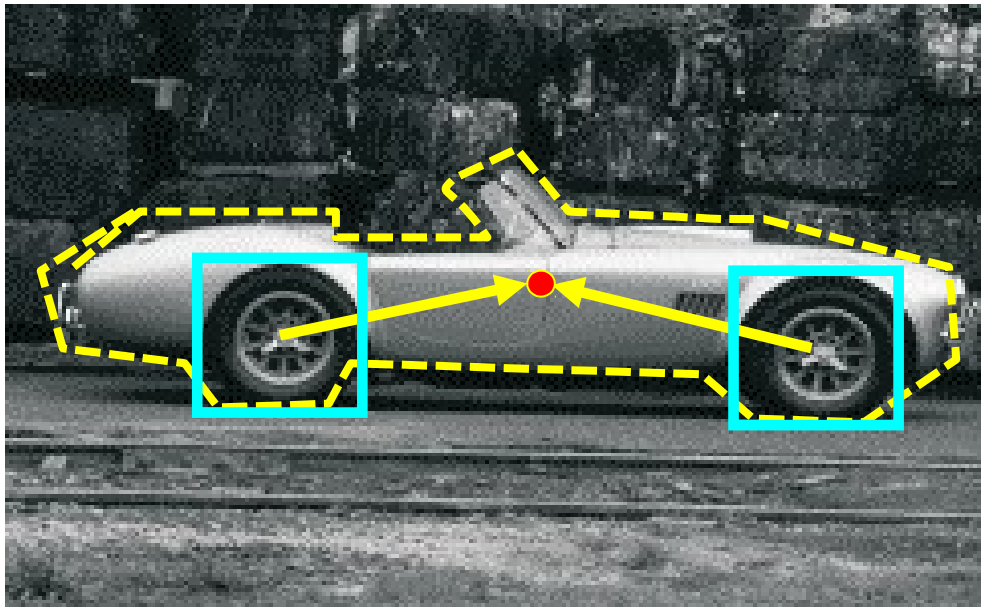




Slide Credit:

Implicit shape models

- Visual codebook is used to index votes for object position



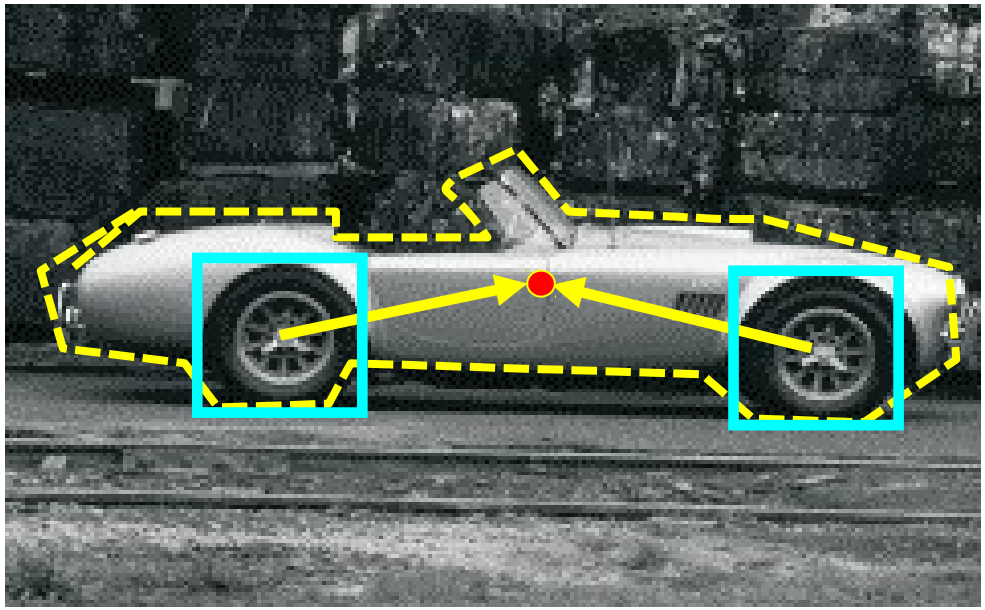
training image annotated with object localization info

visual codeword with displacement vectors

B. Leibe, A. Leonardis, and B. Schiele,
Combined Object Categorization and Segmentation with an Implicit Shape Model,
ECCV Workshop on Statistical Learning in Computer Vision 2004

Implicit shape models

- Visual codebook is used to index votes for object position



training image annotated with object localization info

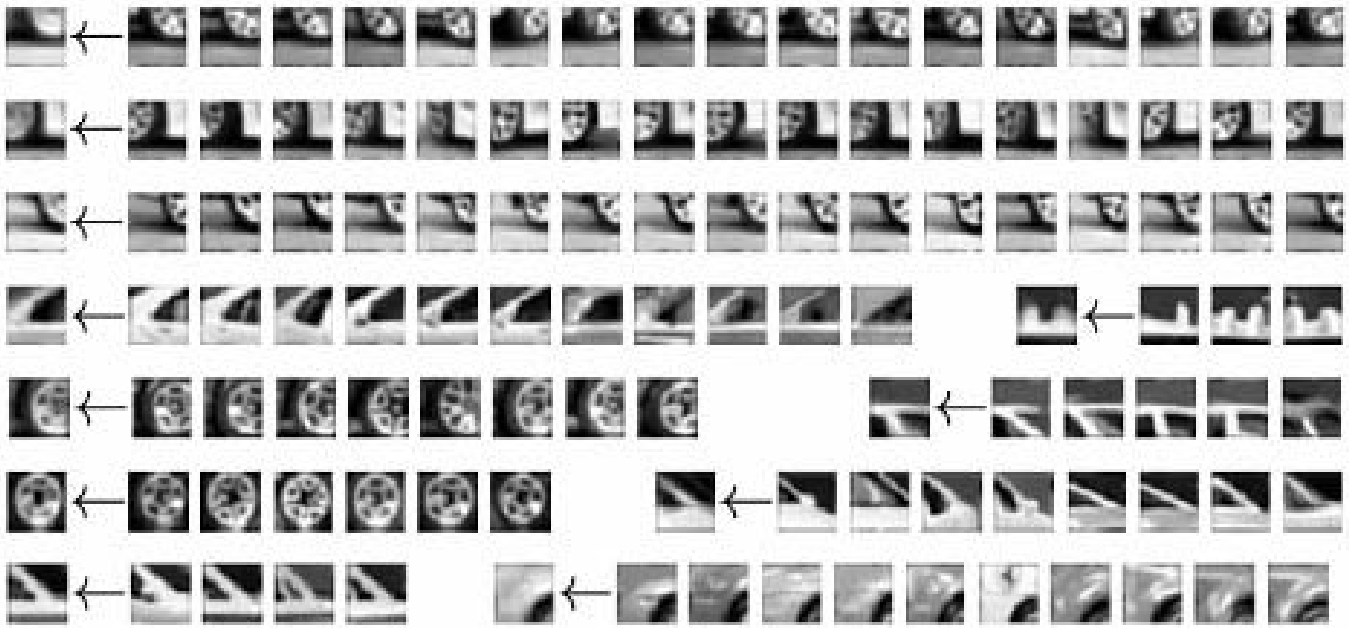
visual codeword with displacement vectors

