Image Categorization

Computer Vision

CS 543 / ECE 549

University of Illinois

Derek Hoiem

- Thanks for feedback
- HW 3 is out
- Project guidelines are out

Last classes

- Object recognition: localizing an object instance in an image

- Face recognition: matching one face image to another

Today's class: categorization

- Overview of image categorization
- Representation
  - Image histograms
- Classification
  - Important concepts in machine learning
  - What the classifiers are and when to use them

- What is a category?

- Why would we want to put an image in one?

  To predict, describe, interact. To organize.

- Many different ways to categorize

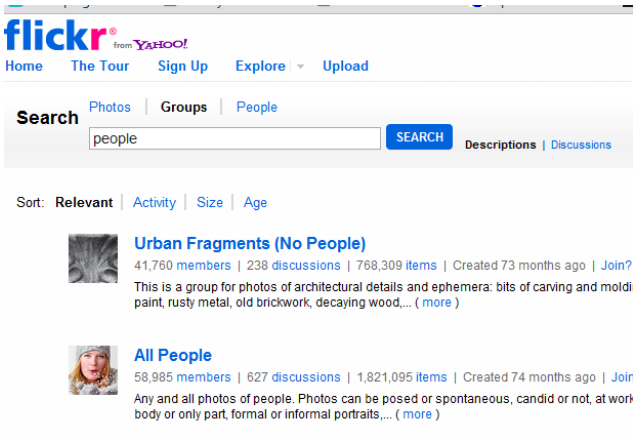


Image Categorization

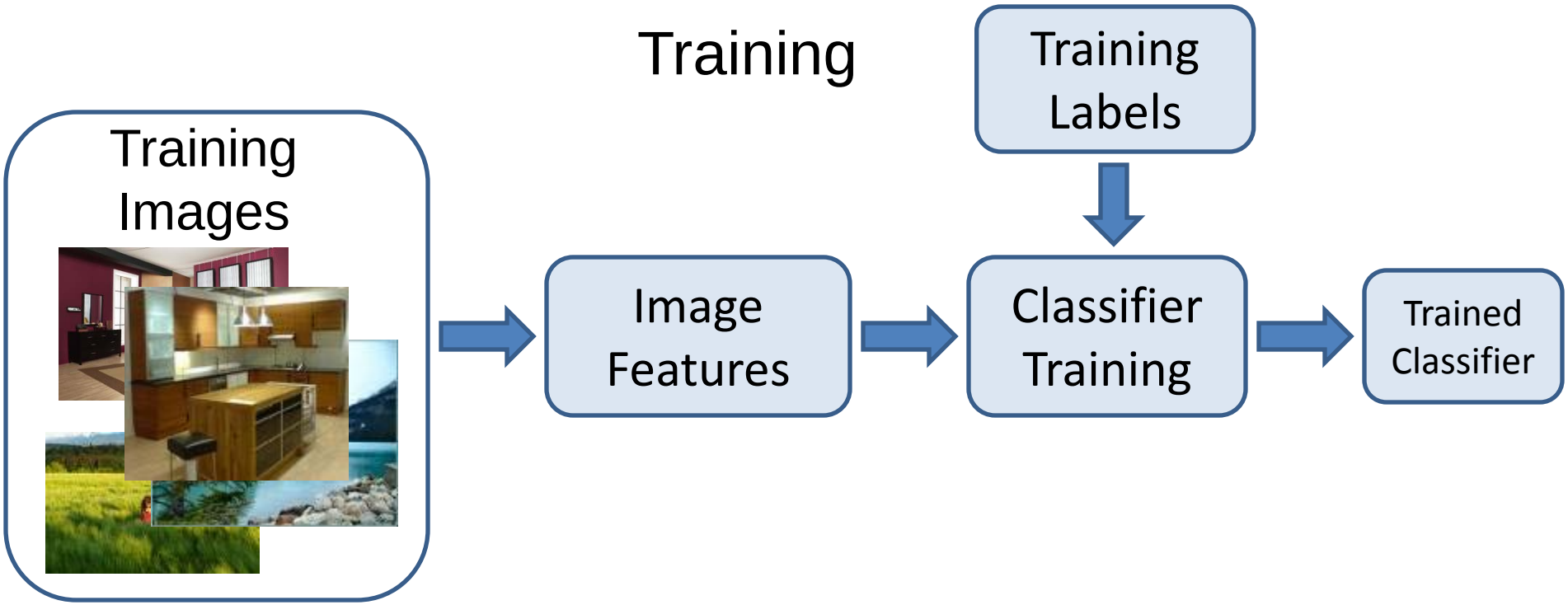
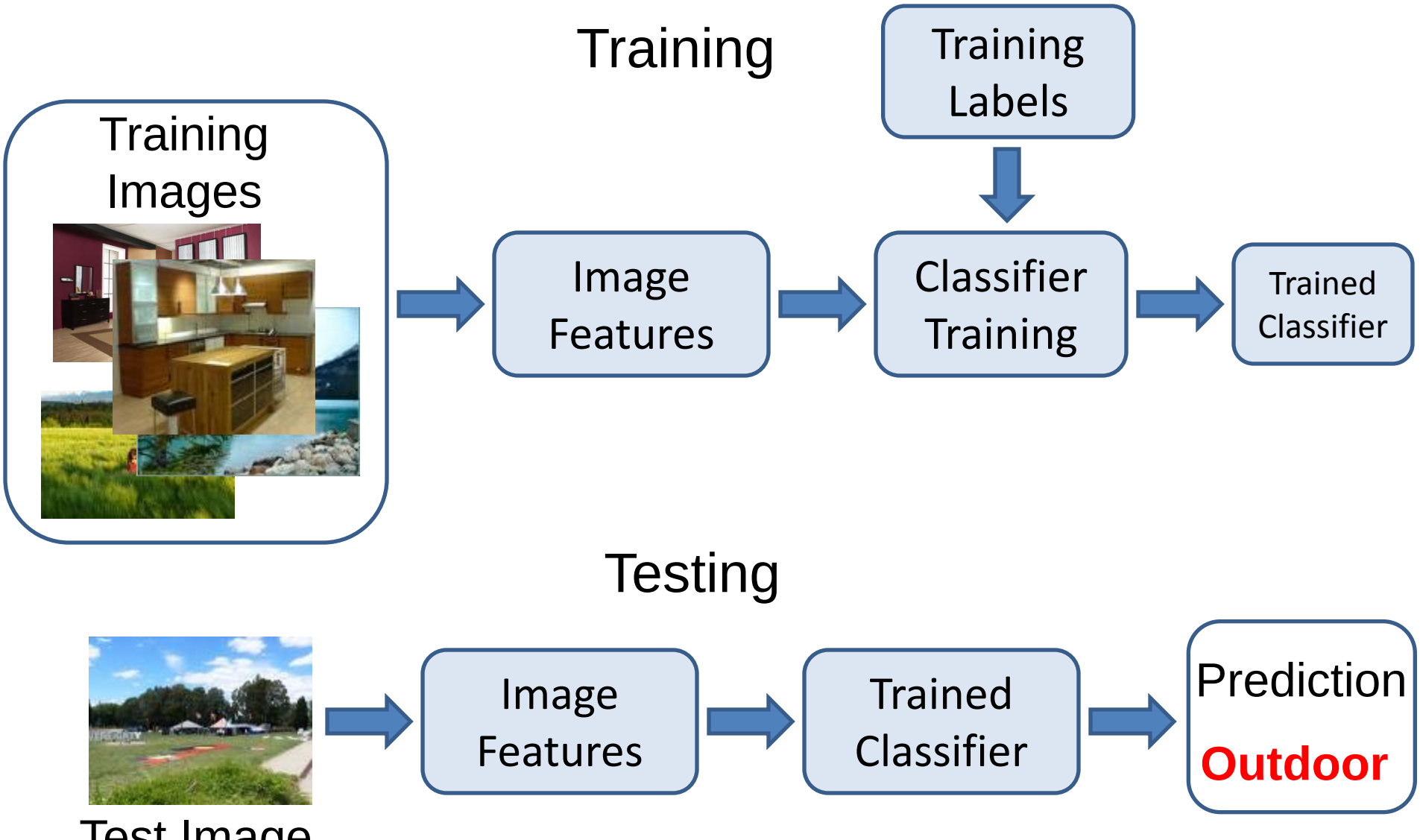
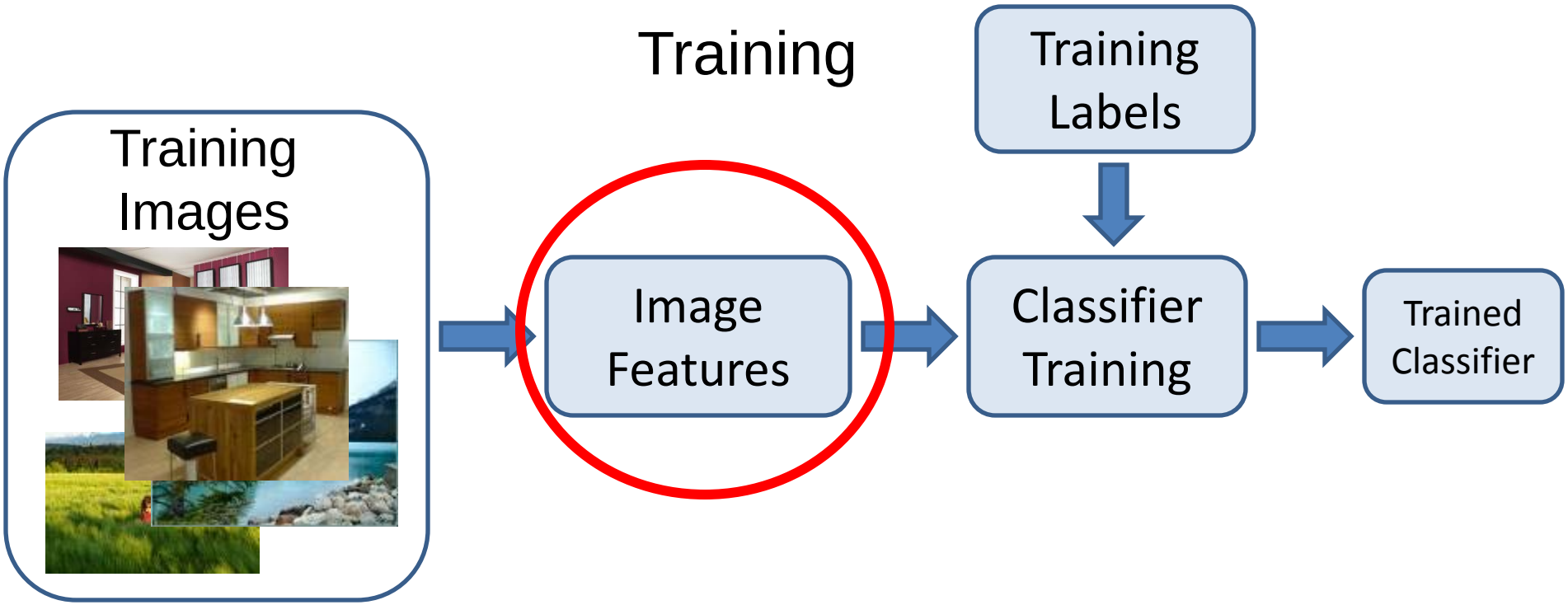


