

# Classifiers

D.A. Forsyth

# Classifiers

- Take a measurement  $x$ , predict a bit (yes/no; 1/-1; 1/0; etc)

# The big problems

- Image classification
  - eg this picture contains a parrot
- Object detection
  - eg in this box in the picture is a parrot

# Image classification - Strategy

- Make features
  - quite complex strategies, later
- Put in classifier

# Image classification - scenes



FIGURE 16.2: Some scenes are easily identified by humans. These are examples from the SUN dataset (Xiao *et al.* 2010) of scene categories that people identify accurately from images; the label above each image gives its scene type. *This figure was originally published as Figure 2 of “SUN database: Large-scale Scene Recognition from Abbey to Zoo,” by J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, Proc. IEEE CVPR 2010, © IEEE, 2010.*

# Image classification - material



FIGURE 16.1: Material is not the same as object category (the three cars on the top are each made of different materials), and is not the same as texture (the three checkered objects on the bottom are made of different materials). Knowing the material that makes up an object gives us a useful description, somewhat distinct from its identity and its texture. *This figure was originally published as Figures 2 and 3 of "Exploring Features in a Bayesian Framework for Material Recognition," by C. Liu, L. Sharan, E. Adelson, and R. Rosenholtz Proc. CVPR 2010, 2010 © IEEE, 2010.*

# Image classification - offensive

whale - Google Search 6/11/12 12:43


+You Search Images Maps Play YouTube News Gmail Documents Calendar More ▾

whale

**Search** About 148,000,000 results (0.06 seconds)

Web  
Images  
Maps  
Videos  
News  
Shopping  
More

Related searches: [killer whale](#) [humpback whale](#) [blue whale](#) [fin whale](#) [cartoon whale](#) [whales underwater](#)



**Any time**  
Past 24 hours  
Past week  
Custom range...

**All results**  
By subject

**Any size**  
Large  
Medium  
Icon  
Larger than...

# Detection with a classifier

- Search
  - all windows
  - at relevant scales
- Prepare features
- Classify
  
- Issues
  - how to get only one response
  - speed
  - accuracy



# Classifiers

- Take a measurement  $x$ , predict a bit (yes/no; 1/-1; 1/0; etc)
- Strategies:
  - non-parametric
    - nearest neighbor
  - probabilistic
    - histogram
    - logistic regression
  - decision boundary
    - SVM

# Basic ideas in classifiers

- Loss

- errors have a cost, and different types of error have different costs
  - this means each classifier has an associated risk
- Total risk

$$R(s) = Pr\{1 \rightarrow 2 | \text{using } s\} L(1 \rightarrow 2) + Pr\{2 \rightarrow 1 | \text{using } s\} L(2 \rightarrow 1)$$

- Bayes risk
  - smallest possible value of risk, over all classification strategies

# Nearest neighbor classification

- Examples
  - $(x_i, y_i)$ 
    - here  $y$  is yes/no or -1/1 or 1/0 or...
  - training set
- Strategy
  - to label new example (test example)
    - find closest training example
    - report its label
- Advantage
  - in limit of very large number of training examples, risk is  $2 \times$  bayes risk
- Issue
  - how do we find closest example?
  - what distance should we use?

# k-nearest neighbors

- Strategy
  - to classify test example
    - find k-nearest neighbors of test point
    - vote (it's a good idea to have k odd)
- Issues
  - how do we find nearest neighbors?
  - what distance should we use?

# Nearest neighbors

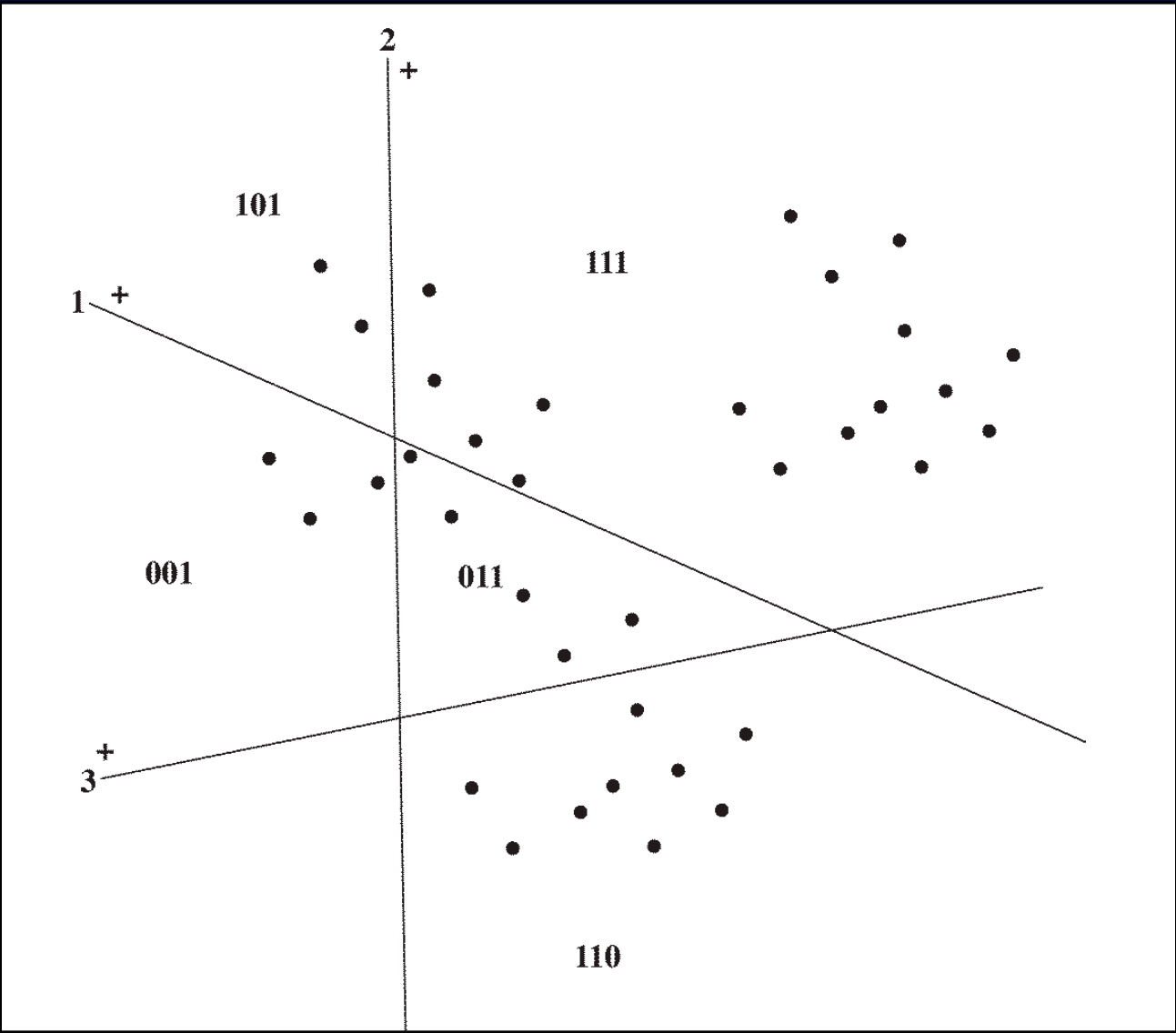
- Exact nearest neighbor in large dataset
  - linear search is very good
  - very hard to do better (surprising fact)
- Approximate nearest neighbor is easier
  - methods typically give probabilistic guarantees
  - good enough for our purposes
  - methods
    - locality sensitive hashing
    - k-d tree with best bin first