B. Leibe, A. Leonardis, and B. Schiele,
Combined Object Categorization and Segmentation with an Implicit Shape Model,
ECCV Workshop on Statistical Learning in Computer Vision 2004

Slide Credit:

Implicit shape models: Training

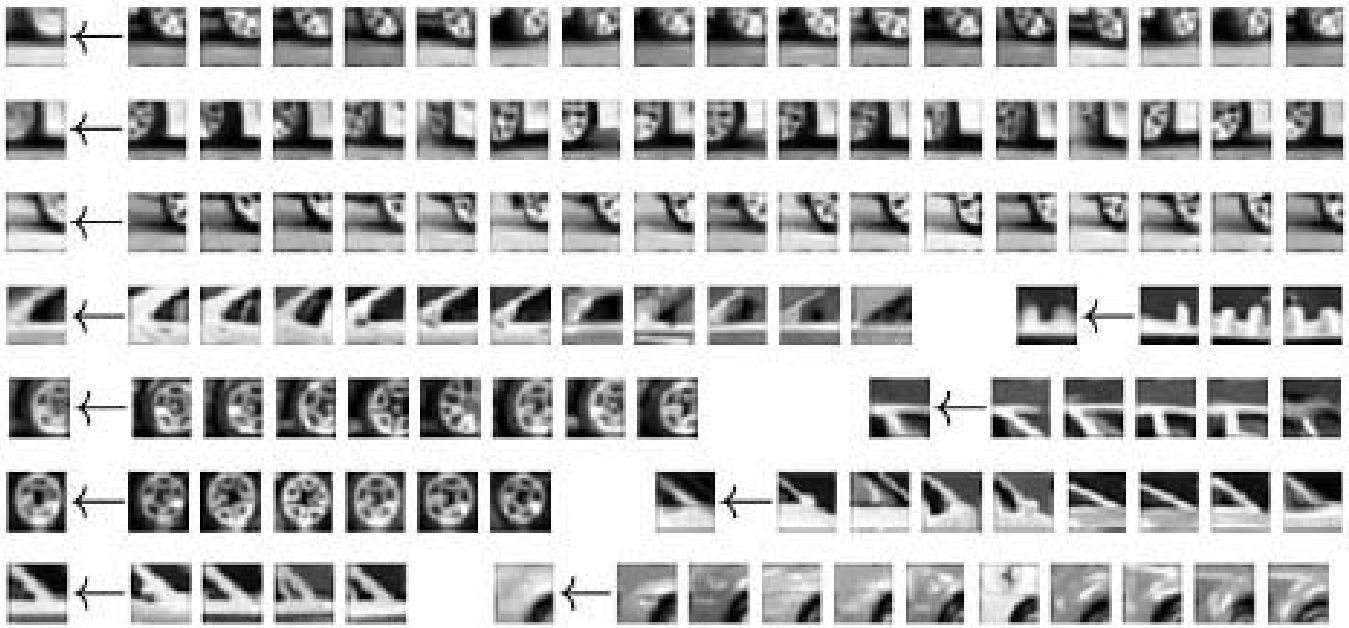
1. Build codebook of patches around extracted interest points using clustering



Slide Credit:

Implicit shape models: Training

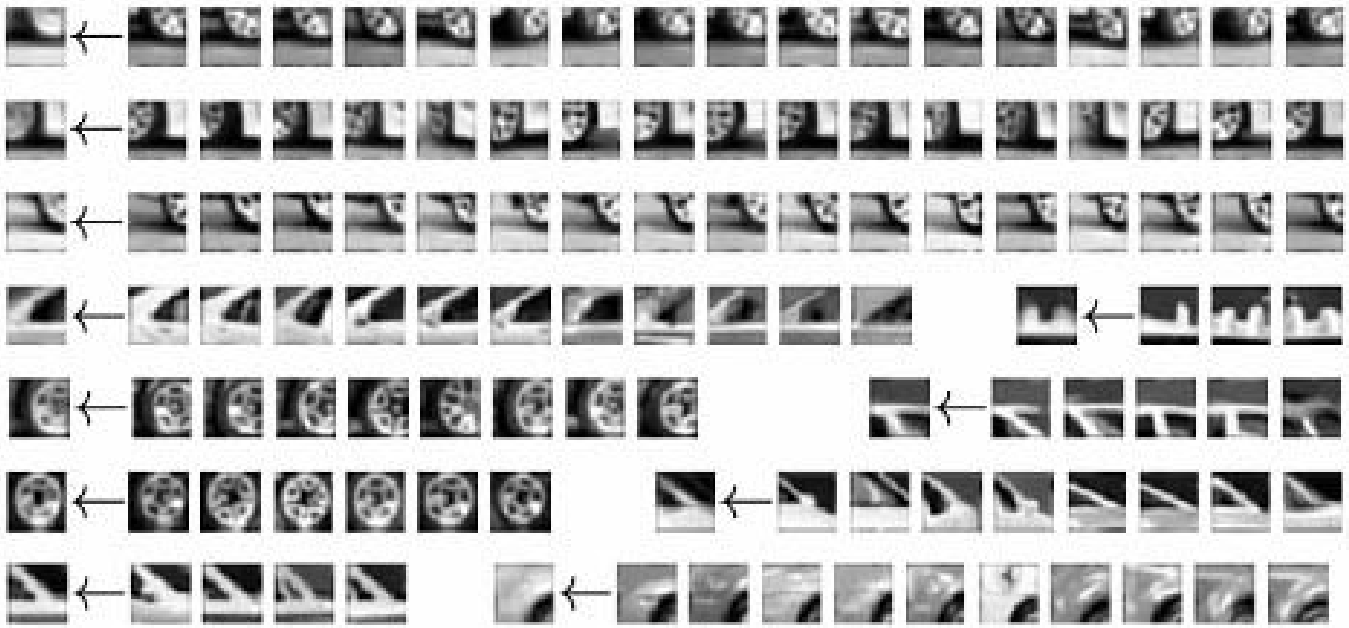
1. Build codebook of patches around extracted interest points using clustering



Slide Credit:

Implicit shape models: Training

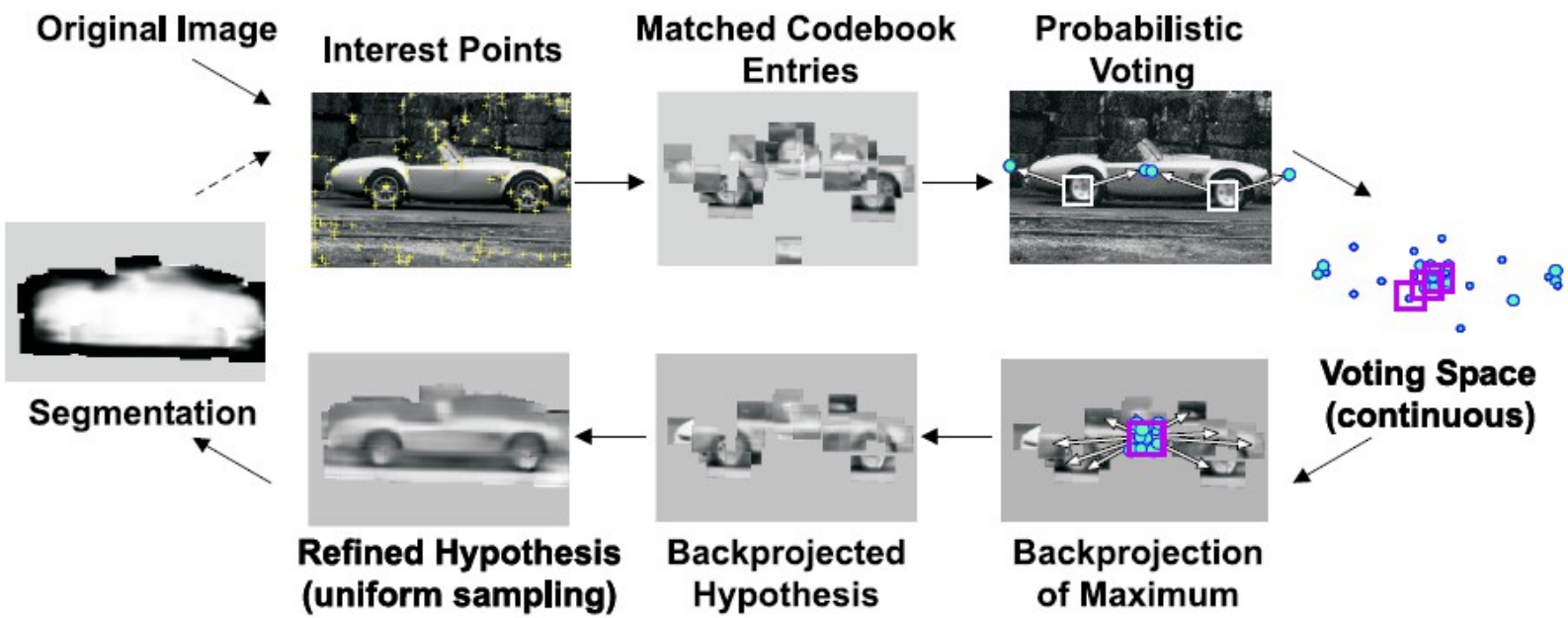
1. Build codebook of patches around extracted interest points using clustering



Slide Credit:

Implicit shape models: Testing

1. Given test image, extract patches, match to codebook entry
2. Cast votes for possible positions of object center
3. Search for maxima in voting space
4. Extract weighted segmentation mask based on stored masks for the codebook occurrences



Slide Credit:

Example: Results on Cows

Original image

Slide Credit: Source: B. Leibe

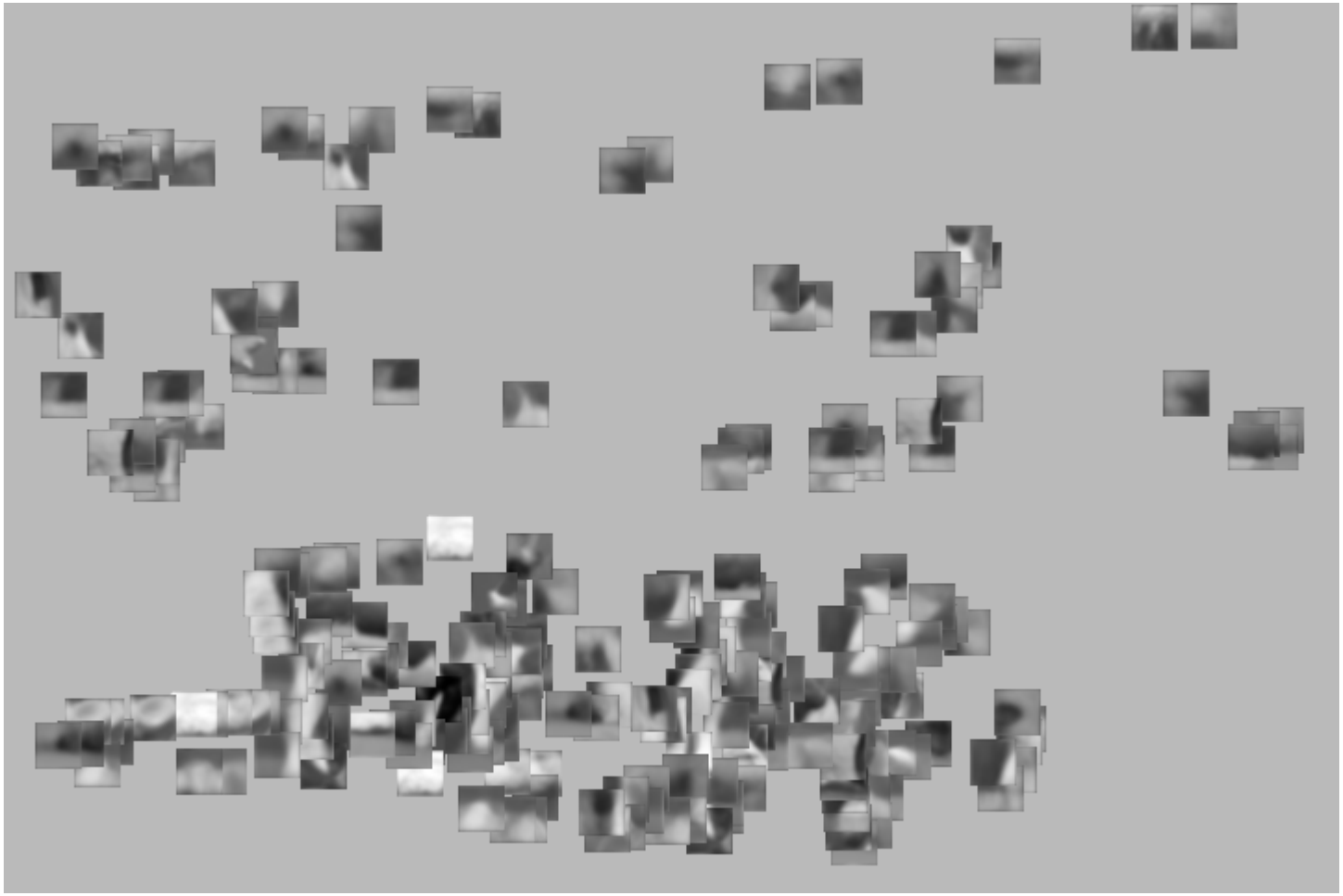
Example: Results on Cows

Interest points

Source: B. Leibe

Slide Credit:

Example: Results on Cows

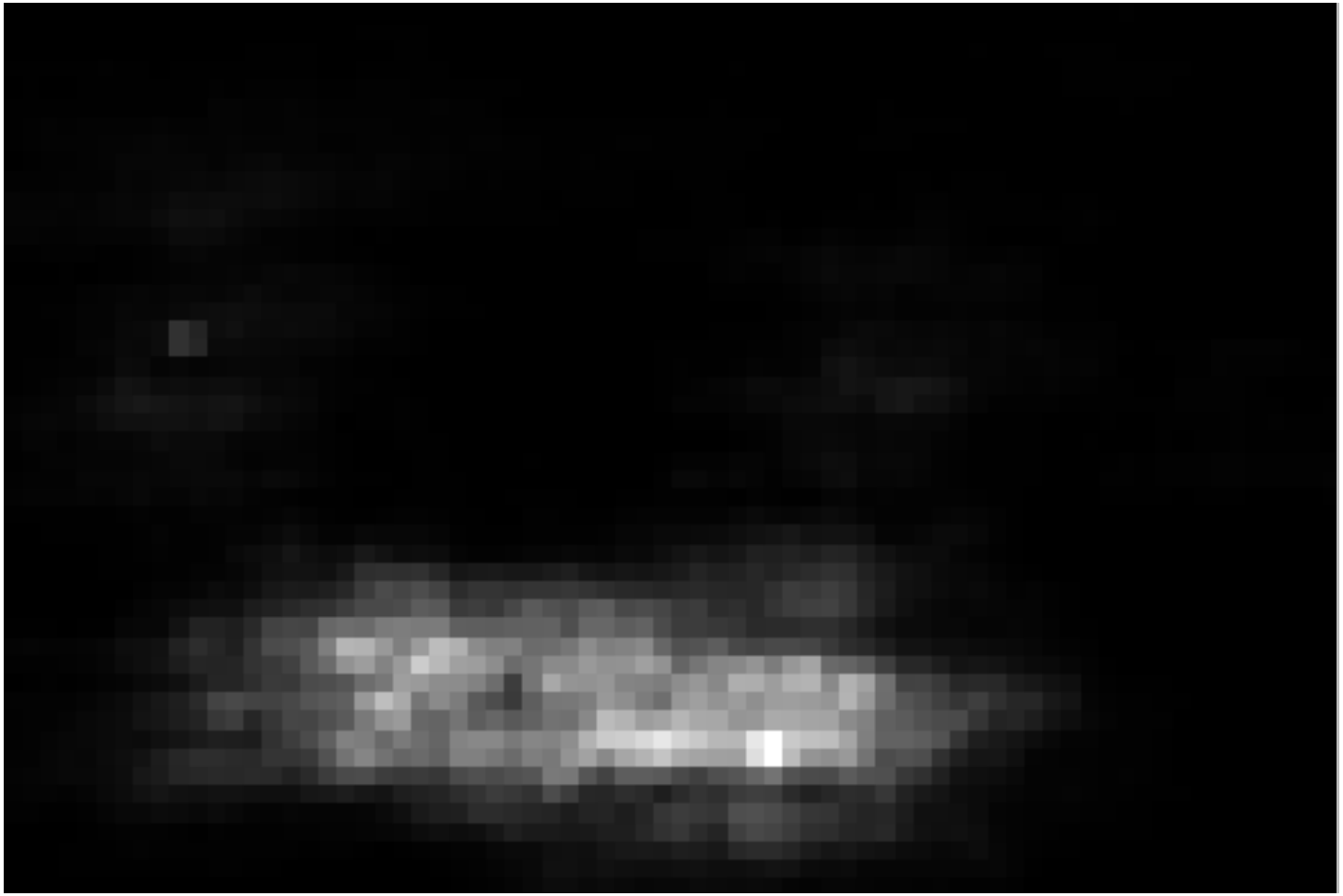


Matched patches

Source: B. Leibe

Slide Credit:

Example: Results on Cows



Probabilistic votes

Slide Credit: Source: B. Leibe

Example: Results on Cows

Hypothesis 1

Slide Credit: Source: B. Leibe

Example: Results on Cows

Hypothesis 1

Slide Credit: Source: B. Leibe

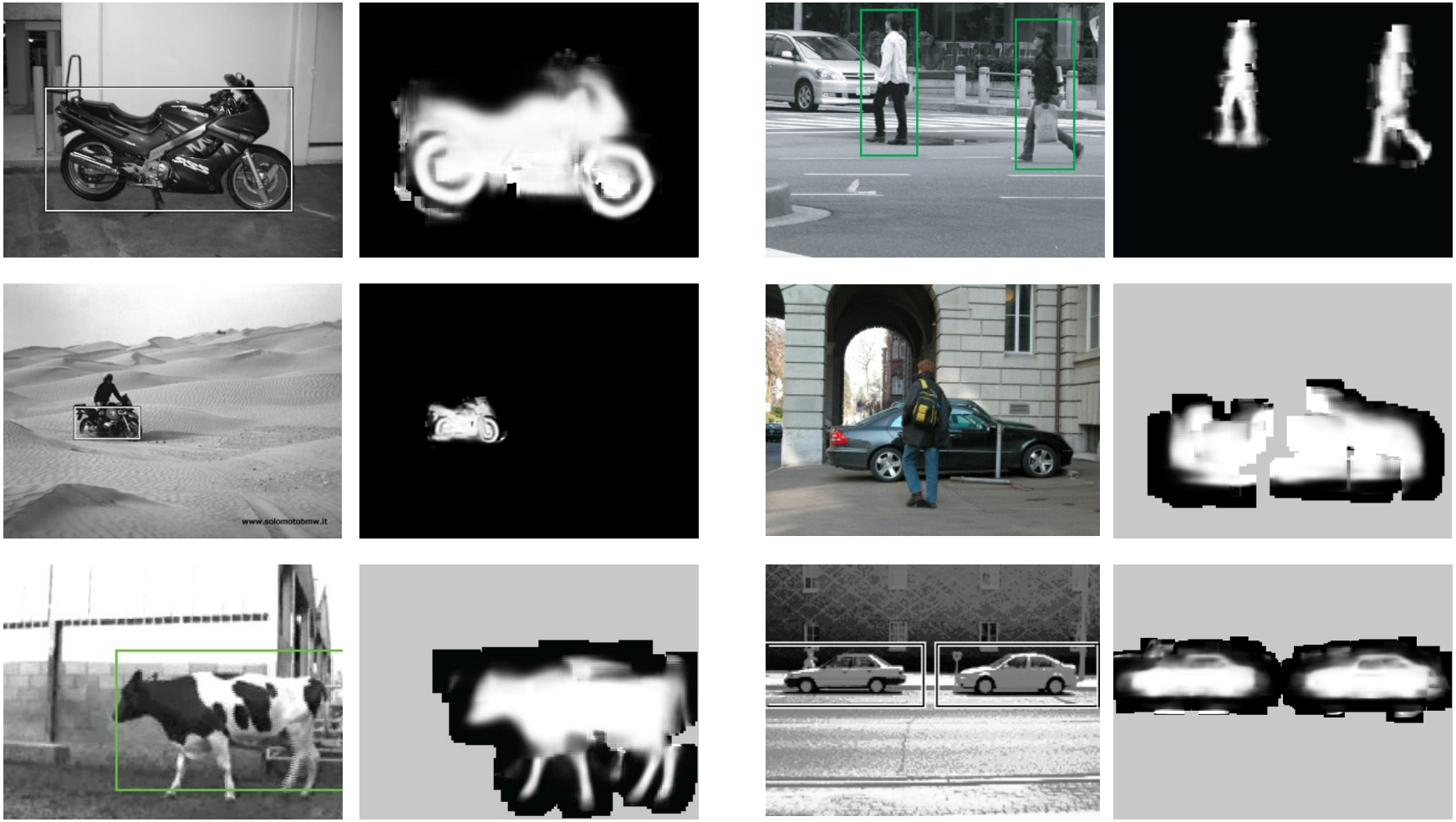
Example: Results on Cows

Hypothesis 3

Source: B. Leibe

Slide Credit:

Additional examples



B. Leibe, A. Leonardis, and B. Schiele,
Robust Object Detection with Interleaved Categorization and Segmentation, IJCV 77 (1-3), pp. 259-289, 2008.