Image Categorization



Part 1: Image features



General Principles of Representation

- Coverage
  - Ensure that all relevant info is captured
- Concision
  - Minimize number of features without sacrificing coverage
- Directness
  - Ideal features are independently useful for prediction

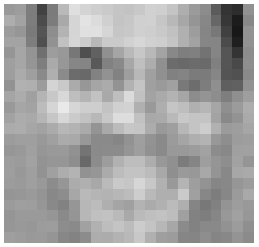
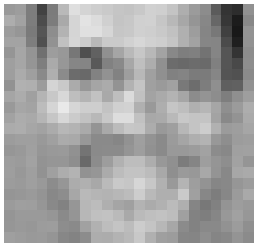


Image Intensity

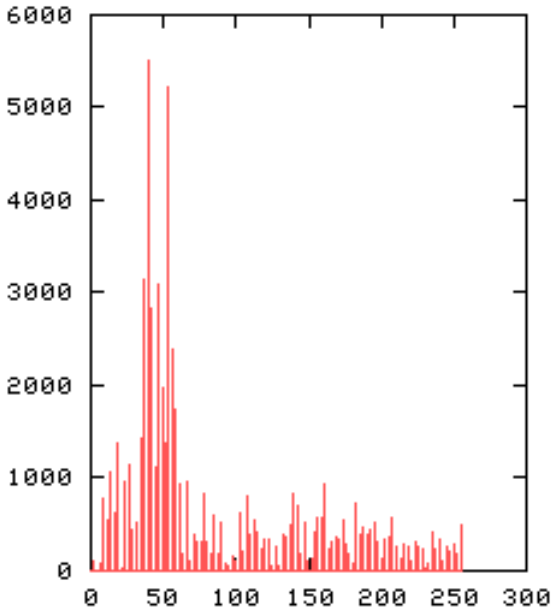
Image representations

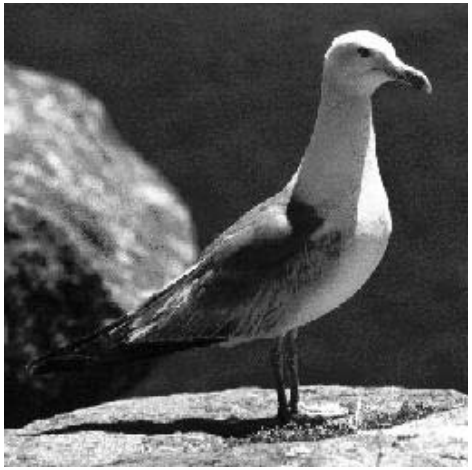
- Templates
  - Intensity, gradients, etc.



- Histograms
  - Color, texture, SIFT descriptors, etc.

Image Representations: Histograms





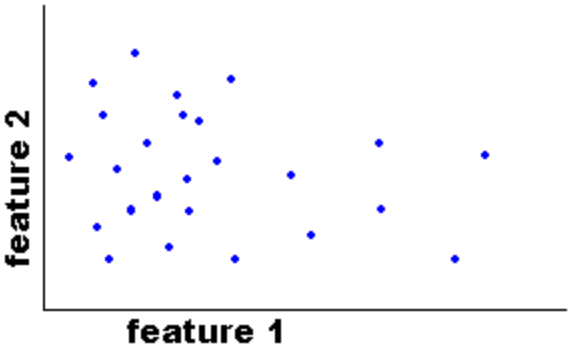
Global histogram

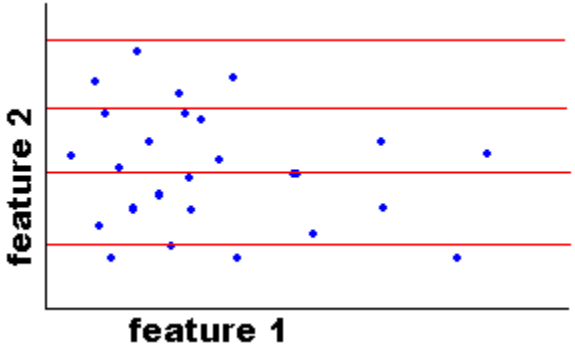
- Represent distribution of features
  - Color, texture, depth, …

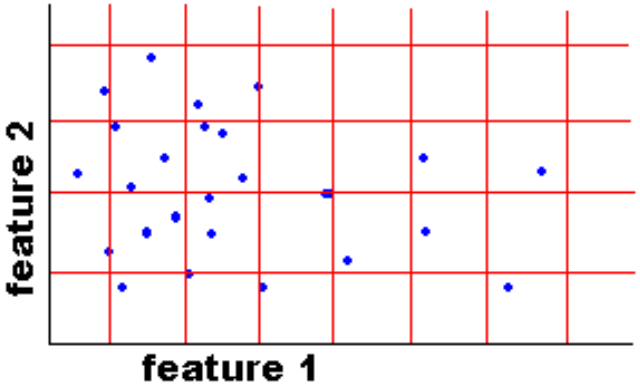
Images from Dave Kauchak

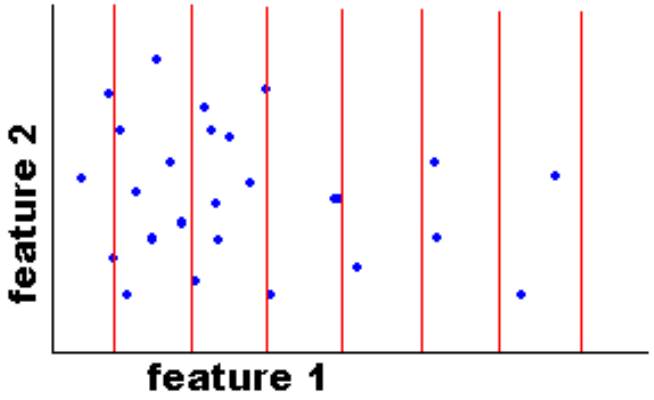
Image Representations: Histograms

Histogram: Probability or count of data in each bin









- Joint histogram
  - Requires lots of data
  - Loss of resolution to avoid empty bins

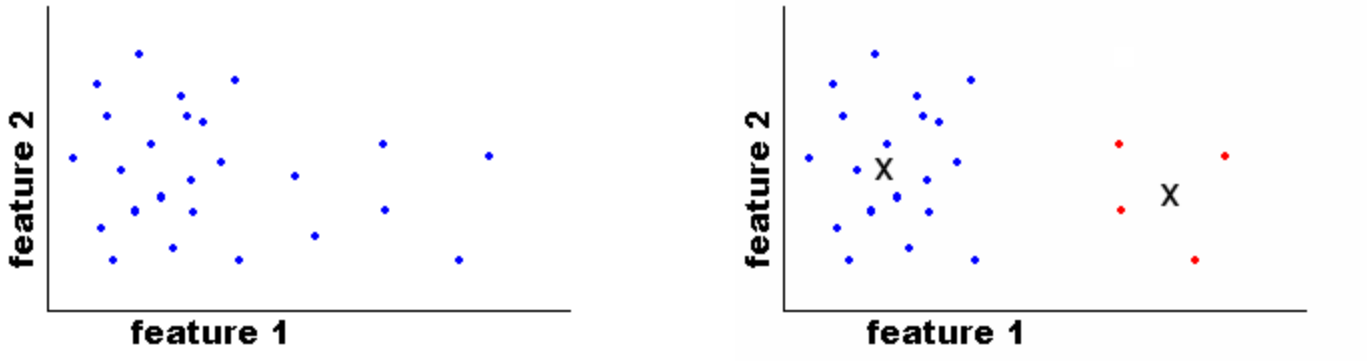
Marginal histogram

- Requires independent features
- More data/bin than joint histogram

Images from Dave Kauchak

Image Representations: Histograms

Clustering



Use the same cluster centers for all images

Images from Dave Kauchak

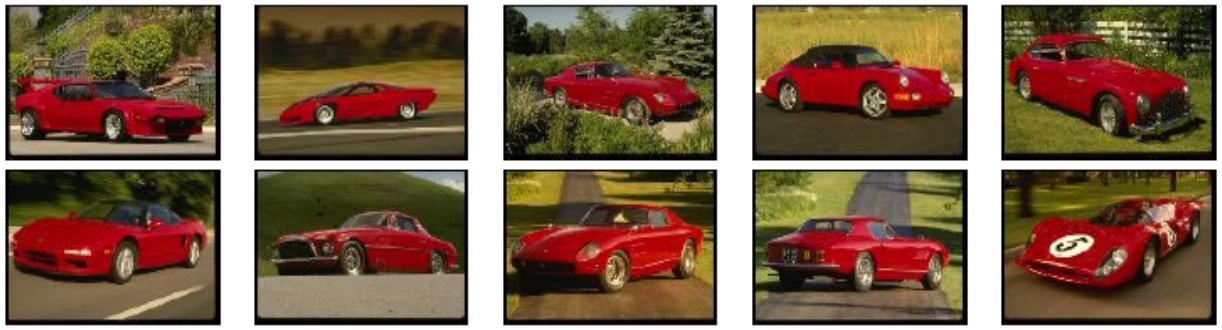
Computing histogram distance

$$\text{histint}(h_i, h_j) = 1 - \sum_{m=1}^{K} \min\left(h_i(m), h_j(m)\right)$$

Histogram intersection (assuming normalized histograms)

$$\chi^2(h_i, h_j) = \frac{1}{2} \sum_{m=1}^{K} \frac{[h_i(m) - h_j(m)]^2}{h_i(m) + h_j(m)}$$

Chi-squared Histogram matching distance



Cars found by color histogram matching using chi-squared

Histograms: Implementation issues

- Quantization
  - Grids: fast but applicable only with few dimensions
  - Clustering: slower but can quantize data in higher dimensions