# Locality sensitive hashing (LSH)

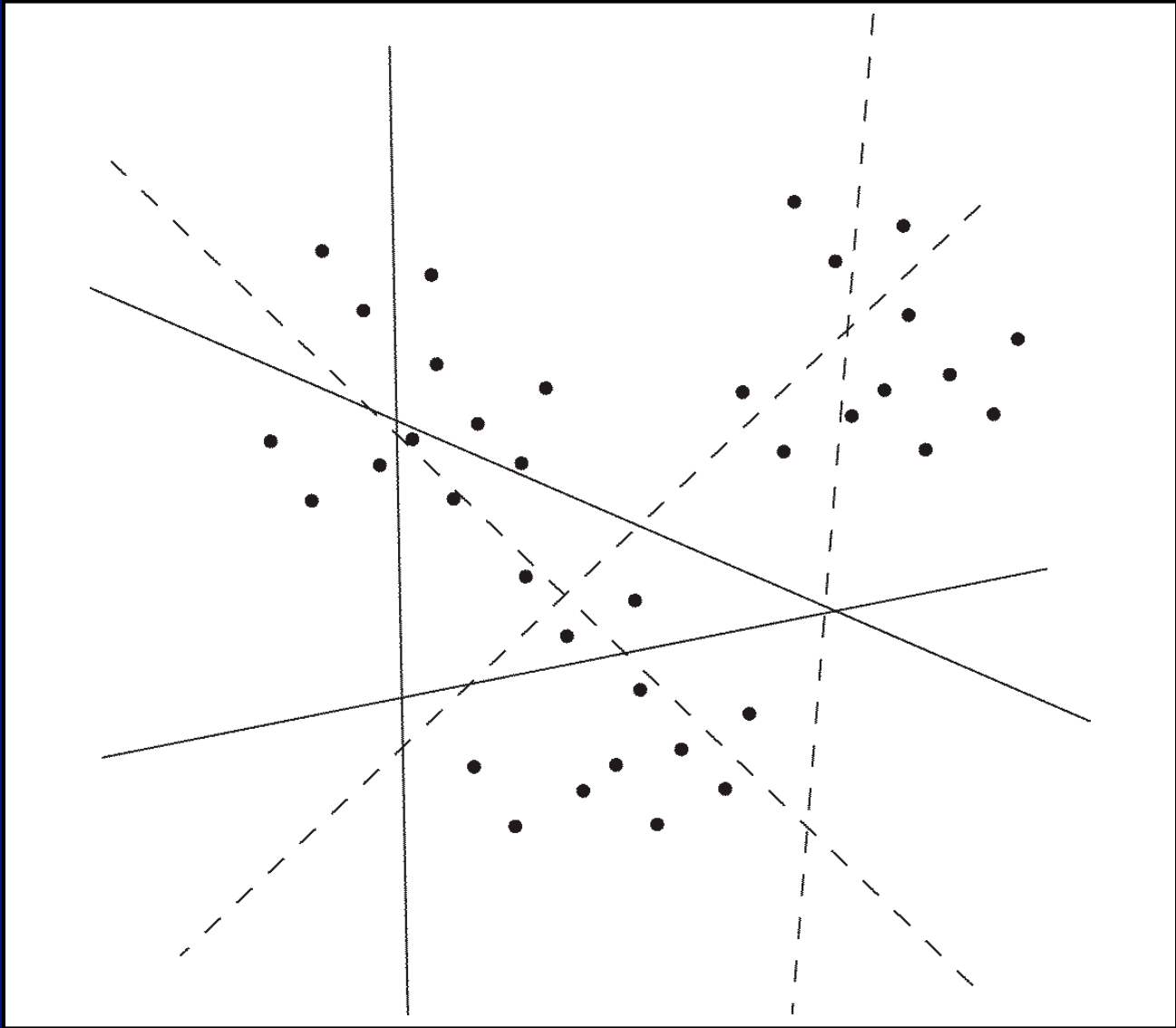
- Build a set of hash tables
- Insert each training data at its hash key
- ANN
  - compute key for test point
  - recover all points in each hash table at that key
  - linear search for distance in these points
  - take the nearest