Slide Credit:

Generative part-based models

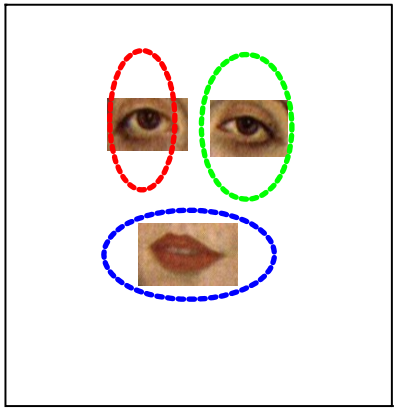
R. Fergus, P. Perona and A. Zisserman,
Object Class Recognition by Unsupervised Scale-Invariant Learning, CVPR 2003

Slide Credit: R. Fergus

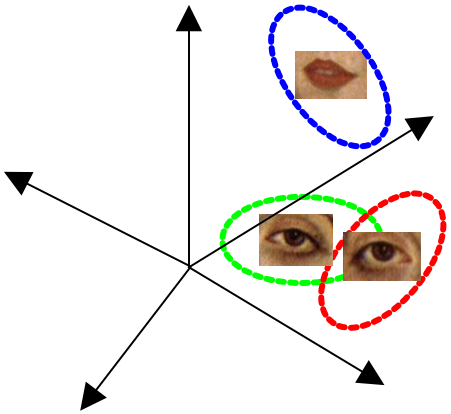
Generative probabilistic model

Foreground model

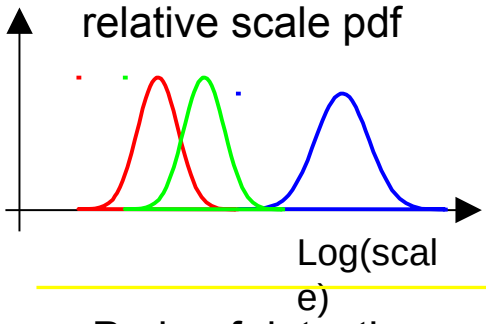
Gaussian shape pdf



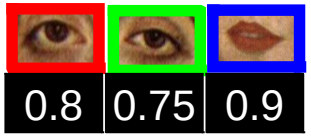
Gaussian part appearance pdf



Gaussian relative scale pdf

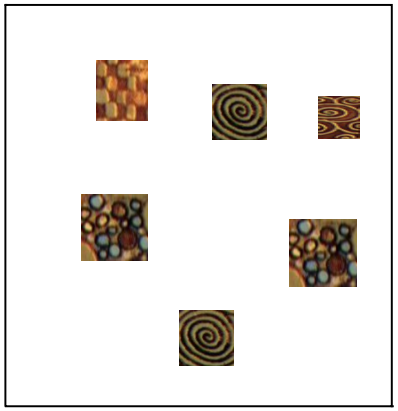


Prob. of detection

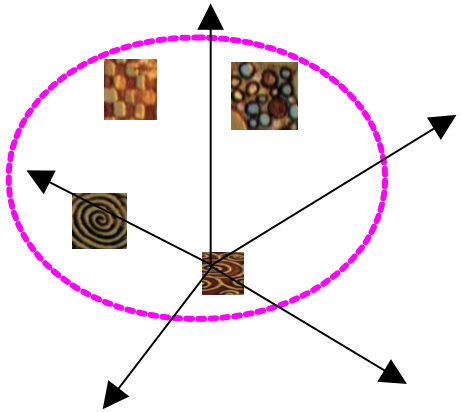


Clutter model

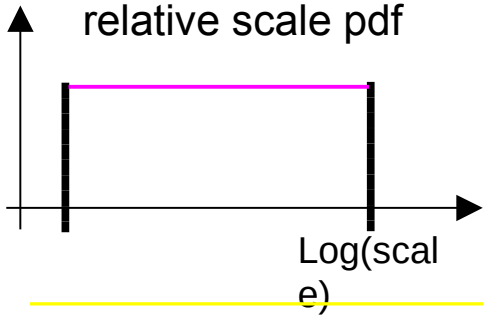
Uniform shape pdf



Gaussian appearance pdf



Uniform relative scale pdf



Poission pdf on # detections

Slide Credit: R. Fergus

Probabilistic model

$$P(image \mid object) = P(appearance, shape \mid object)$$

Part descriptors

Part locations

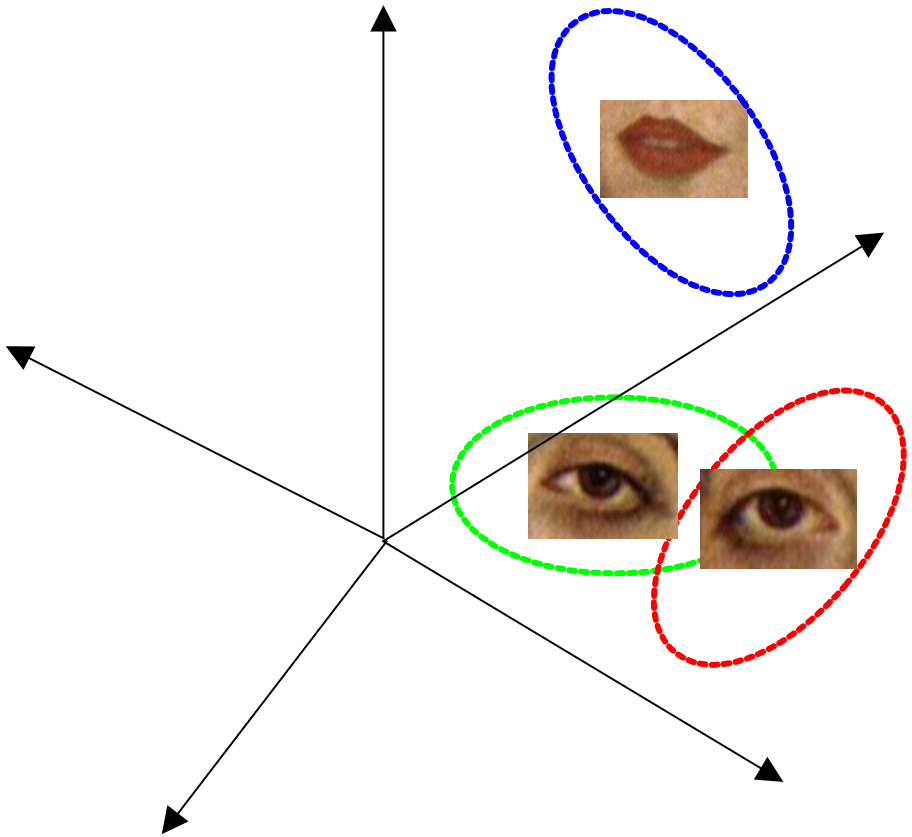
Candidate parts

Slide Credit: R. Fergus

Probabilistic model

$$P(image \mid object) = P(appearance, shape \mid object)$$

$$= \max_h \boxed{P(appearance \mid h, object)}\, p(shape \mid h, object)\, p(h \mid object)$$