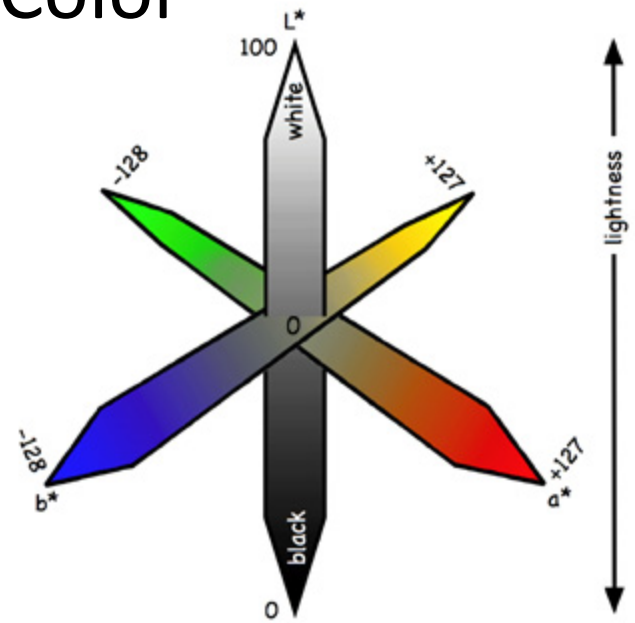
Few Bins
Need less data
Coarser representation

Many Bins
Need more data
Finer representation

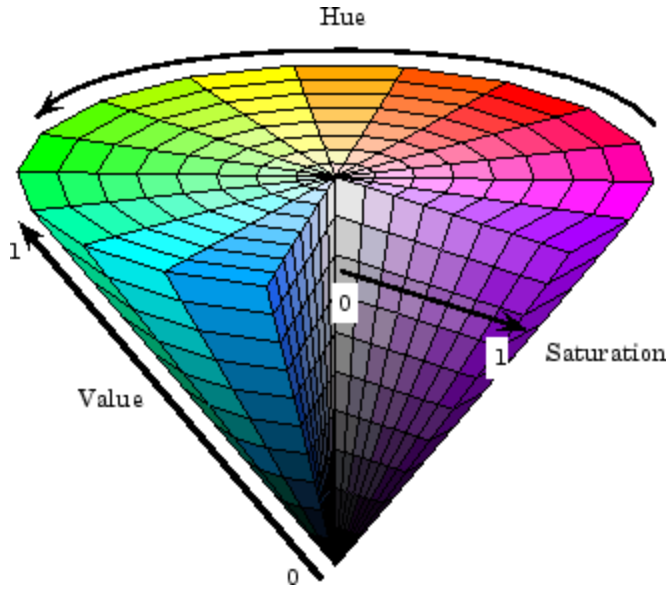
- Matching
  - Histogram intersection or Euclidean may be faster
  - Chi-squared often works better
  - Earth mover's distance is good for when nearby bins represent similar values

What kind of things do we compute histograms of?

- Color



L*a*b* color space

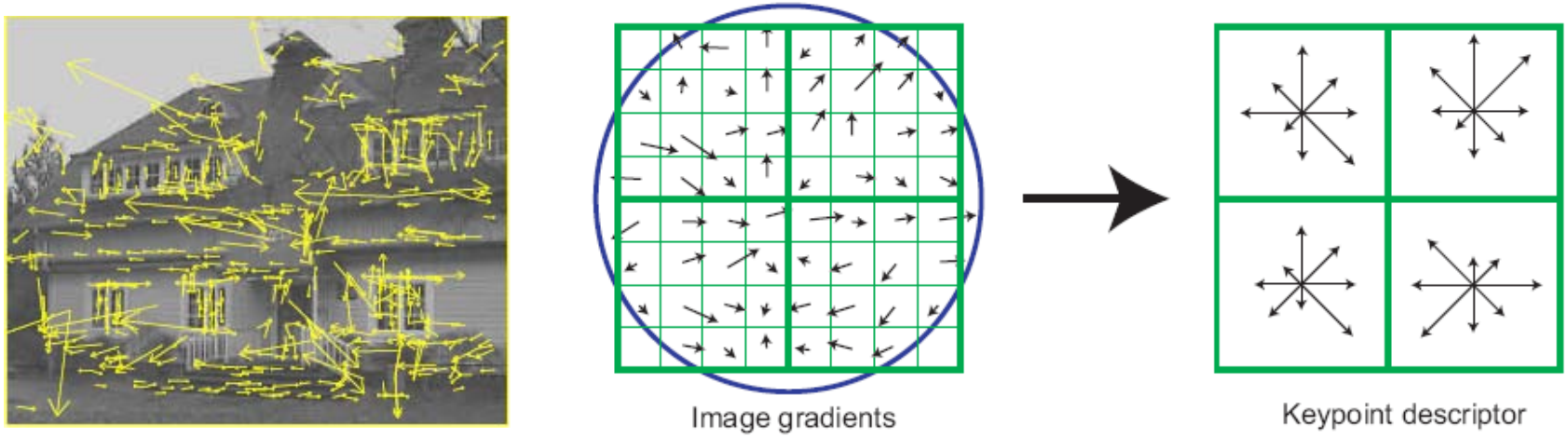


HSV color space

- Texture (filter banks or HOG over regions)

What kind of things do we compute histograms of?

- Histograms of oriented gradients



SIFT – Lowe IJCV 2004

- “Bag of words”

Image Categorization: Bag of Words

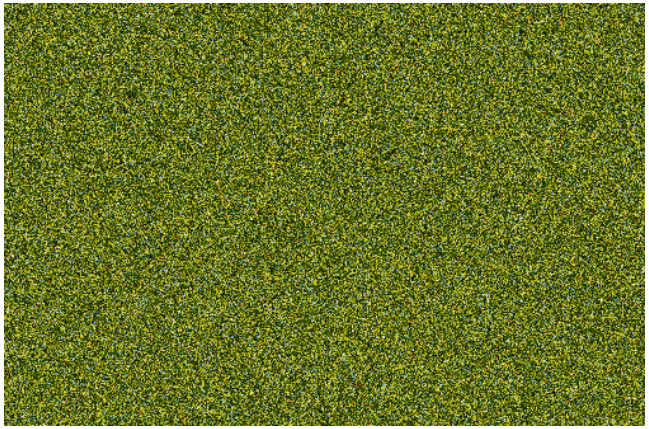
Training

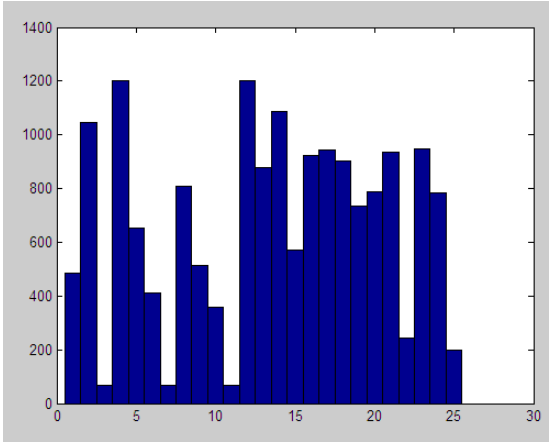
1. Extract keypoints and descriptors for all training images
2. Cluster descriptors
3. Quantize descriptors using cluster centers to get “visual words”
4. Represent each image by normalized counts of “visual words”
5. Train classifier on labeled examples using histogram values as features

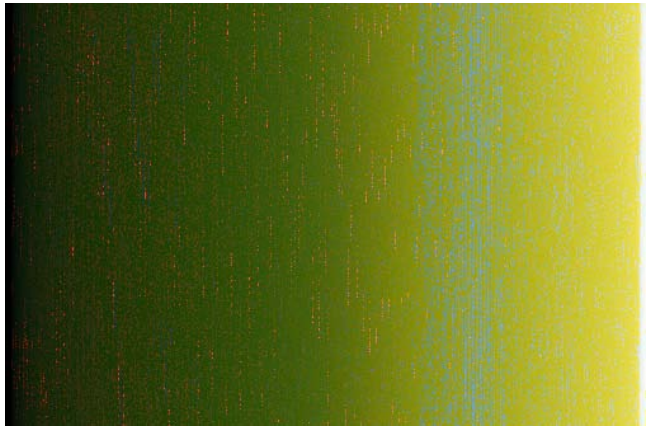
Testing

1. Extract keypoints/descriptors and quantize into visual words
2. Compute visual word histogram
3. Compute label or confidence using classifier

But what about layout?