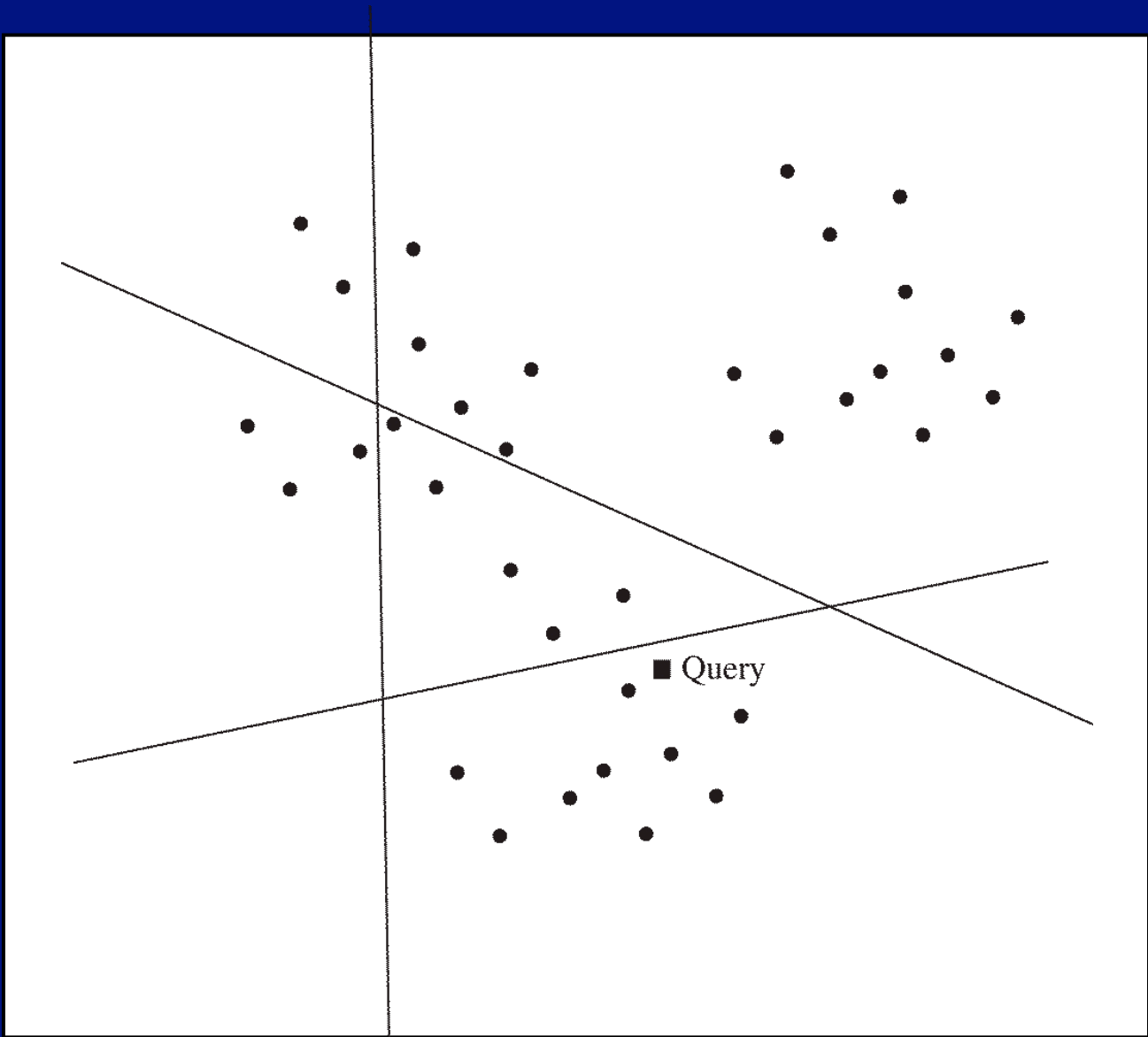
# Hash functions

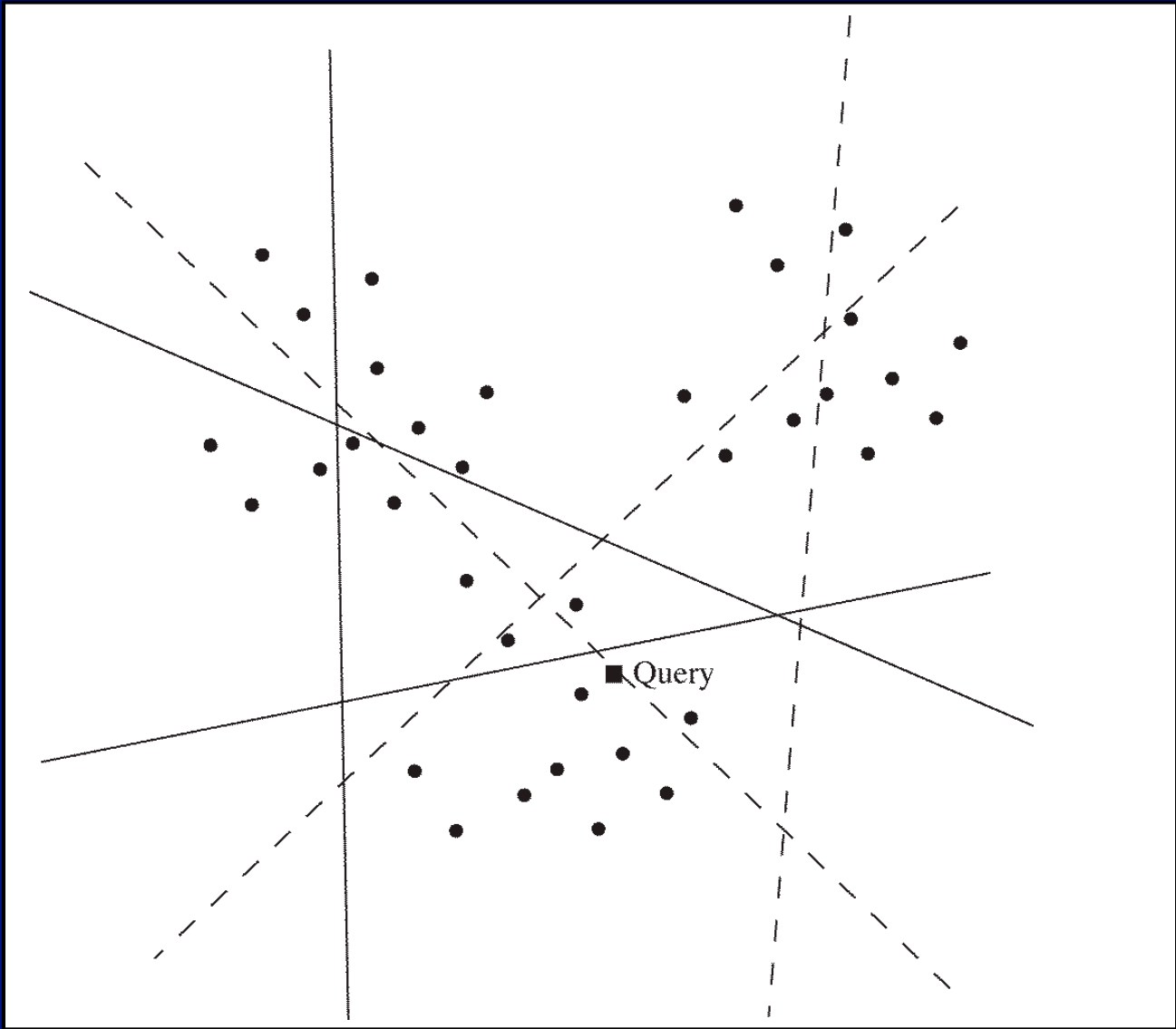
- Random splits
  - for each bit in the key, choose random  $w, b$
  - bit is:  $\text{sign}(w*x+b)$