Distribution over patch descriptors

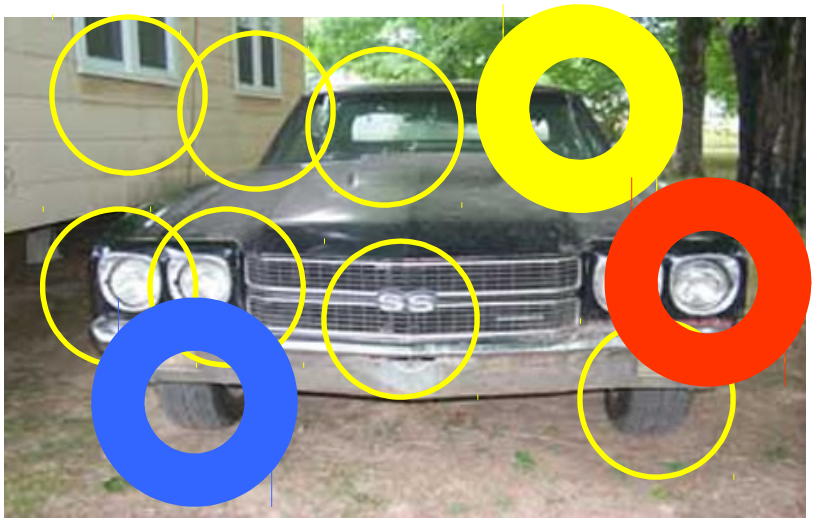
High-dimensional appearance space

Slide Credit: R. Fergus

Probabilistic model

$$P(image \mid object) = P(appearance, shape \mid object)$$
$$= \max_h P(appearance \mid h, object)\, p(shape \mid h, object)\, p(h \mid object)$$

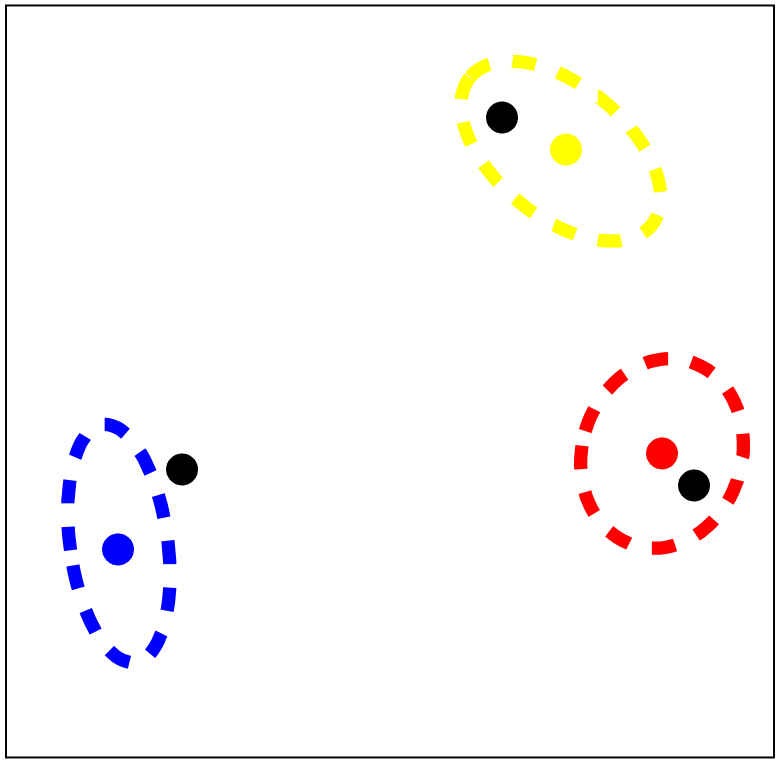
h: assignment of features to parts



Slide Credit: R. Fergus

Probabilistic model

$$P(image \mid object) = P(appearance, shape \mid object)$$
$$= \max_h P(appearance \mid h, object)\, p(shape \mid h, object)\, p(h \mid object)$$