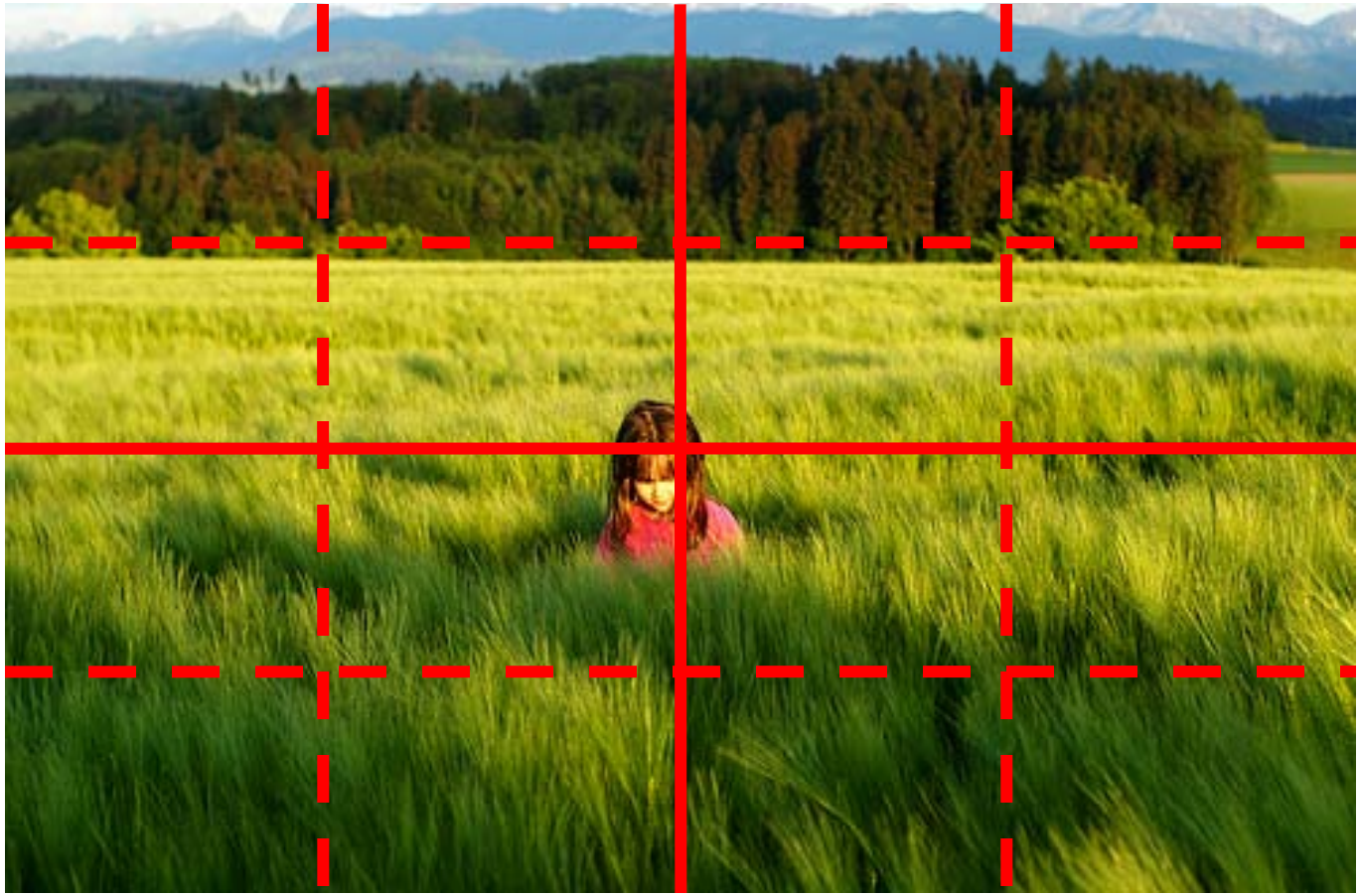






All of these images have the same color histogram

Spatial pyramid



Compute histogram in each spatial bin

Right features depend on what you want to know

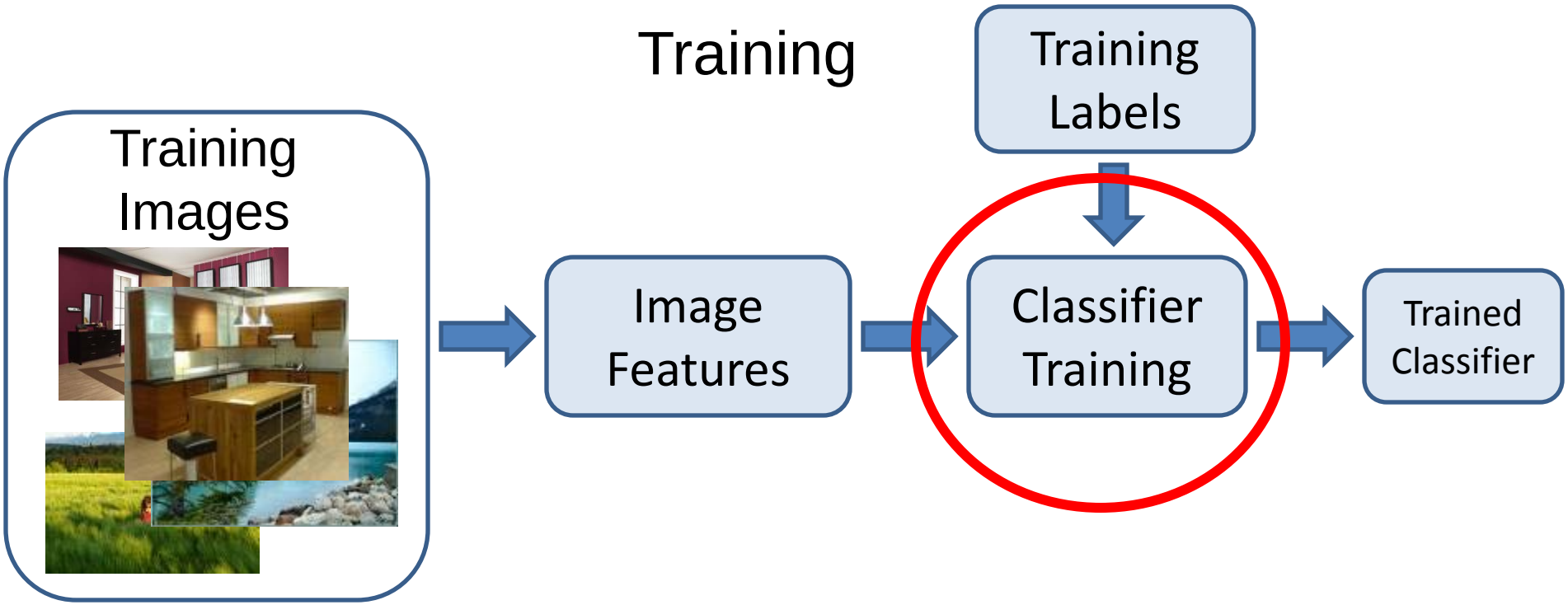
- Shape: scene-scale, object-scale, detail-scale
  - 2D form, shading, shadows, texture, linear perspective
- Material properties: albedo, feel, hardness, …
  - Color, texture
- Motion
  - Optical flow, tracked points
- Distance
  - Stereo, position, occlusion, scene shape
  - If known object: size, other objects

Things to remember about representation

- Most features can be thought of as templates, histograms (counts), or combinations

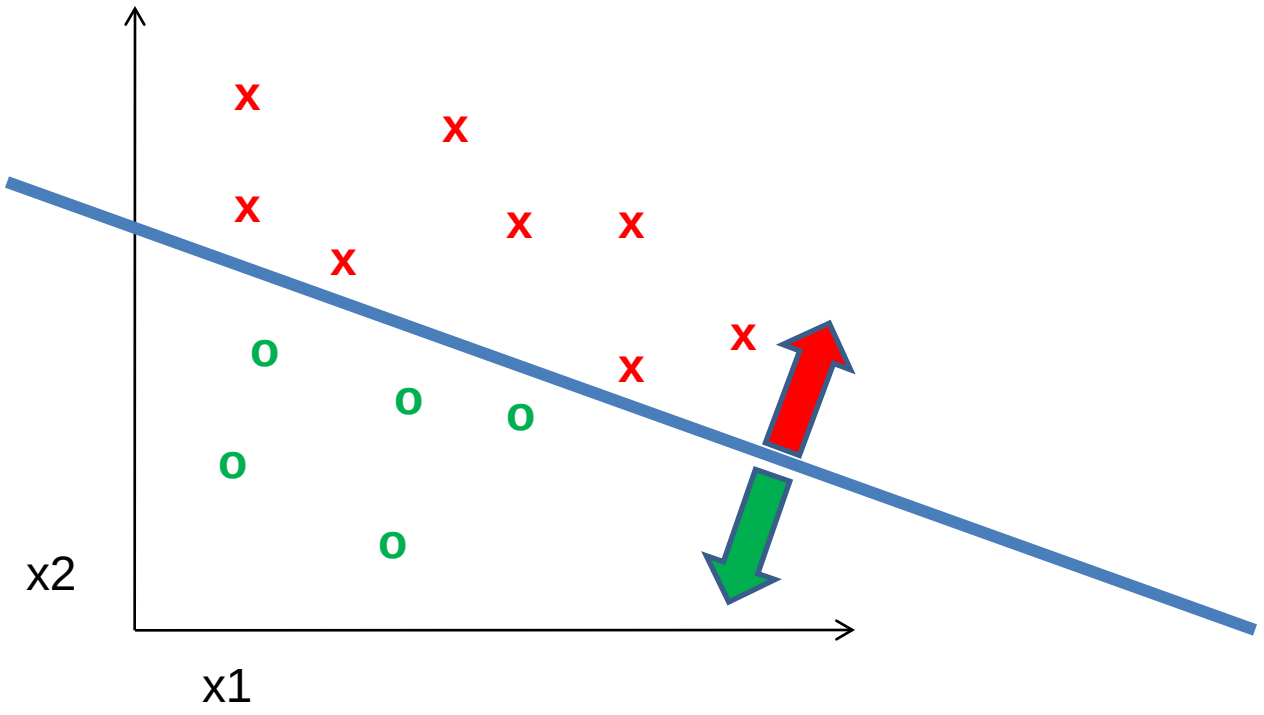
- Think about the right features for the problem
  - Coverage
  - Concision
  - Directness

Part 2: Classifiers



Learning a classifier

Given some set of features with corresponding labels, learn a function to predict the labels from the features



One way to think about it…

- Training labels dictate that two examples are the same or different, in some sense
- Features and distance measures define visual similarity
- Classifiers try to learn weights or parameters for features and distance measures so that visual similarity predicts label similarity

Many classifiers to choose from

- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- RBMs
- Etc.

Which is the best one?

No Free Lunch Theorem

Bias-Variance Trade-off

$$E(MSE) = noise^2 + bias^2 + variance$$

Unavoidable error

Error due to incorrect assumptions

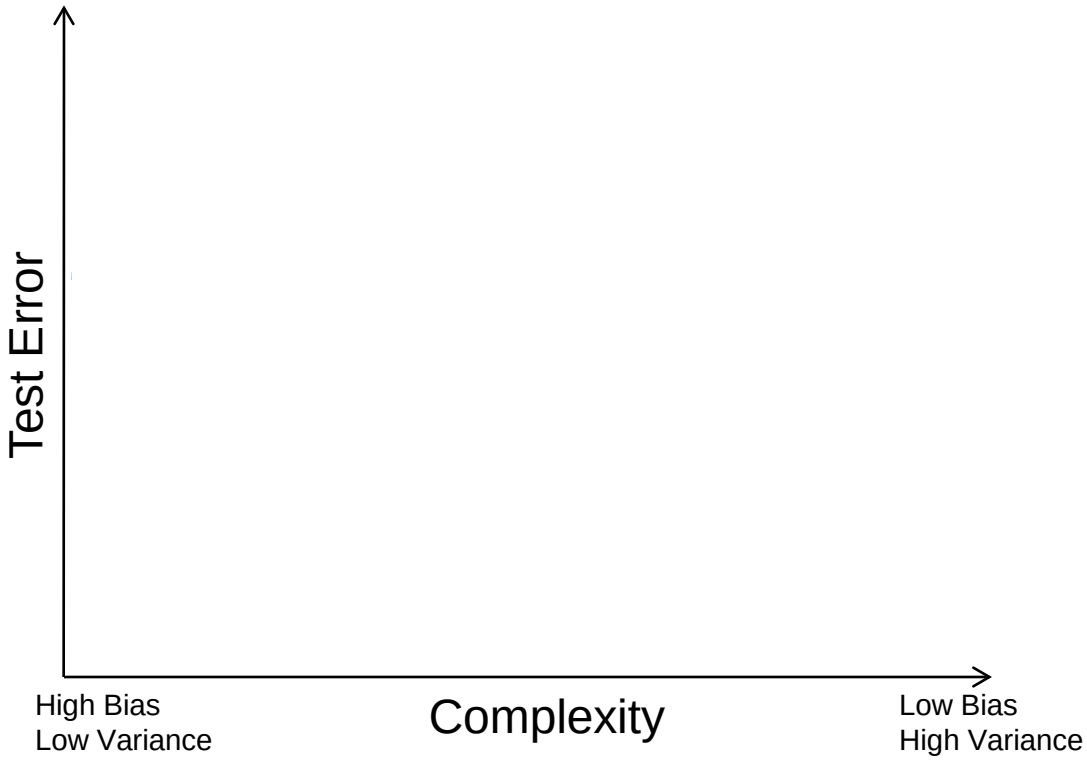
Error due to variance of training samples

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

- http://www.stat.cmu.edu/~larry/=stat707/notes3.pdf
- http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf

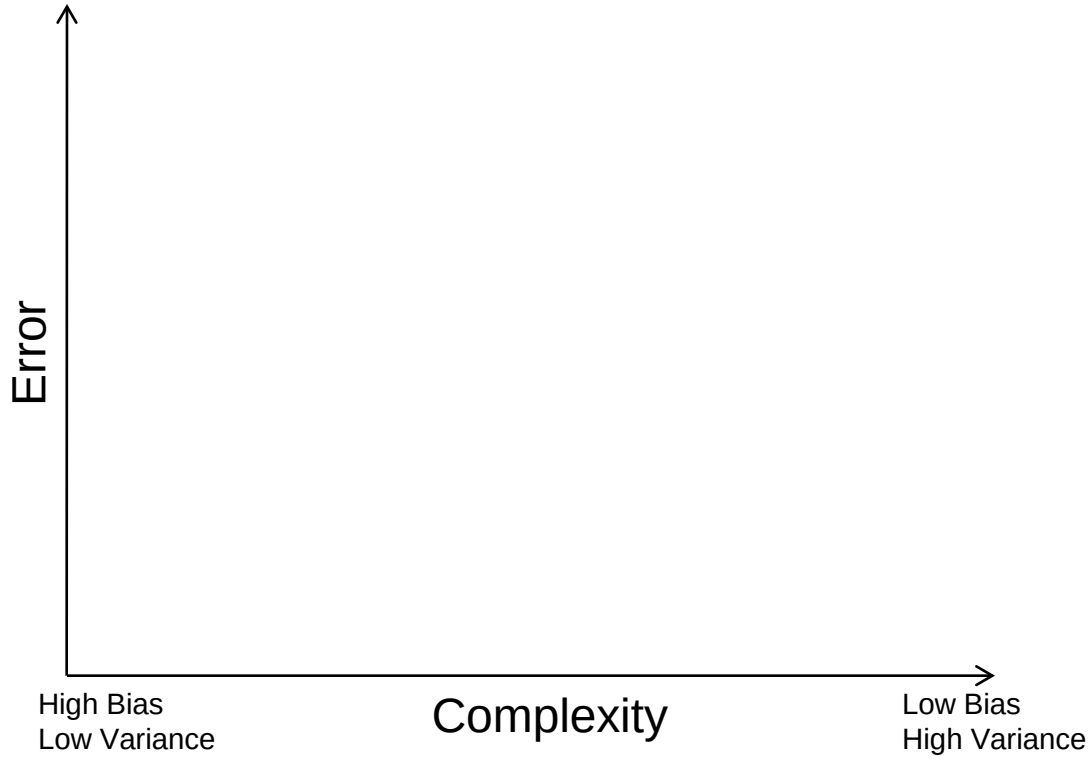
Bias and Variance

$$\text{Error} = \text{noise}^2 + \text{bias}^2 + \text{variance}$$

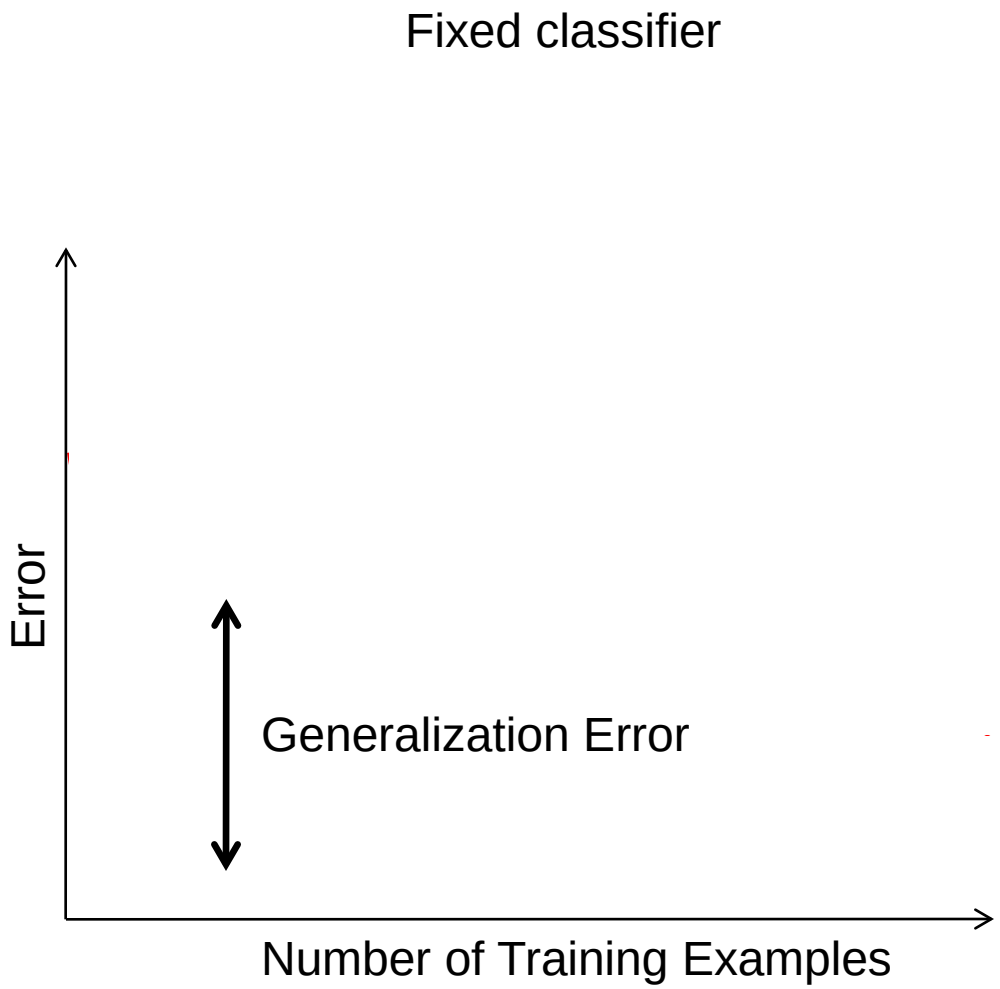


Choosing the trade-off

- Need validation set
- Validation set not same as test set



Effect of Training Size



How to measure complexity?

- VC dimension

What is the VC dimension of a linear classifier for N-dimensional features? For a nearest neighbor classifier?

Upper bound on generalization error

Training error + $\sqrt{\frac{h(\log(2N/h)+1)-\log(\eta/4)}{N}}$

N: size of training set
h: VC dimension
η: 1-probability that bound holds

- Other ways: number of parameters, etc.

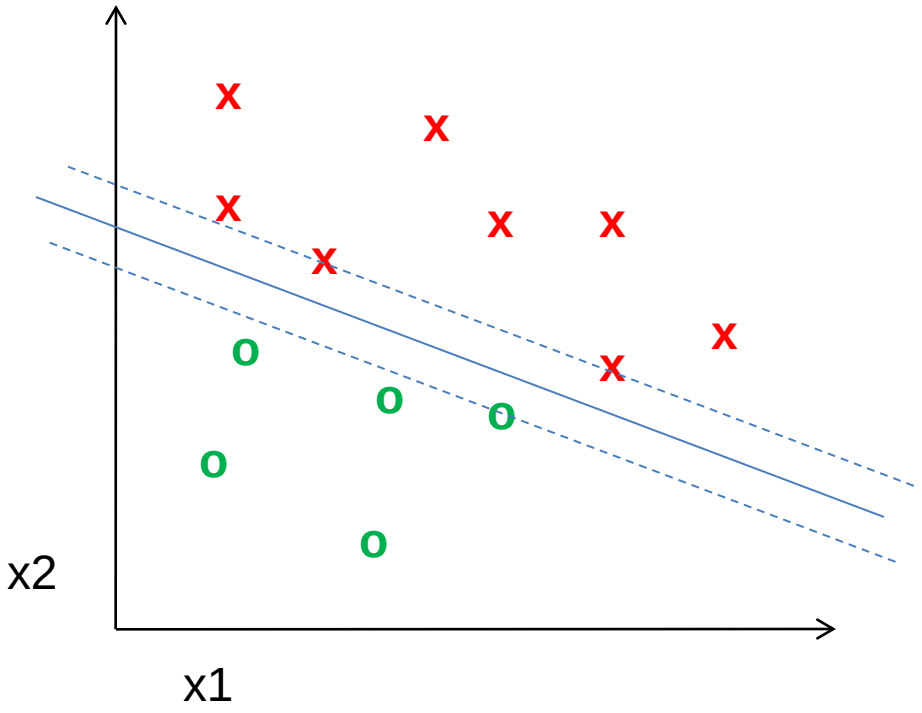
How to reduce variance?

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

Which of these could actually lead to greater error?

Reducing Risk of Error

- Margins



The perfect classification algorithm

- Objective function: encodes the right loss for the problem
- Parameterization: makes assumptions that fit the problem
- Regularization: right level of regularization for amount of training data
- Training algorithm: can find parameters that maximize objective on training set
- Inference algorithm: can solve for objective function in evaluation

Generative vs. Discriminative Classifiers

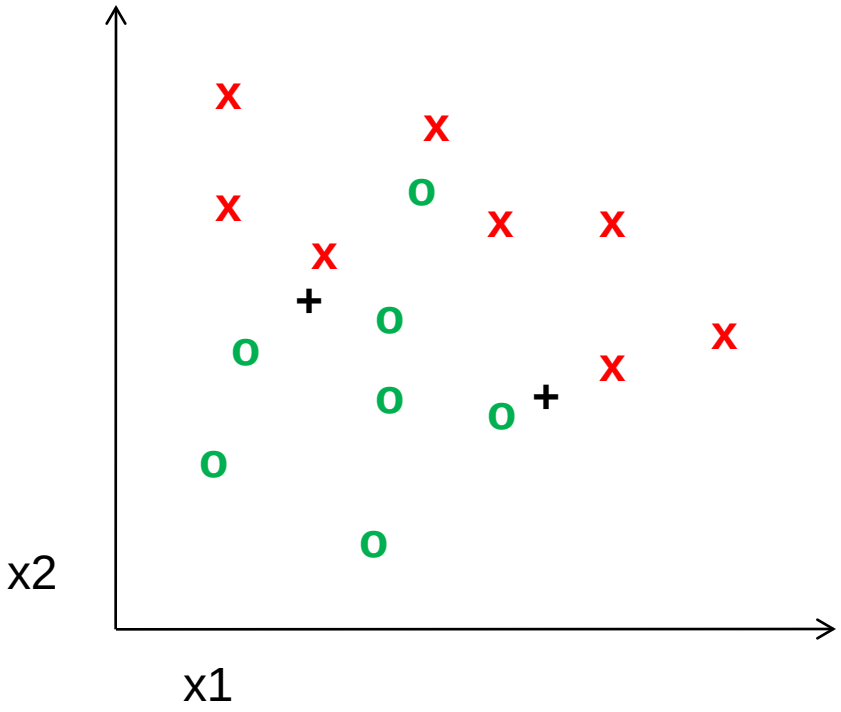
Generative

- Training
  - Models the data and the labels
  - Assume (or learn) probability distribution and dependency structure
  - Can impose priors
- Testing
  - $P(y=1, x) / P(y=0, x) > t$?
- Examples
  - Foreground/background GMM
  - Naïve Bayes classifier
  - Bayesian network

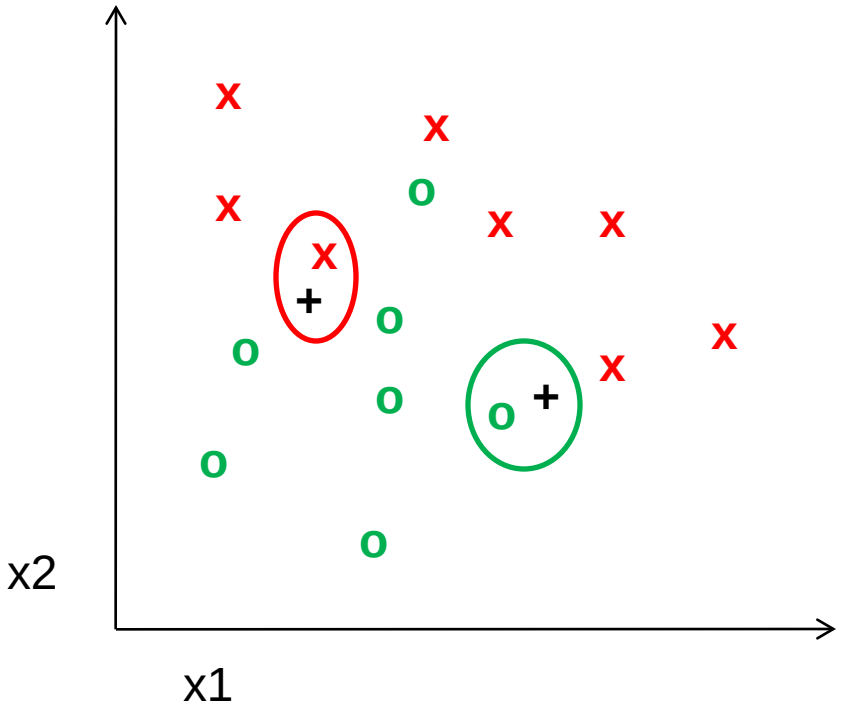
Discriminative

- Training
  - Learn to directly predict the labels from the data
  - Assume form of boundary
  - Margin maximization or parameter regularization
- Testing
  - $f(x) > t$; e.g., $w^T x > t$
- Examples
  - Logistic regression
  - SVM
  - Boosted decision trees

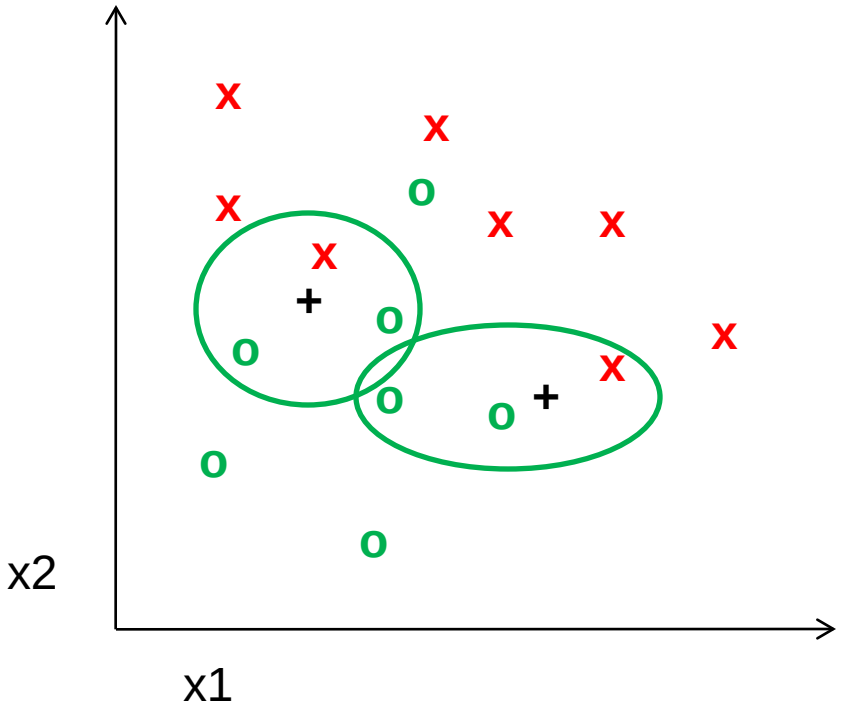
K-nearest neighbor



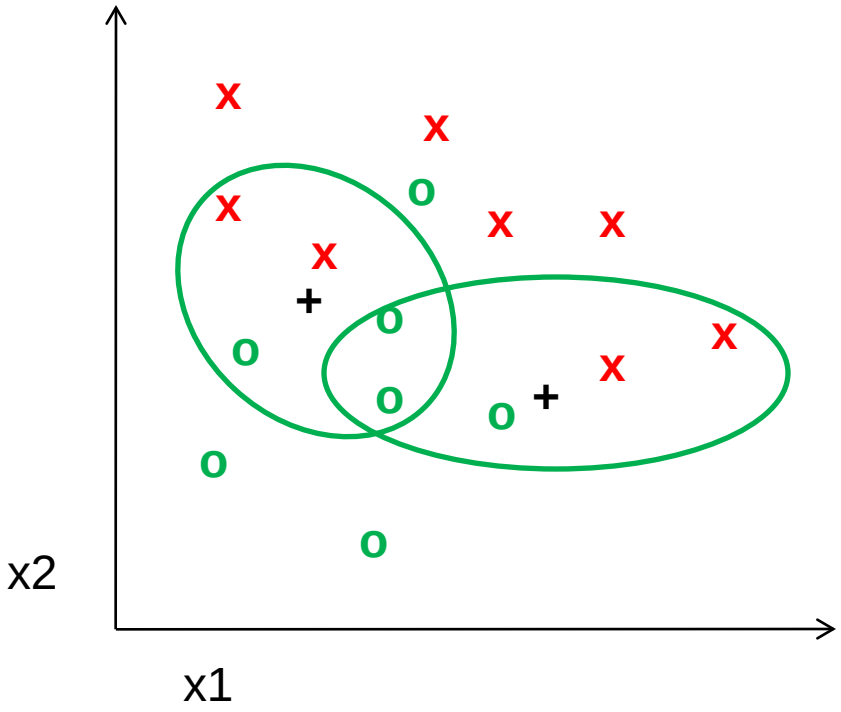
1-nearest neighbor



3-nearest neighbor



5-nearest neighbor



What is the parameterization? The regularization? The training algorithm? The inference?

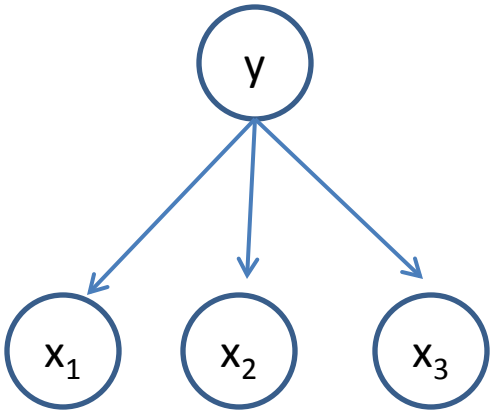
Is K-NN generative or discriminative?

Using K-NN

- Simple, a good one to try first
- With infinite examples, 1-NN provably has error that is at most twice Bayes optimal error

Naïve Bayes

- Objective
- Parameterization
- Regularization
- Training
- Inference

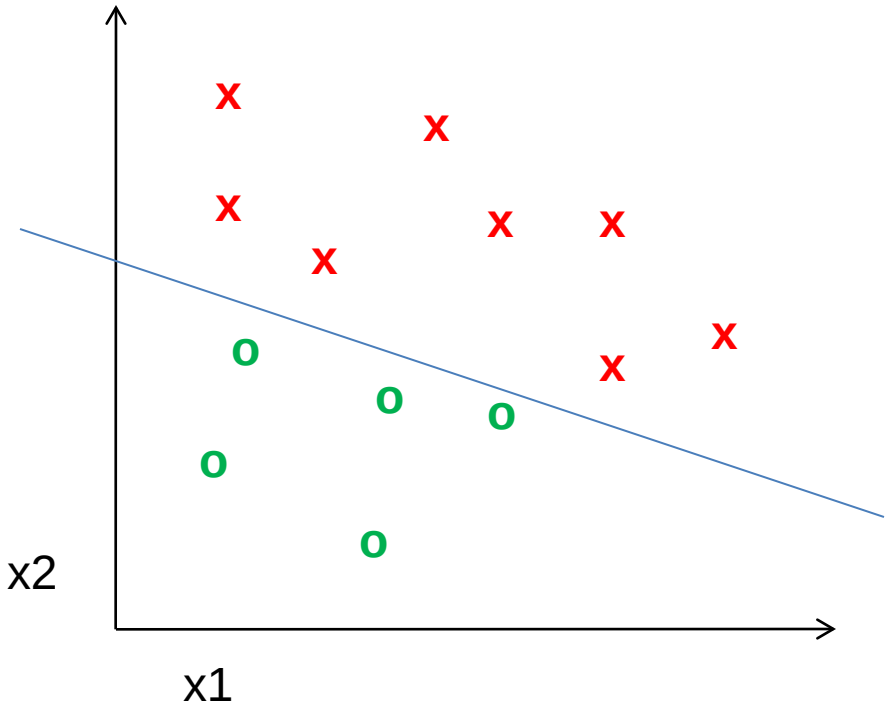


Using Naïve Bayes

- Simple thing to try for categorical data
- Very fast to train/test

Classifiers: Logistic Regression

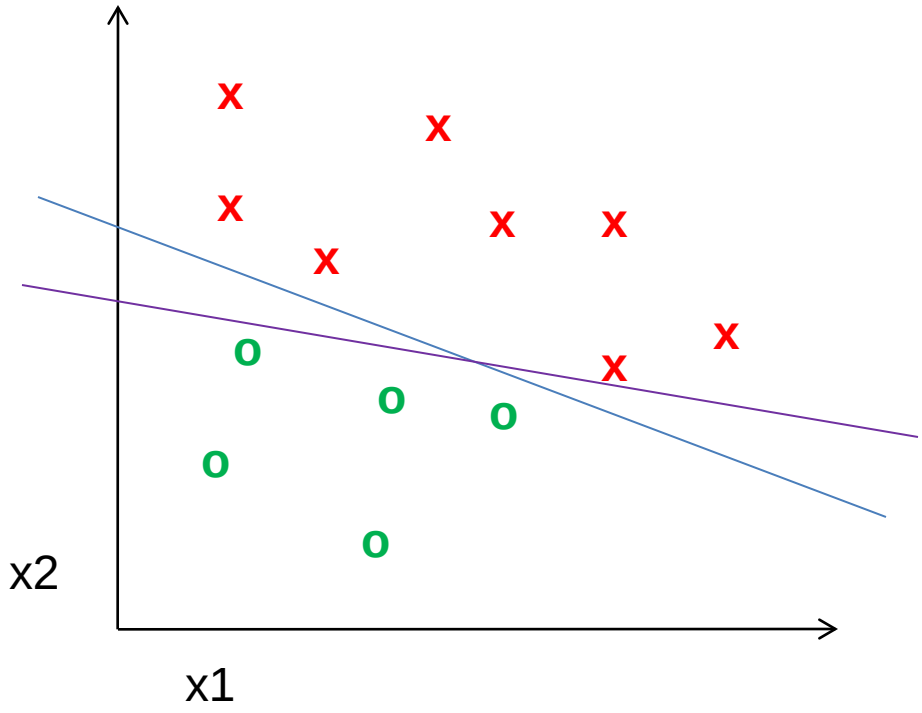
- Objective
- Parameterization
- Regularization
- Training
- Inference



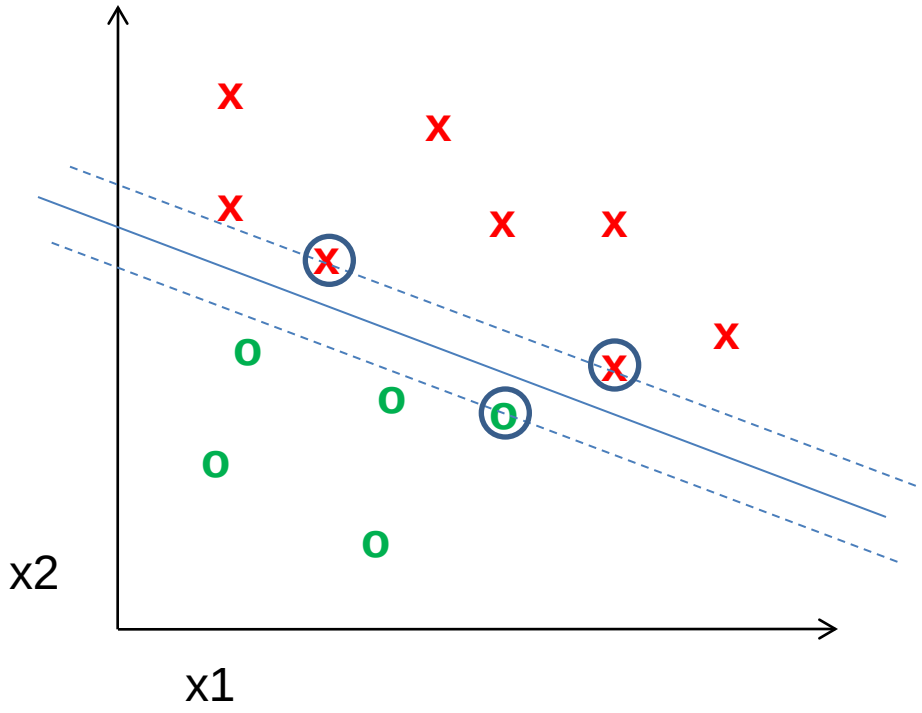
Using Logistic Regression

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

Classifiers: Linear SVM

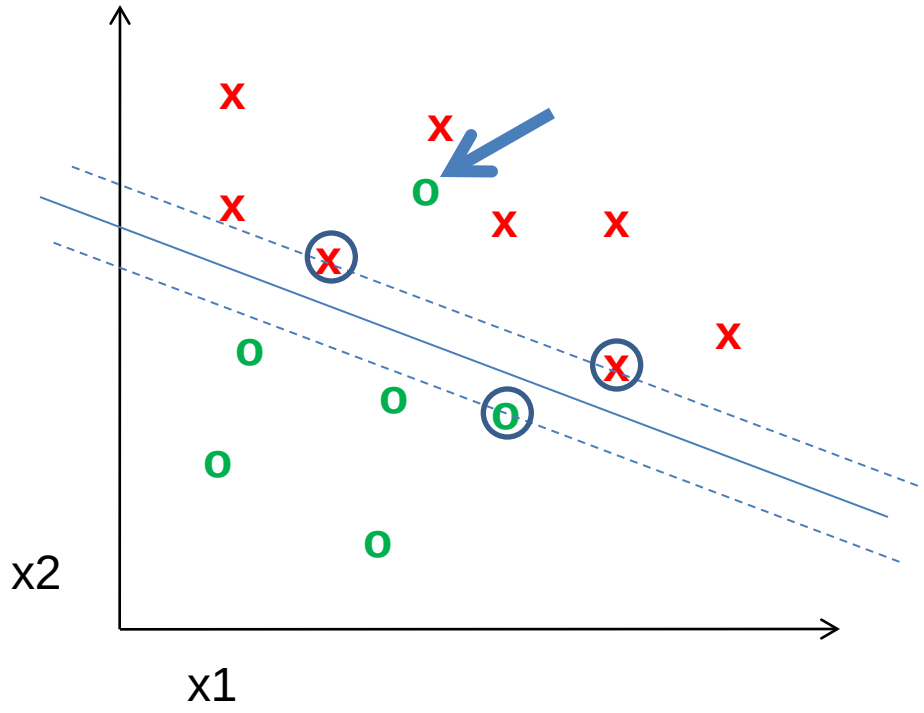


Classifiers: Linear SVM

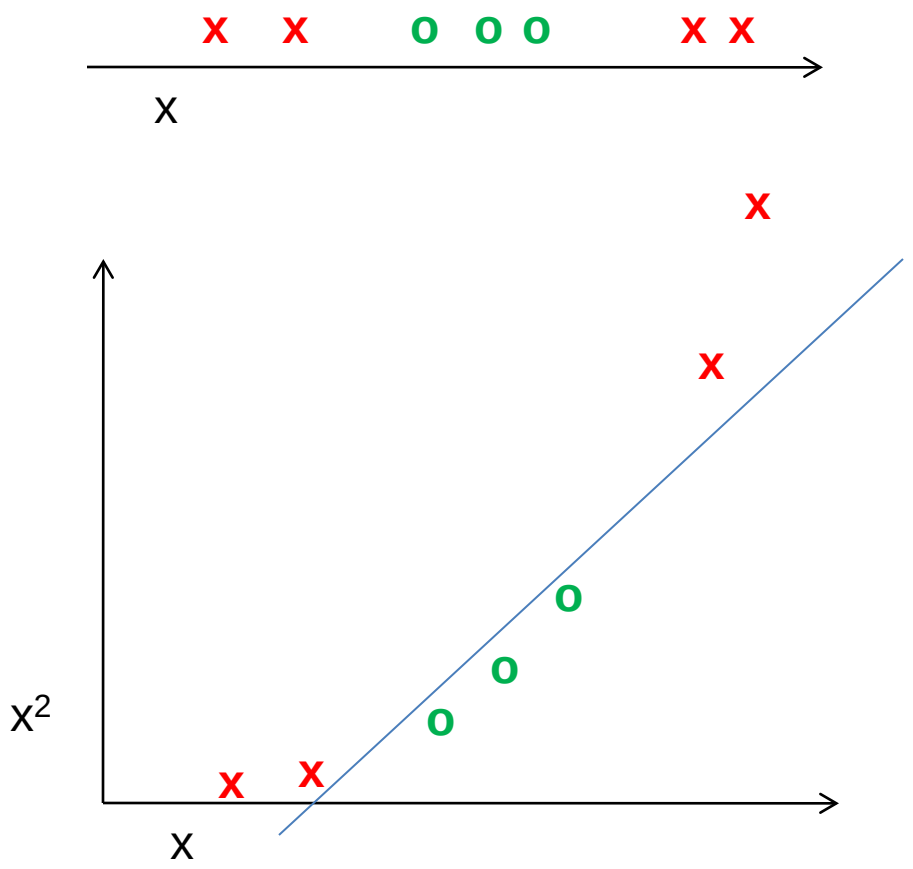


Classifiers: Linear SVM

- Objective
- Parameterization
- Regularization
- Training
- Inference



Classifiers: Kernelized SVM

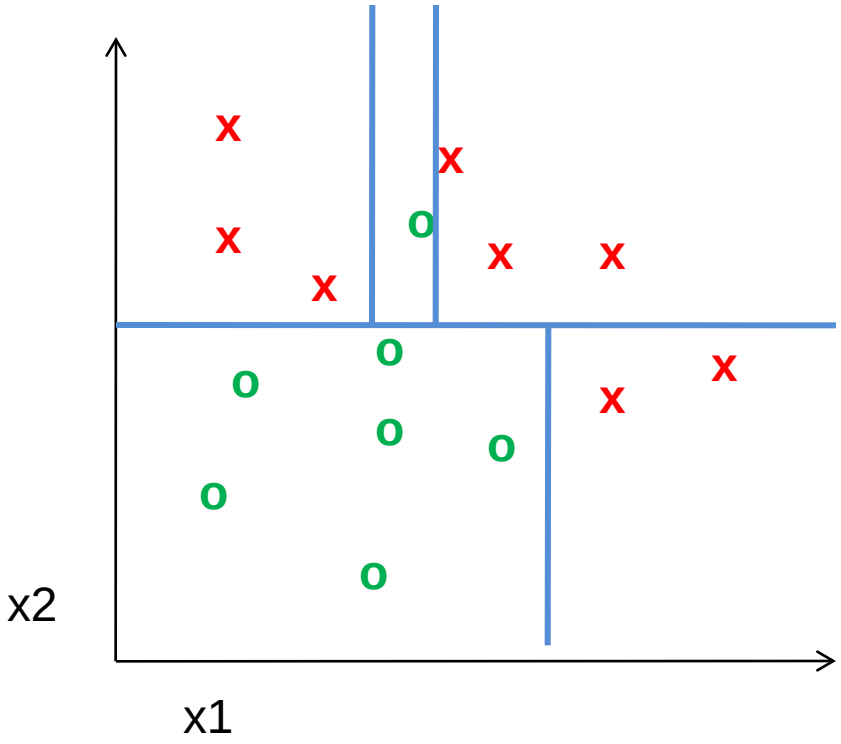


Using SVMs

- Good general purpose classifier
  - Generalization depends on margin, so works well with many weak features
  - No feature selection
  - Usually requires some parameter tuning

- Choosing kernel
  - Linear: fast training/testing – start here
  - RBF: related to neural networks, nearest neighbor
  - Chi-squared, histogram intersection: good for histograms (but slower, esp. chi-squared)
  - Can learn a kernel function

Classifiers: Decision Trees



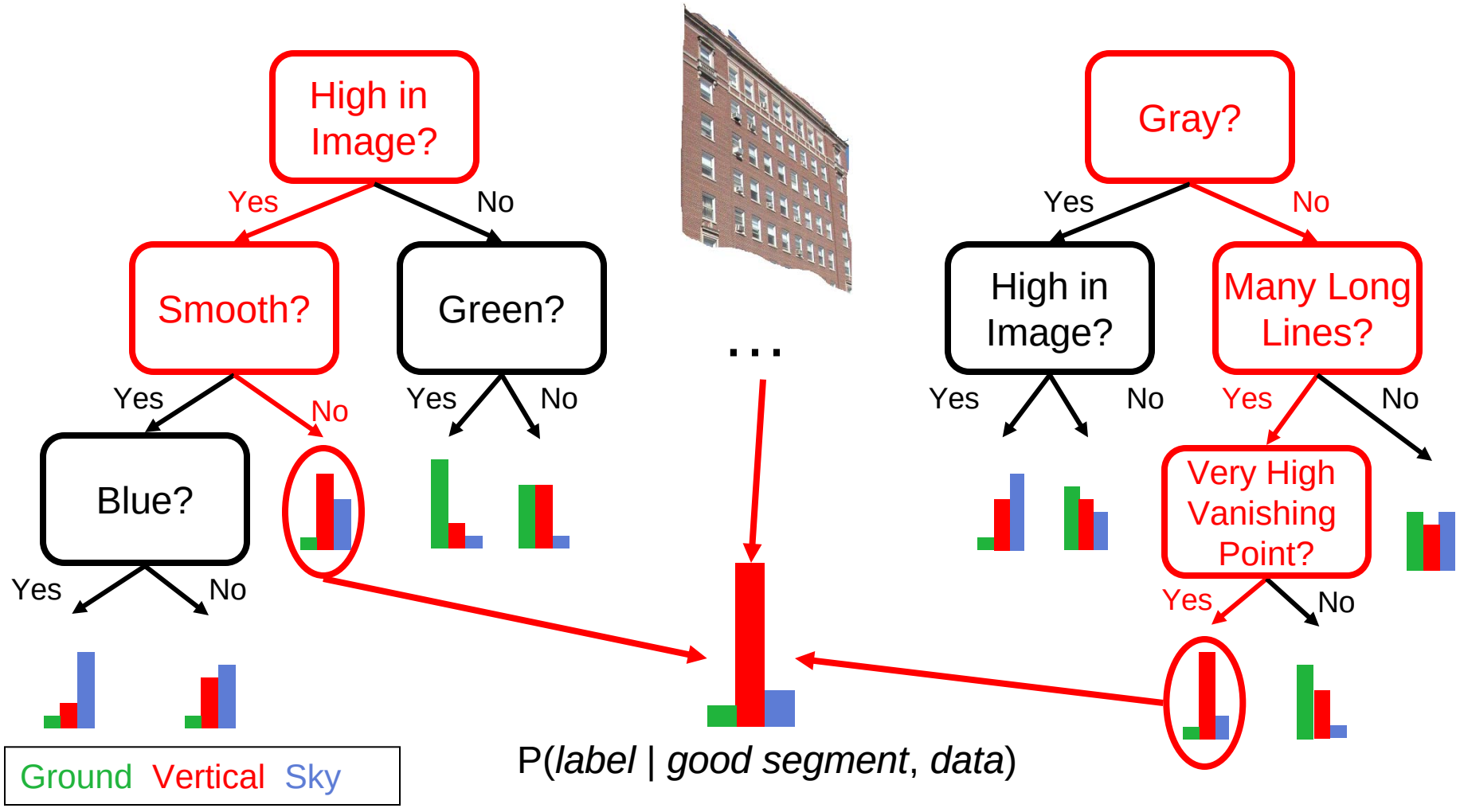
Ensemble Methods: Boosting

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 $\text{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (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 $\text{sign}[\sum_{m=1}^{M} c_m f_m(x)]$

figure from Friedman et al. 2000

Boosted Decision Trees

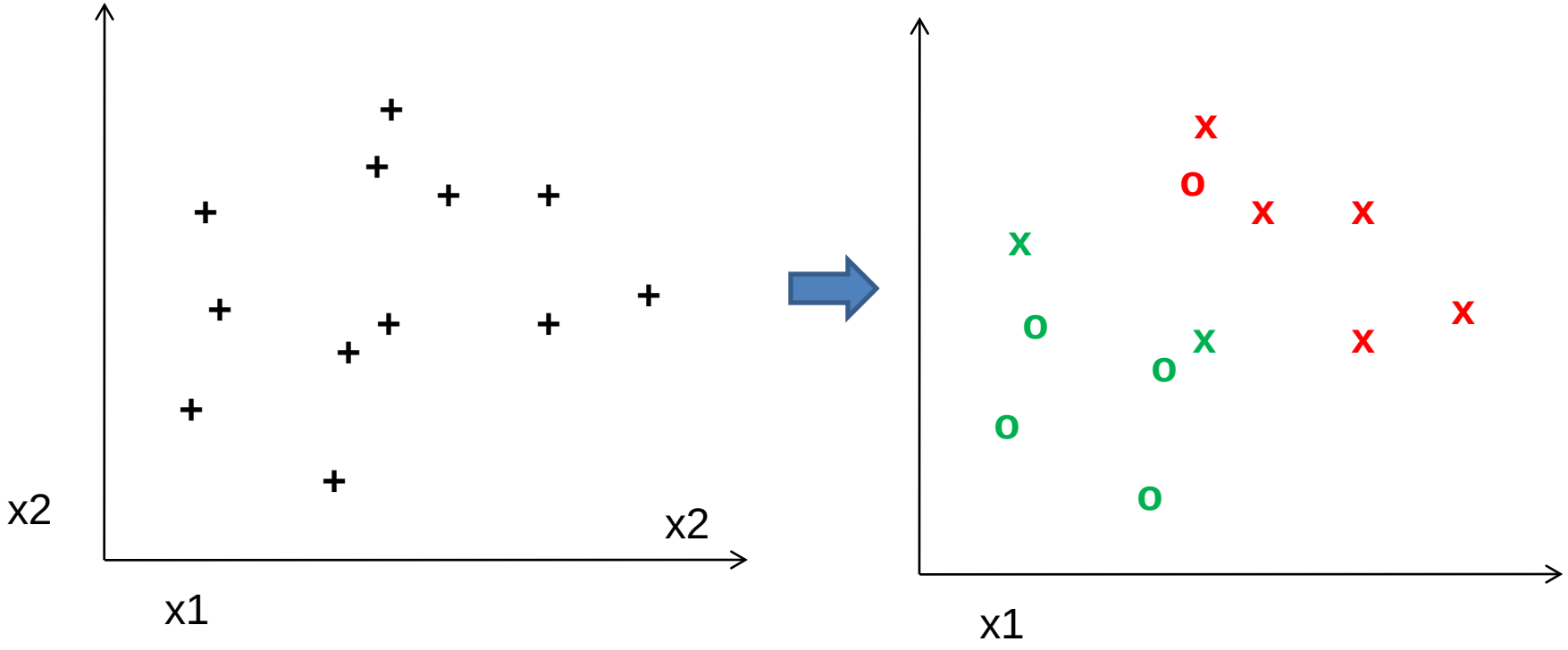


[Collins et al. 2002]

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

Clustering (unsupervised)



Two ways to think about classifiers

1. What is the objective? What are the parameters? How are the parameters learned? How is the learning regularized? How is inference performed?

2. How is the data modeled? How is similarity defined? What is the shape of the boundary?

Comparison

assuming x in {0 1}

	Learning Objective	Training	Inference
Naïve Bayes	$\text{maximize} \sum_i \left[\sum_j \log P(x_{ij} \mid y_i; \theta_j) + \log P(y_i; \theta_0) \right]$	$\theta_{kj} = \frac{\sum_i \delta(x_{ij} = 1 \wedge y_i = k) + r}{\sum_i \delta(y_i = k) + Kr}$	$\theta_1^T \mathbf{x} + \theta_0^T (1 - \mathbf{x}) > 0$ 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 $i = \underset{i}{\text{argmin}} \; K(\hat{\mathbf{x}}_i, \mathbf{x})$

What to remember about classifiers

- No free lunch: 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)

Next class

- Object category detection overview

Some Machine Learning References

- General
  - Tom Mitchell, *Machine Learning*, McGraw Hill, 1997
  - Christopher Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995
- Adaboost
  - Friedman, Hastie, and Tibshirani, “Additive logistic regression: a statistical view of boosting”, Annals of Statistics, 2000
- SVMs
  - http://www.support-vector.net/icml-tutorial.pdf