# LSH - issues

- Parameters
  - How many hash tables?
  - How many bits in key?
- Issues
  - quite good when data is spread out
  - can be weak when it is clumpy
    - too many points in some buckets, too few in others

# kd-trees (outline)

- Build a kd-tree, splitting on median
- Walk the tree
  - find leaf in which query point lies
  - backtrack, pruning branches that are further away than best point so far

# kd-Trees

- Standard construction fails in high dimensions
  - too much backtracking
- Good approximate nearest neighbor, if we
  - probe only a fixed number of leaves
  - use best bin first heuristic
- Very good for clumpy data

# Approximate nearest neighbors

- In practice
  - fastest method depends on dataset
  - parameters depend on dataset
  - search methods, parameters using dataset
  - FLANN (<http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN>)
    - can do this search

# Basic ideas in classifiers

- Loss
  - errors have a cost, and different types of error have different costs
    - this means each classifier has an associated risk
  - Total risk

$$R(s) = Pr\{1 \rightarrow 2 | \text{using } s\} L(1 \rightarrow 2) + Pr\{2 \rightarrow 1 | \text{using } s\} L(2 \rightarrow 1)$$

- Expected loss of classifying a point gives

1 if  $p(1|\mathbf{x})L(1 \rightarrow 2) > p(2|\mathbf{x})L(2 \rightarrow 1)$

2 if  $p(1|\mathbf{x})L(1 \rightarrow 2) < p(2|\mathbf{x})L(2 \rightarrow 1)$



# Histogram based classifiers

- Represent class-conditional densities with histogram
- Advantage:
  - estimates become quite good
    - (with enough data!)
- Disadvantage:
  - Histogram becomes big with high dimension
    - but maybe we can assume feature independence?

# Finding skin

- Skin has a very small range of (intensity independent) colours, and little texture
  - Compute an intensity-independent colour measure, check if colour is in this range, check if there is little texture (median filter)
  - See this as a classifier - we can set up the tests by hand, or learn them.

# Histogram classifier for skin

$$\frac{P(rgb | skin)}{P(rgb | \neg skin)} \geq \Theta$$

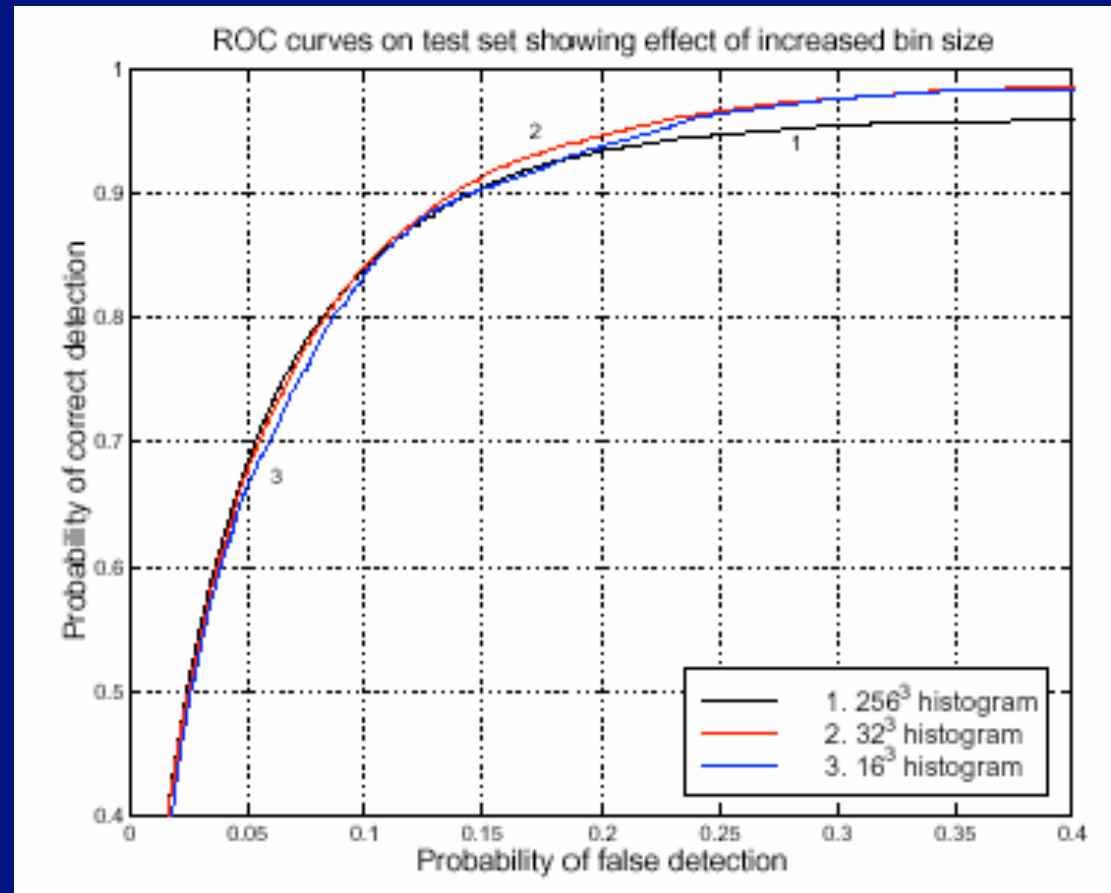


Figure from Jones+Rehg, 2002

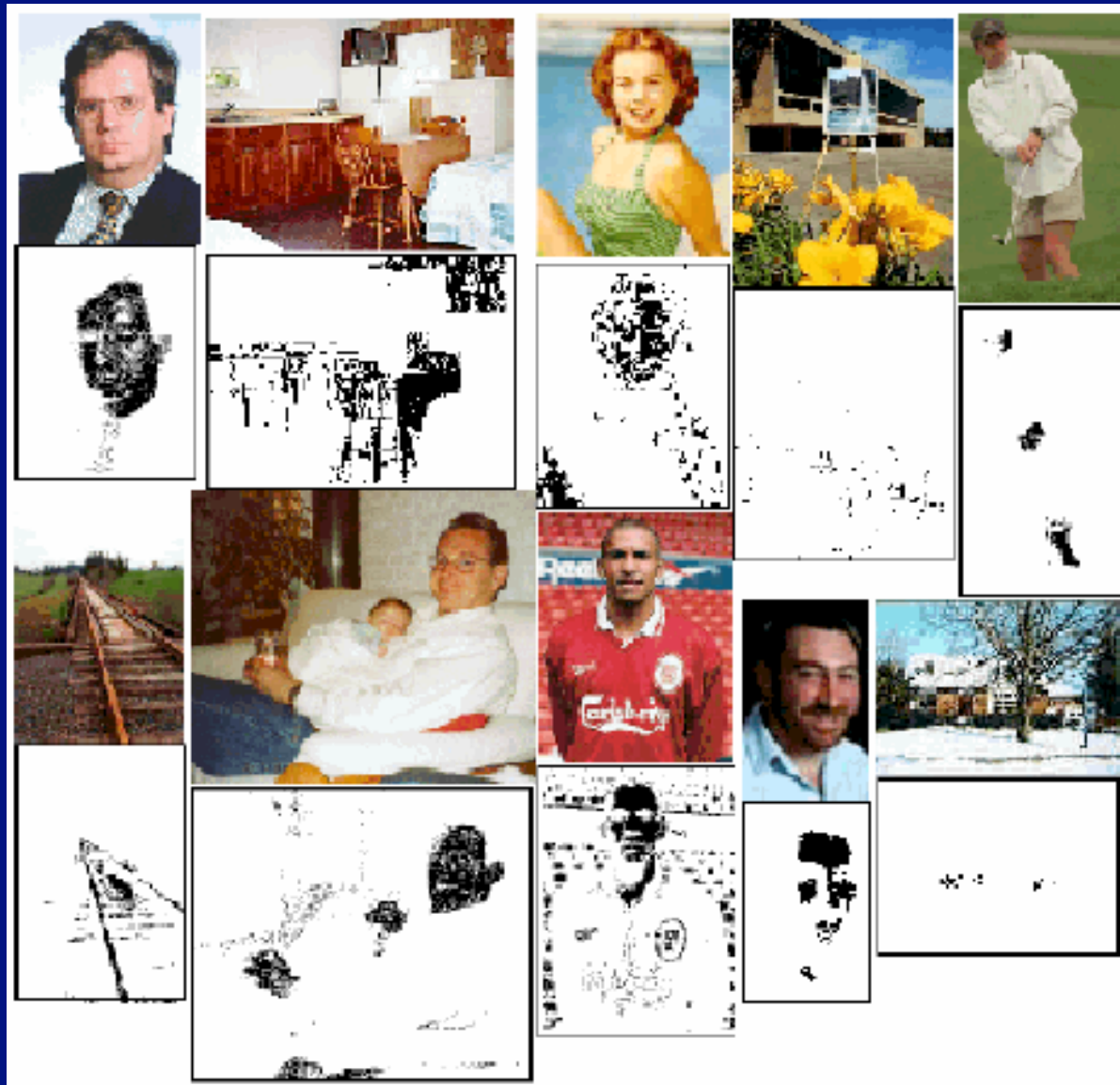


Figure from Jones+Rehg, 2002

# Curse of dimension - I

- This won't work for many features
  - try R, G, B, and some texture features
  - too many histogram buckets

# Naive Bayes

- Previously, we detected with a likelihood ratio test

$$\frac{P(\text{features}|\text{event})}{P(\text{features}|\text{not event})} > \text{threshold}$$

- Now assume that features are conditionally independent given event

$$P(f_0, f_1, f_2, \dots, f_n|\text{event}) = P(f_0|\text{event})P(f_1|\text{event})P(f_2|\text{event}) \dots P(f_n|\text{event})$$

# Naive Bayes

- (not necessarily perjorative)
- Histogram doesn't work when there are too many features
  - the curse of dimension, first version
  - assume they're independent conditioned on the class, cross fingers
  - reduction in degrees of freedom
  - very effective for face finders
    - relations may not be all that important
  - very effective for high dimensional problems
    - bias vs. variance

# Logistic Regression

- Build a parametric model of the posterior,
  - $p(\text{class}|\text{information})$
- For a 2-class problem, assume that
  - $\log(P(1|\text{data})) - \log(P(0|\text{data})) = \text{linear expression in data}$
- Training
  - maximum likelihood on examples
  - problem is convex
- Classifier boundary
  - linear



# Logistic regression

*Logistic regression* is a classifier that gives a good, simple example of why regularization should be helpful. In logistic regression, we model the class-conditional densities by requiring that

$$\log \frac{p(1|\mathbf{x})}{p(-1|\mathbf{x})} = \mathbf{a}^T \mathbf{x}$$

where  $\mathbf{a}$  is a vector of parameters. The decision boundary here will be a hyperplane passing through the origin of the feature space. Notice that we can turn this into a general hyperplane in the original feature space by extending each example's feature vector by attaching a 1 as the last component. This trick simplifies notation, which is why we adopt it here. It is straightforward to estimate  $\mathbf{a}$  using maximum likelihood. Note that

$$p(1|\mathbf{x}) = \frac{\exp \mathbf{a}^T \mathbf{x}}{1 + \exp \mathbf{a}^T \mathbf{x}}$$

and

$$p(-1|\mathbf{x}) = \frac{1}{1 + \exp \mathbf{a}^T \mathbf{x}},$$

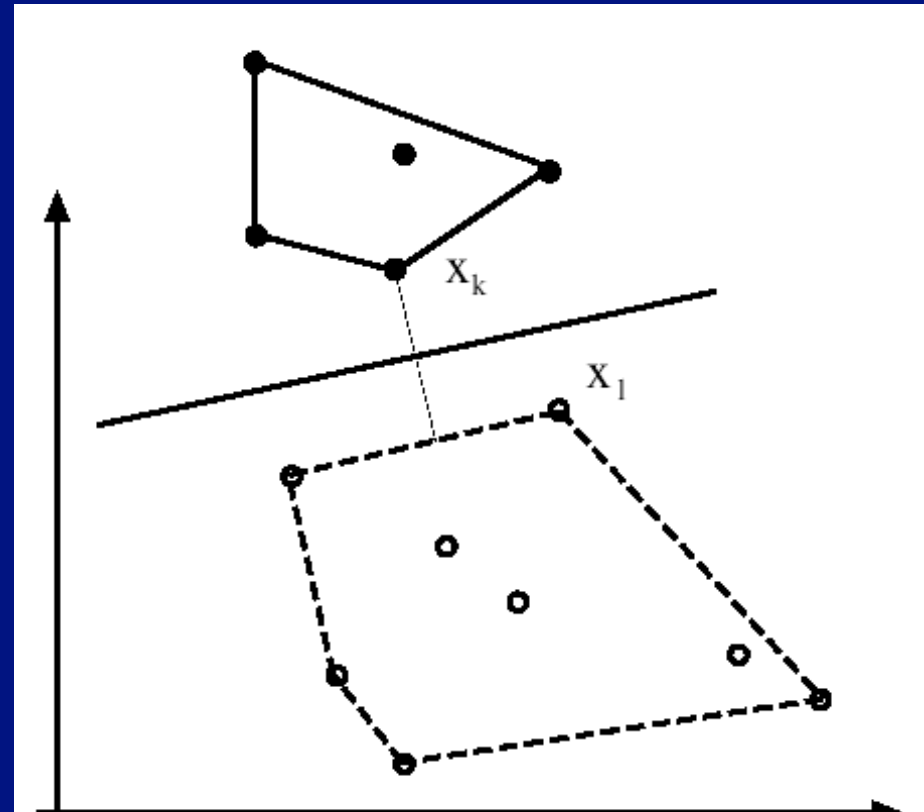
so that we can estimate the correct set of parameters  $\hat{\mathbf{a}}$  by solving for the minimum of the negative log-likelihood, i.e.,

$$\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \left[ - \sum_{i \in \text{examples}} \left( \frac{1 + y_i}{2} \right) \mathbf{a}^T \mathbf{x} - \log (1 + \mathbf{a}^T \mathbf{x}) \right].$$

It turns out that this problem is convex, and is easily solved by Newton's method (e.g., Hastie *et al.* (2009)).

# Decision boundaries

- The boundary matters
  - but the details of the probability model may not
- Seek a boundary directly
  - when we do so, many or most examples are irrelevant
- Support vector machine



# Support Vector Machines, easy case

- Classify with  $\text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$

- Linearly separable data means

$$y_i (\mathbf{w} \cdot \mathbf{x}_i + b) > 0$$

- Choice of hyperplane means

$$y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

- Hence distance

$$\begin{aligned} \text{dist}(\mathbf{x}_k, \text{hyperplane}) + \text{dist}(\mathbf{x}_l, \text{hyperplane}) &= \left( \frac{\mathbf{w}}{|\mathbf{w}|} \cdot \mathbf{x}_k + \frac{b}{|\mathbf{w}|} \right) - \left( \frac{\mathbf{w}}{|\mathbf{w}|} \cdot \mathbf{x}_l + \frac{b}{|\mathbf{w}|} \right) \\ &= \frac{\mathbf{w}}{|\mathbf{w}|} \cdot (\mathbf{x}_k - \mathbf{x}_l) = \frac{2}{|\mathbf{w}|} \end{aligned}$$

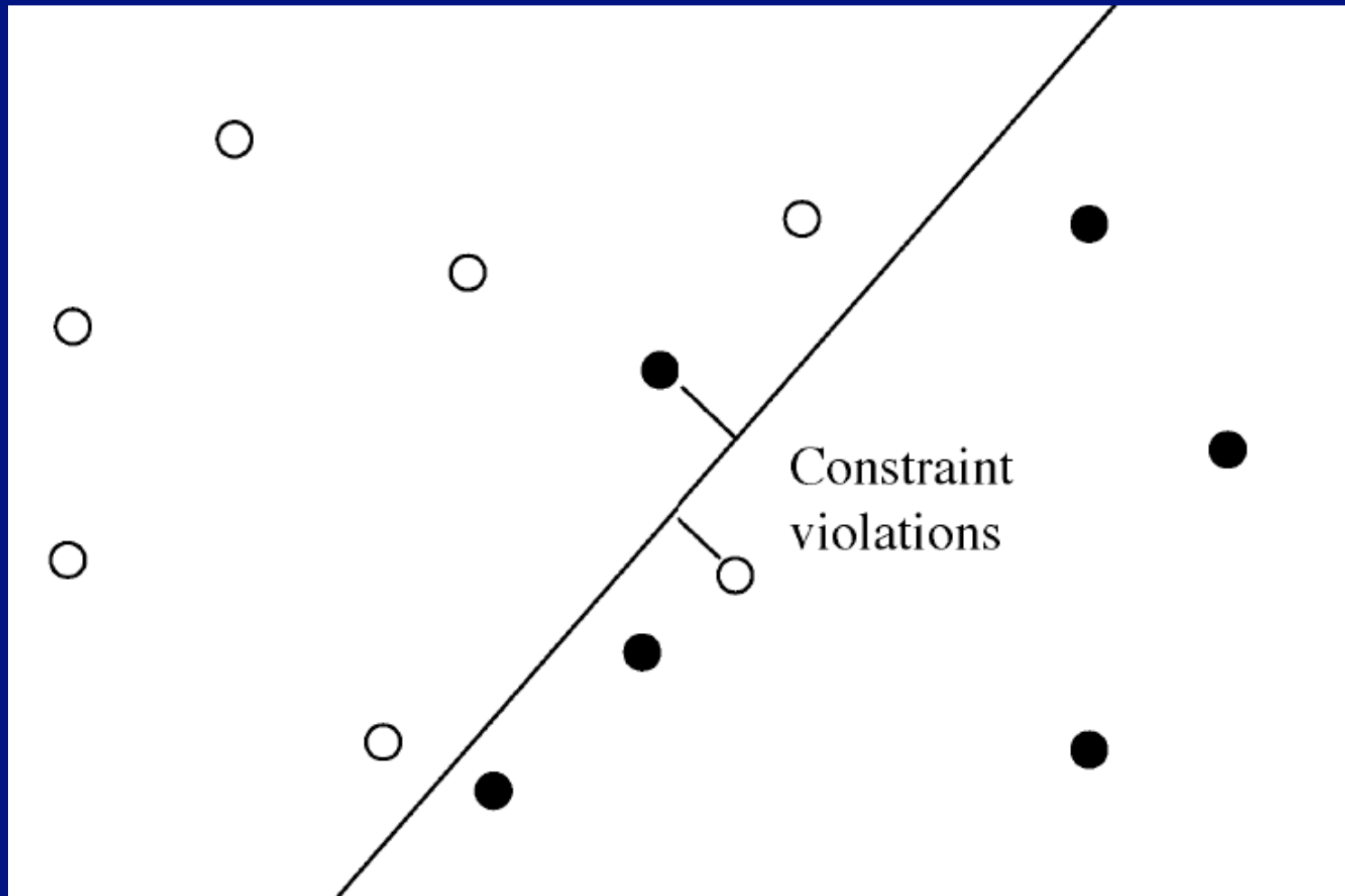
# Support Vector Machines, separable case

$$\text{minimize } (1/2)\mathbf{w} \cdot \mathbf{w}$$

$$\text{subject to } y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

By being clever about what  $\mathbf{x}$  means, I can have much more interesting boundaries.

# Data not linearly separable



# Data not linearly separable

Objective function becomes

$$\|\mathbf{w}\|^2/2 + C (\sum_i \xi_i)$$

Constraints become

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 - \xi_i \quad \text{for } y_i = +1$$

$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 + \xi_i \quad \text{for } y_i = -1$$

$$\xi_i \geq 0 \quad \forall i.$$

# SVM's

- Optimization problem is rather special
  - never ever use general purpose software for this
  - very well studied
- Methods available on the web
  - SVMlite
  - LibSVM
  - Pegasos
  - many others
- There are automatic feature constructions as well

### 15.3.3 Solving for SVMS and Kernel Machines

We obtain a support vector machine by solving one of the constrained optimization problems given above. Although these problems are quadratic programs, it is not a good idea to simply dump them into a general-purpose optimization package, because they have quite special structure. One would usually use one of the many packages available on the web for SVMs.

There are two general threads in solving for SVMs. One can either solve the primal problems (the ones shown here), or write out the Lagrangian, eliminate the primal variables  $w$  and  $b$ , and solve the dual problem in the Lagrange multipliers. This dual problem has a large number of variables because there is one Lagrange multiplier for each active constraint. However, our original argument about convex hulls suggests that most of these must be zero at the solution. Equivalently, most constraints are not active, because relatively few points are enough to determine a separating hyperplane. Dual solvers typically exploit this property, and are built around an efficient search for nonzero Lagrange multipliers.

The alternative to solving the dual problem is to solve the primal problem. This approach is particularly useful when the dataset is very large, and is unlikely to be linearly separable. In this case, the objective function is an estimate of the loss incurred in applying the classifier, regularized by the norm of the hyperplane. For many applications, it is sufficient to get the error rate below a threshold (as opposed to exactly minimizing it). This means that the value of the objective function is a guide to when training can stop, as long as one trains in primal. Modern primal training algorithms visit single examples at random, updating the estimated classifier slightly on each visit. These algorithms can be very efficient for extremely large datasets.

LIBSVM (which can be found using Google, or at <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>) is a dual solver that is now widely used; it searches for nonzero Lagrange multipliers using a clever procedure known as SMO (sequential minimal optimization). A good primal solver is PEGASOS; source code can be found using Google, or at <http://www.cs.huji.ac.il/~shais/code/index.html>.

SVMLight (Google, or <http://svmlight.joachims.org/>) is a comprehensive SVM package with numerous features. It can produce sophisticated estimates of the error rate, learn to rank as well as to classify, and copes with hundreds of thousands of examples. Andrea Vedaldi, Manik Varma, Varun Gulshan, and Andrew Zisserman publish code for a multiple kernel learning-based image classifier at <http://www.robots.ox.ac.uk/~vgg/software/MKL/>. Manik Varma publishes code for general multiple-kernel learning at <http://research.microsoft.com/en-us/um/people/manik/code/GMKL/download.html>, and for multiple-kernel learning using SMO at <http://research.microsoft.com/en-us/um/people/manik/code/SMO-MKL/download.html>. Peter Gehler and Sebastian Nowozin publish code for their recent multiple-kernel learning method at <http://www.vision.ee.ethz.ch/~pgehler/projects/iccv09/index.html>.

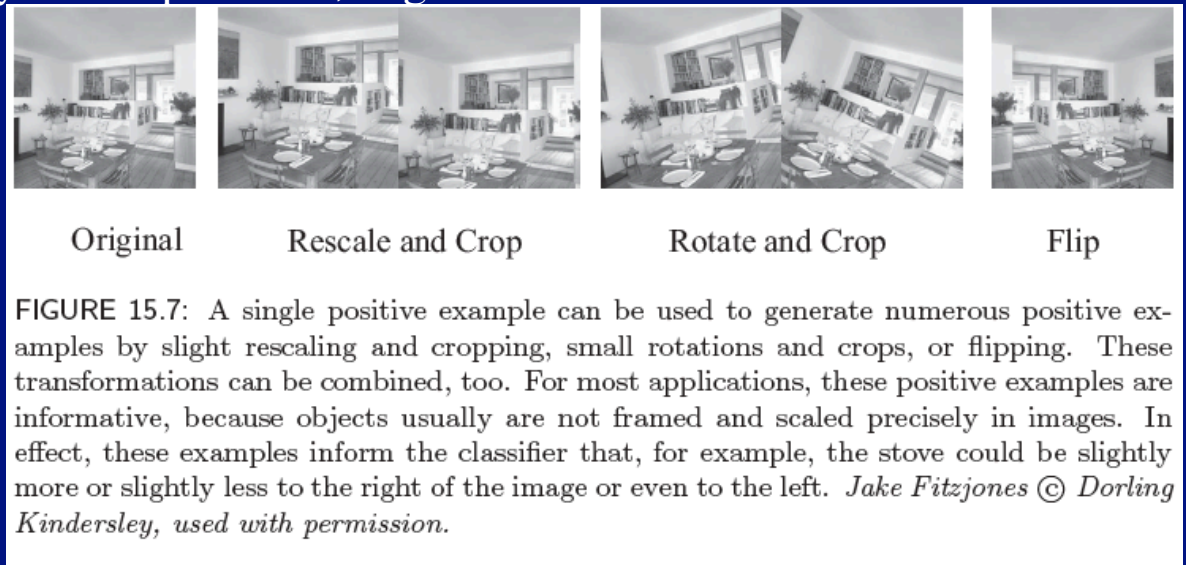


# Multiclass classification

- Many strategies
  - Easy with k-nearest neighbors
  - 1-vs-all
    - for each class, construct a two class classifier comparing it to all other classes
    - take the class with best output
      - if output is greater than some value
  - Multiclass logistic regression
    - $\log(P(i|\text{features})) - \log(P(k|\text{features})) = (\text{linear expression})$
    - many more parameters
    - harder to train with maximum likelihood
    - still convex

# Useful tricks

- Jittering data
  - you can make many useful positives, negatives out of some



- Hard negative mining
  - negatives are often common - find ones that you get wrong by a search

# Evaluating classifiers

- Always
  - train on training set, evaluate on test set
    - test set performance might/should be worse than training set
- Options
  - Total error rate
    - always less than 50% for two class
  - Receiver operating curve
    - because we might use different thresholds
  - Class confusion matrix
    - for multiclass

# Receiver operating curve

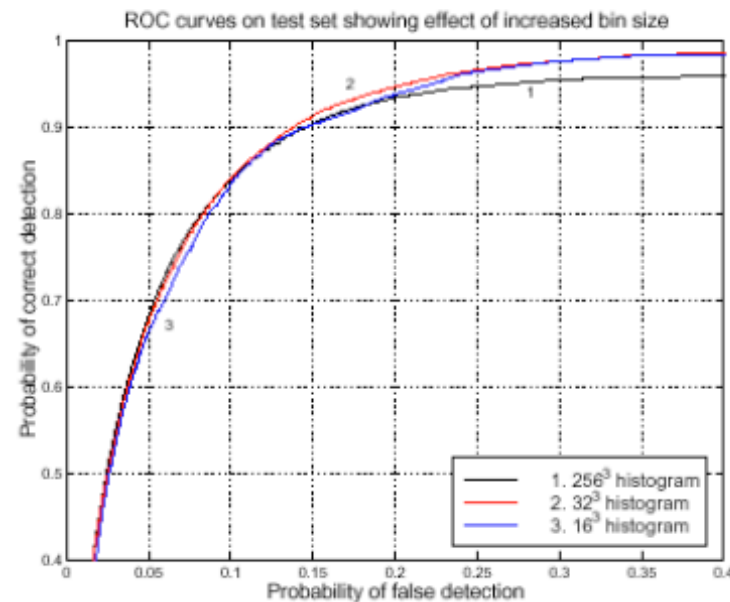


FIGURE 15.4: The receiver operating curve for a classifier, used to build a skin detector by Jones and Rehg. This curve plots the detection rate against the false-negative rate for a variety of values of the parameter  $\theta$ . A perfect classifier has an ROC that, on these axes, is a horizontal line at 100% detection. There are three different versions of this classifier, depending on the detailed feature construction; each has a slightly different ROC. *This figure was originally published as Figure 7 of "Statistical color models with application to skin detection," by M.J. Jones and J. Rehg, Proc. IEEE CVPR, 1999 © IEEE, 1999.*

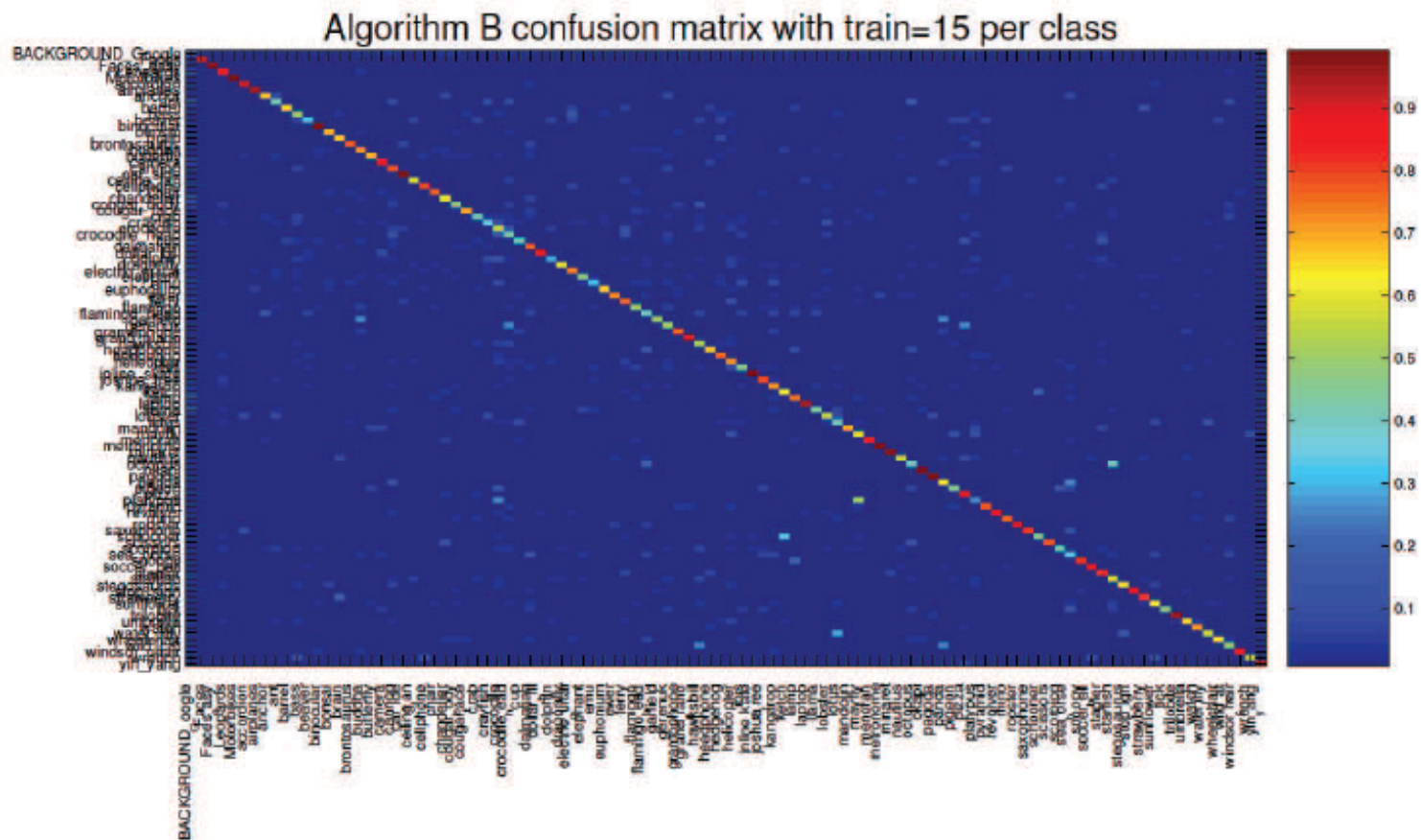


FIGURE 15.3: An example of a class confusion matrix from a recent image classification system, due to Zhang *et al.* (2006a). The vertical bar shows the mapping of color to number (warmer colors are larger numbers). Note the redness of the diagonal; this is good, because it means the diagonal values are large. There are spots of large off-diagonal values, and these are informative, too. For example, this system confuses: schooners and ketches (understandable); waterlily and lotus (again, understandable); and platypus and mayfly (which might suggest some feature engineering would be a good idea). *This figure was originally published as Figure 5 of “SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition,” by H. Zhang, A. Berg, M. Maire, and J. Malik, Proc. IEEE CVPR, 2006, © IEEE, 2006.*