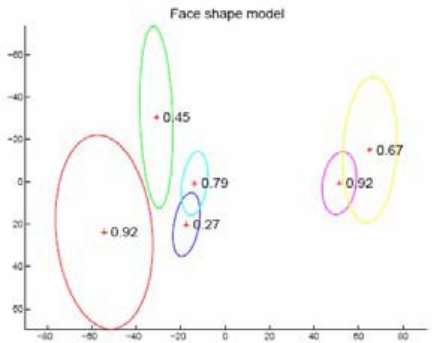
Distribution over joint part positions

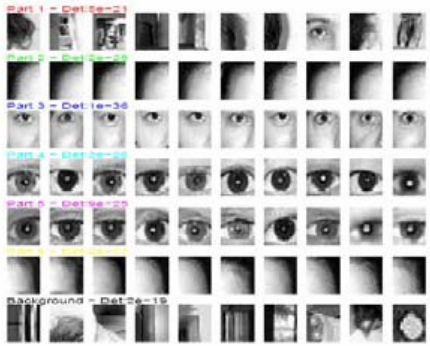
2D image space

Slide Credit: R. Fergus

Results: Faces

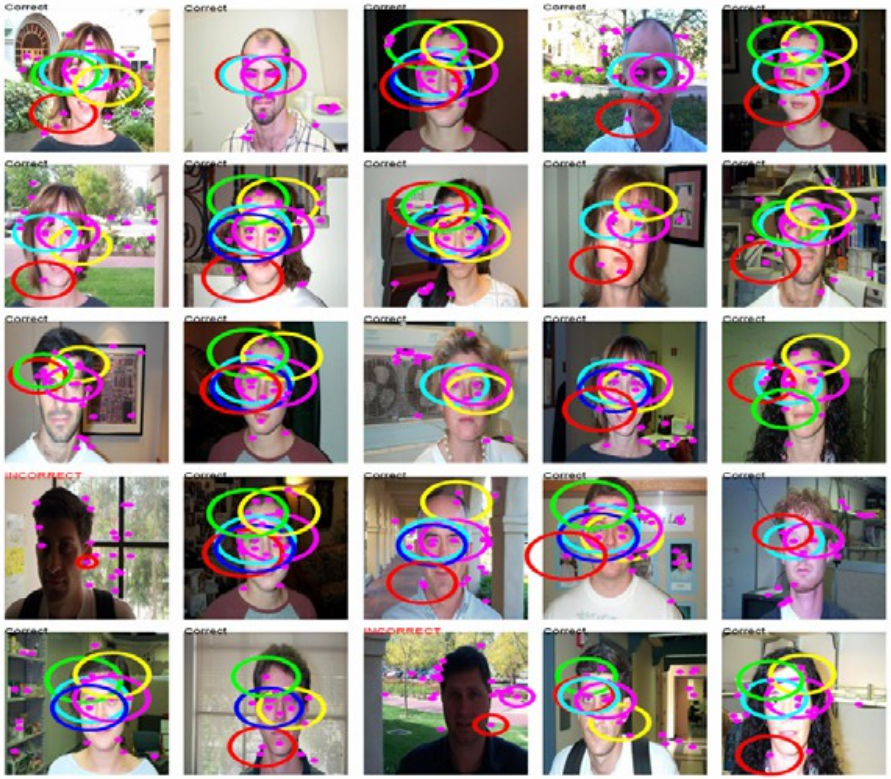
Face shape model





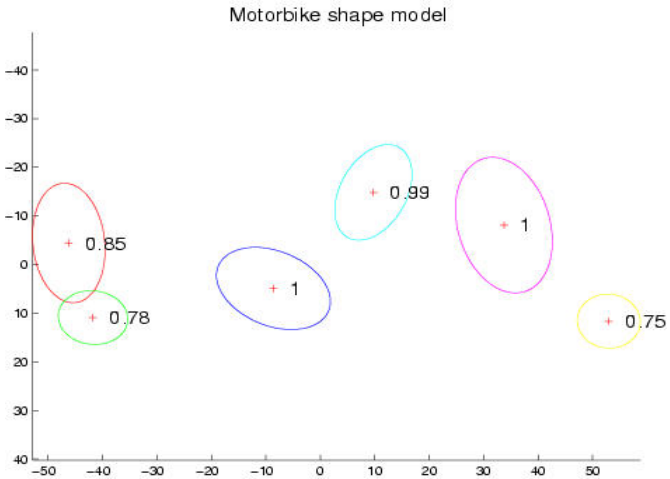
Patch appearance model

Recognition results



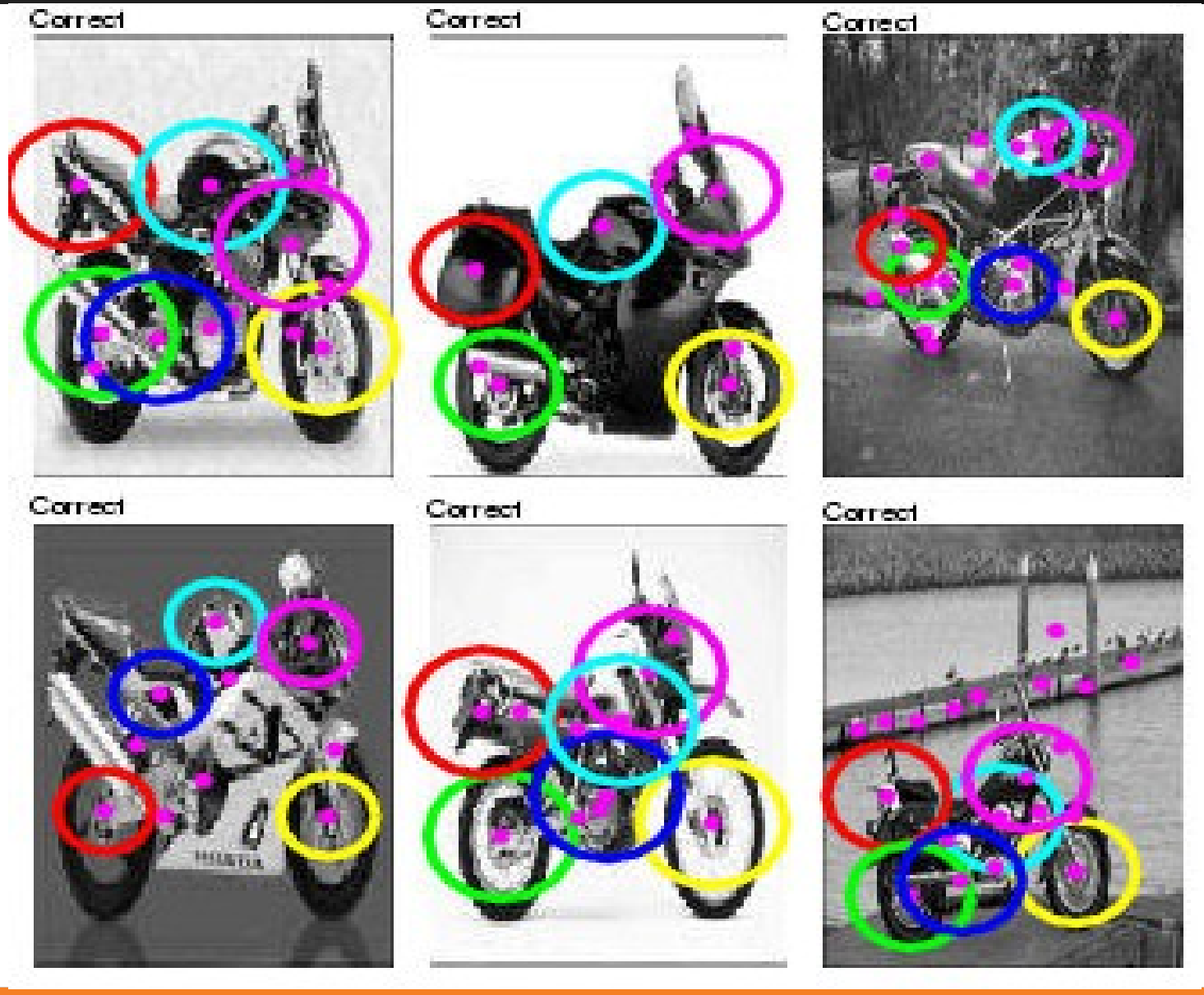
Slide Credit: R. Fergus

Results: Motorbikes



Slide Credit:

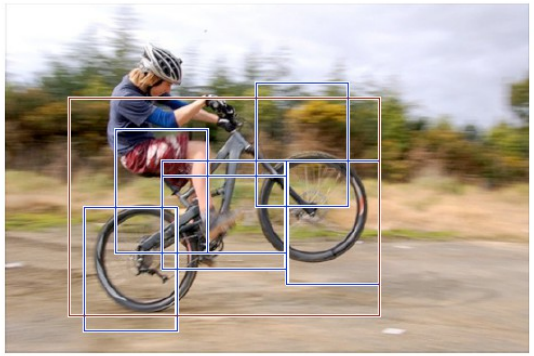
Motorbikes

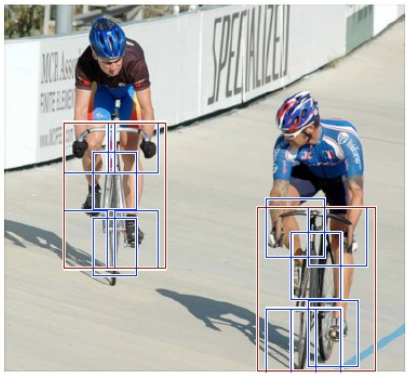


Slide Credit:

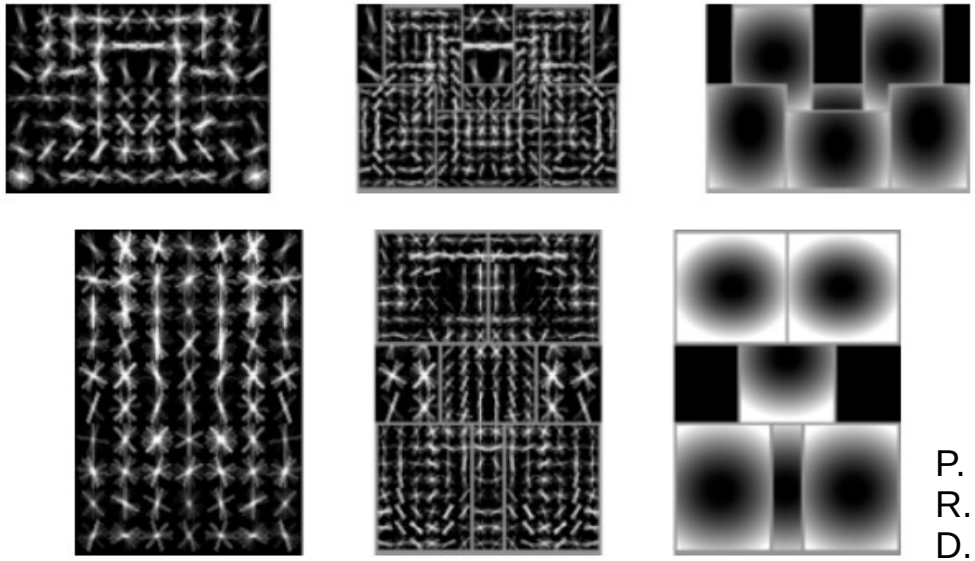
Deformable Parts Model

Detections





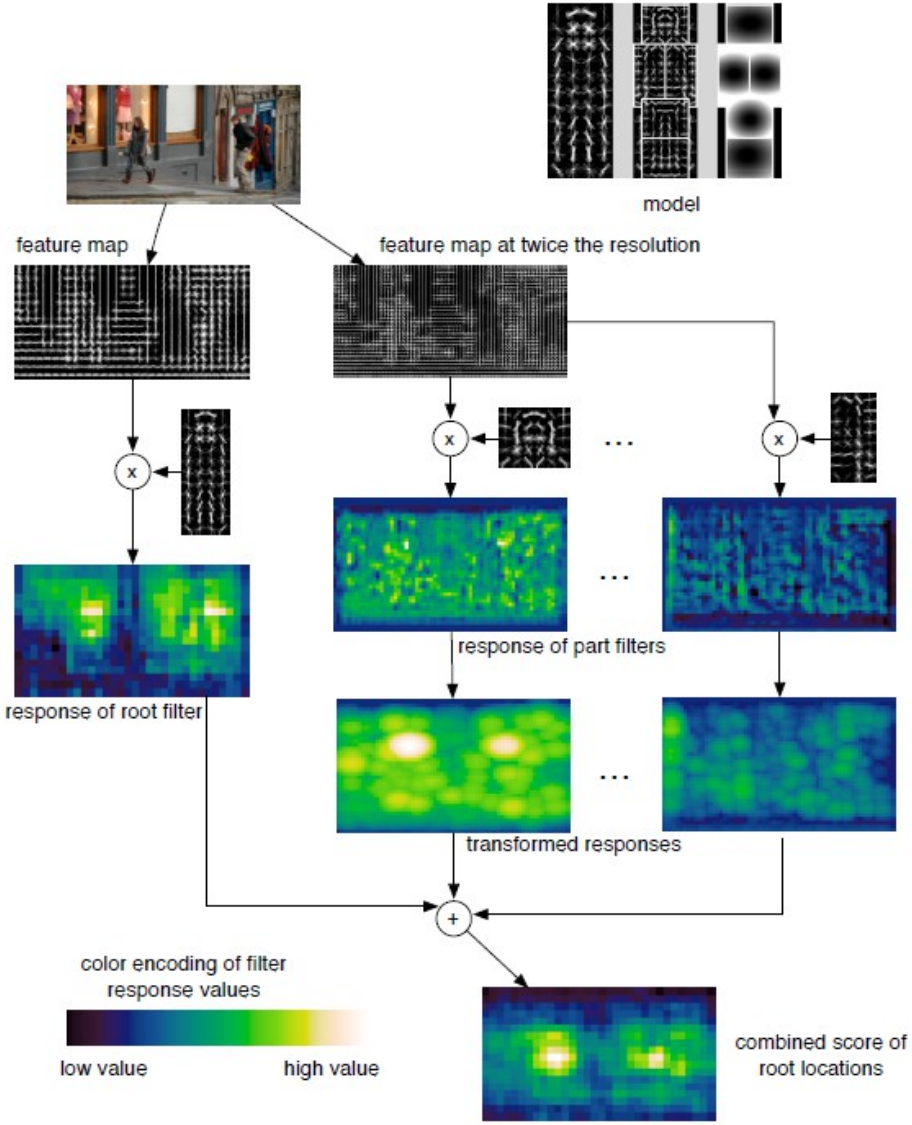
Template Visualization



root filters coarse resolution | part filters finer resolution | deformation models

P. Felzenszwalb,
R. Girshick,
D. McAllester,
D. Ramanan
"Object Detection with Discriminatively Trained Part Based Models"

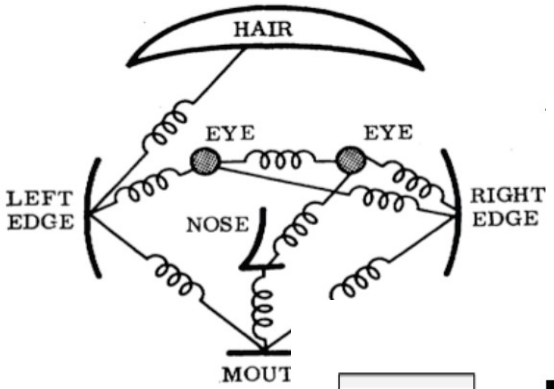
Slide Credit: Felzenszwalb

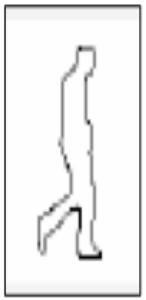


Slide Credit:

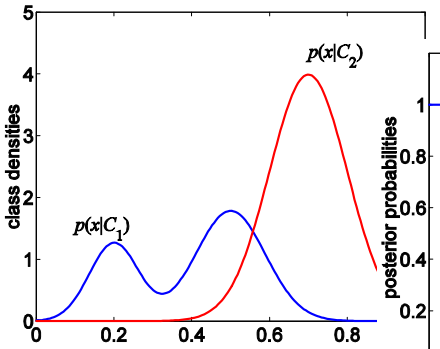
So many options... How to choose?

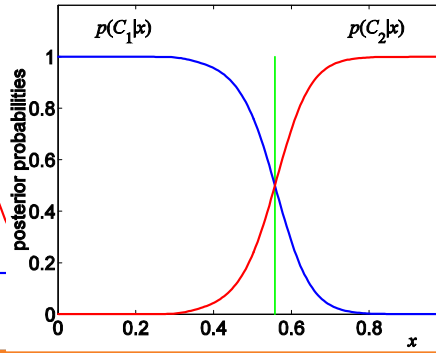
- **Hypothesis** generation
  - Sliding window, Segmentation, feature point detection, random, search
- **Encoding** of (local) image data
  - Colors, Edges, Corners, Histogram of Oriented Gradients, Wavelets, Convolution Filters
- **Relationship** of different parts to each other
  - Blur or histogram, Tree/Star, Pairwise/Covariance
- **Learning** from labeled examples
  - Selecting representative examples (templates), Clustering, Building a cascade
  - Classifiers: Bayes, Logistic regression, SVM, Decision Trees, AdaBoost, ...
  - Generative vs. Discriminative
- **Verification -** removing redundant, overlaping, incompatible examples
    - Non-Max Suppression, context priors, geometry





Exemplar Summary





Slide Credit: