Classifiers

Computer Vision
CS 543 / ECE 549
University of Illinois

Derek Hoiem

Today’s class

- Review of image categorization

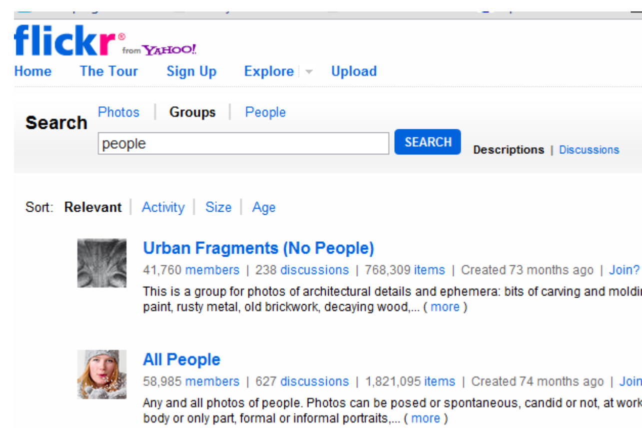
- Classification
  - A few examples of classifiers: nearest neighbor, generative classifiers, logistic regression, SVM
  - Important concepts in machine learning
  - Practical tips

- 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



Examples of Categorization in Vision

- Part or object detection
  - E.g., for each window: face or non-face?
- Scene categorization
  - Indoor vs. outdoor, urban, forest, kitchen, etc.
- Action recognition
  - Picking up vs. sitting down vs. standing …
- Emotion recognition
- Region classification
  - Label pixels into different object/surface categories
- Boundary classification
  - Boundary vs. non-boundary
- Etc, etc.

Image Categorization

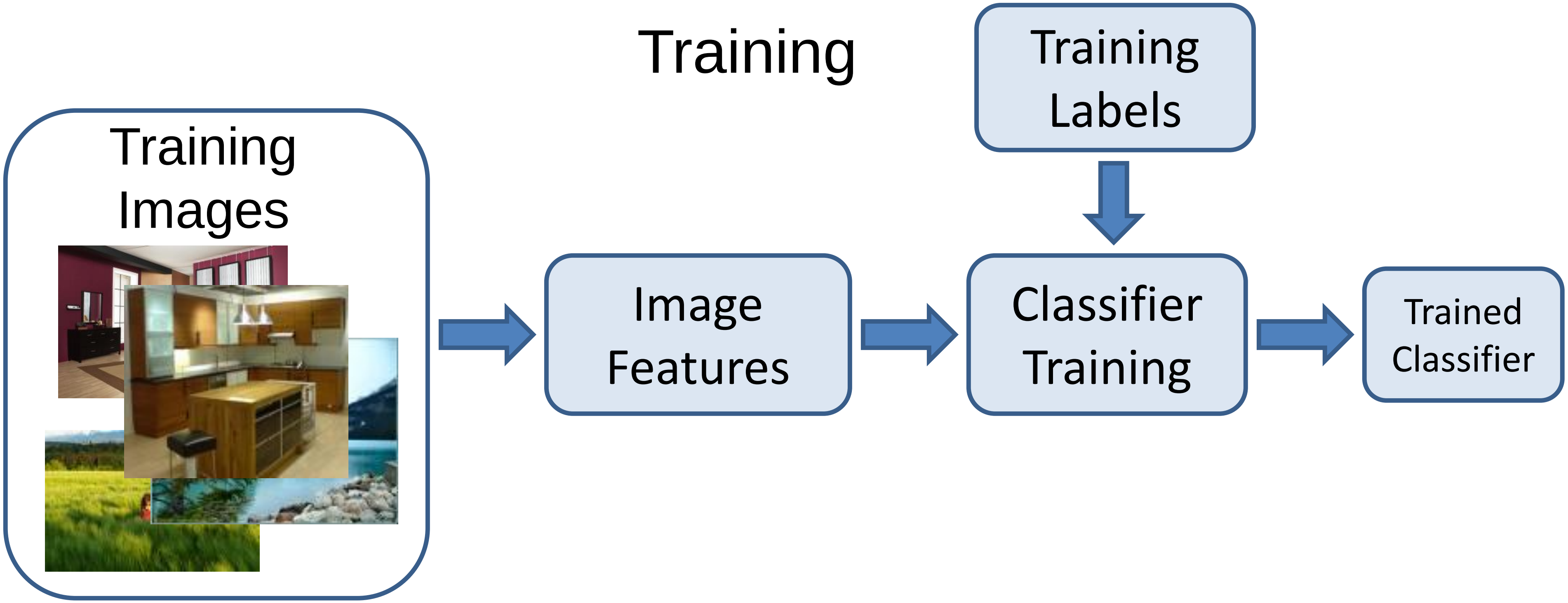
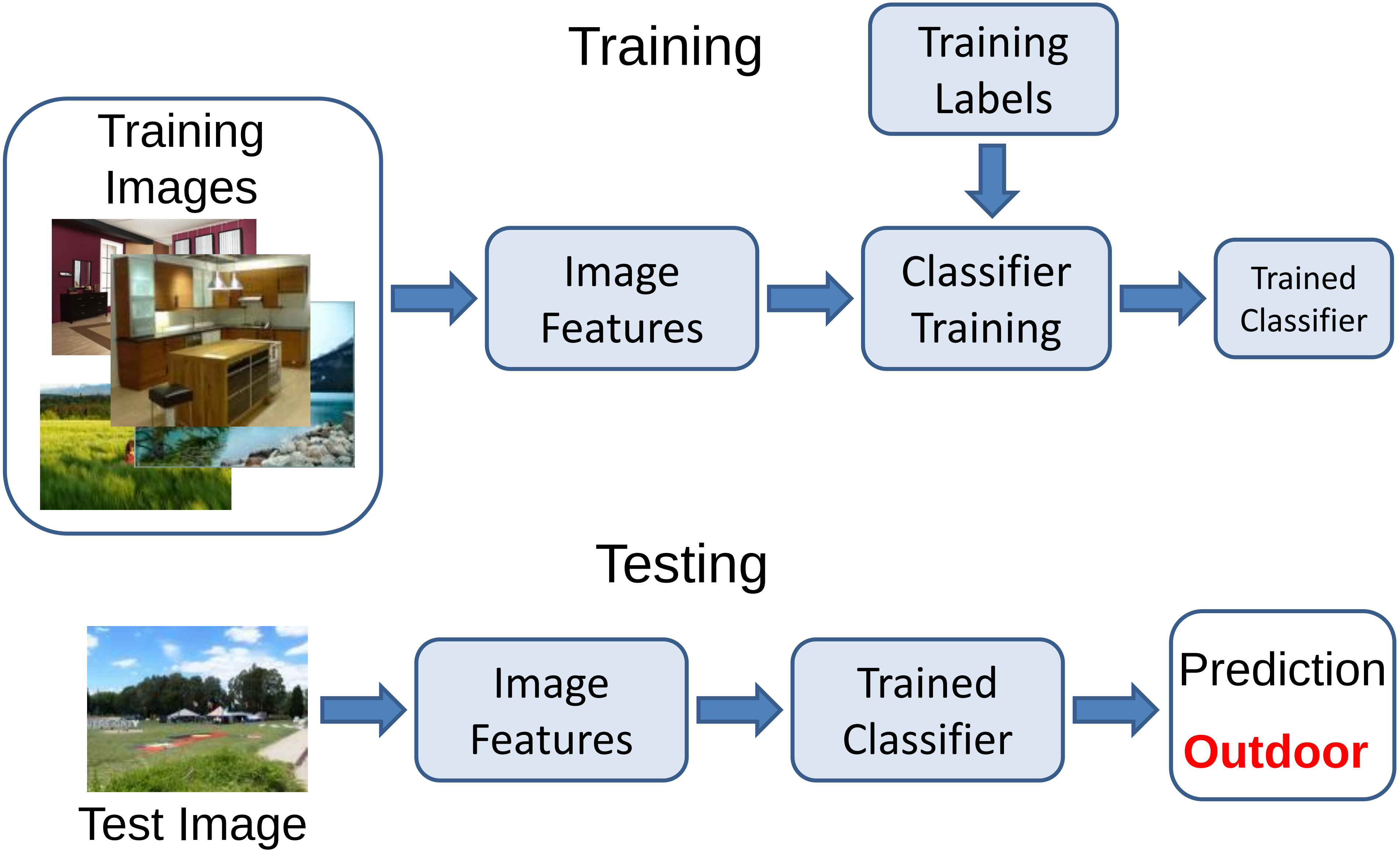


Image Categorization

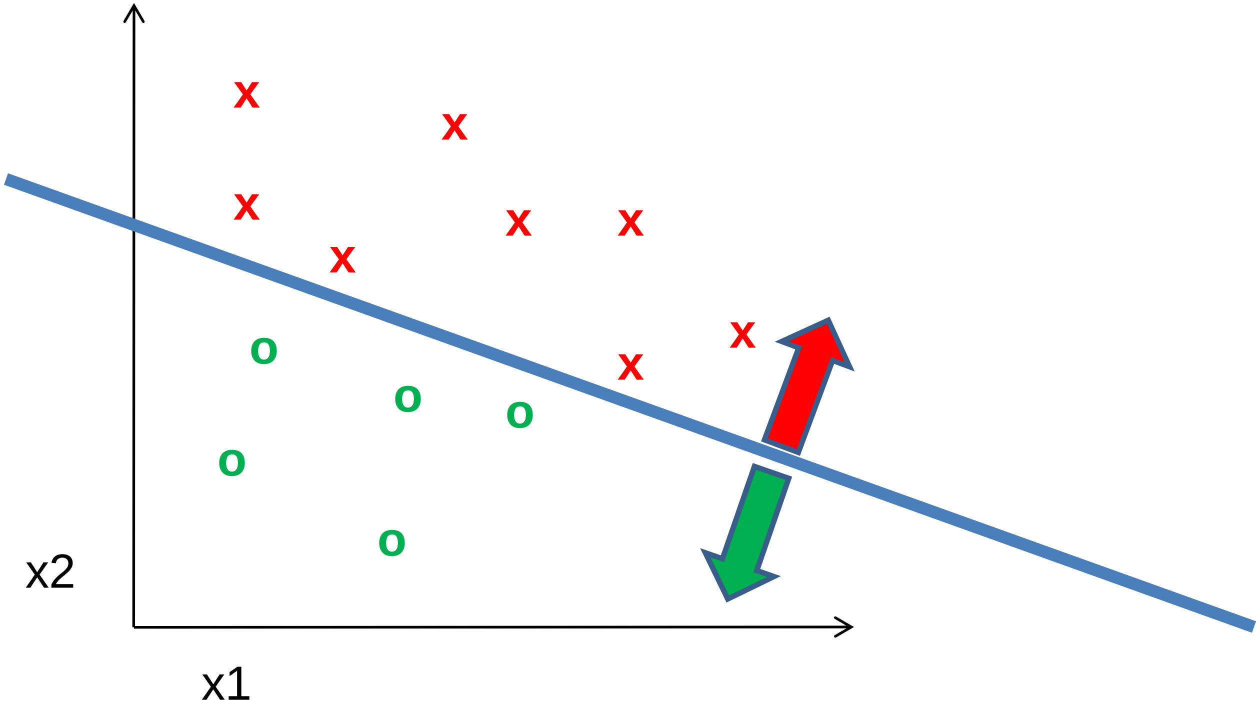


Feature design is paramount

- Most features can be thought of as templates, histograms (counts), or combinations
- Think about the right features for the problem
  - Coverage
  - Concision
  - Directness

Classifier

A classifier maps from the feature space to a label



Different types of classification

- Exemplar-based: transfer category labels from examples with most similar features
  - What similarity function? What parameters?
- Linear classifier: confidence in positive label is a weighted sum of features
  - What are the weights?
- Non-linear classifier: predictions based on more complex function of features
  - What form does the classifier take? Parameters?
- Generative classifier: assign to the label that best explains the features (makes features most likely)
  - What is the probability function and its parameters?

Note: You can always fully design the classifier by hand, but usually this is too difficult. Typical solution: learn from training examples.

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

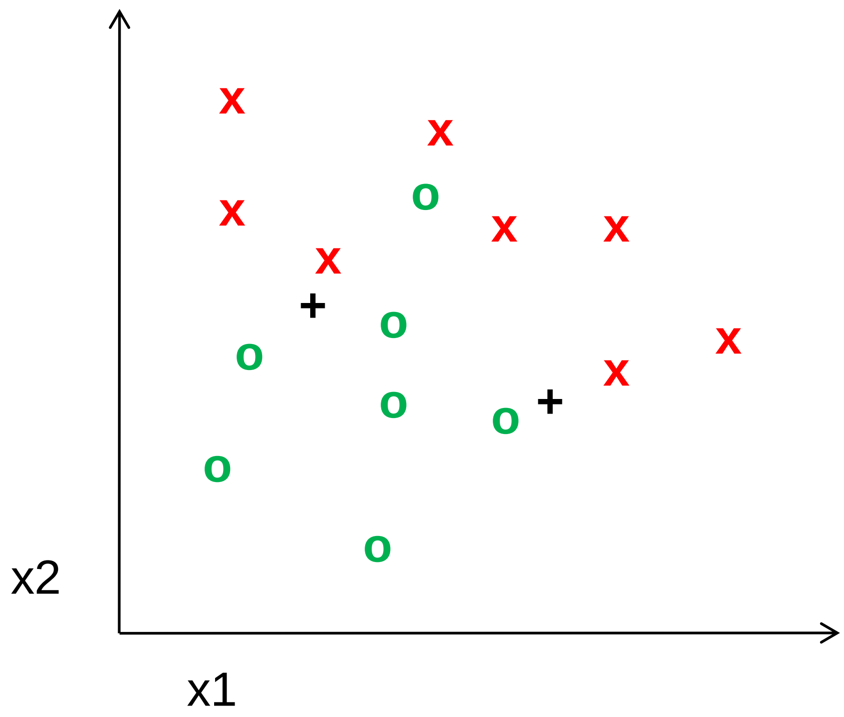
- Goal of training is to learn feature weights or distance measures so that visual similarity predicts label similarity

- *We want the simplest function that is confidently correct*

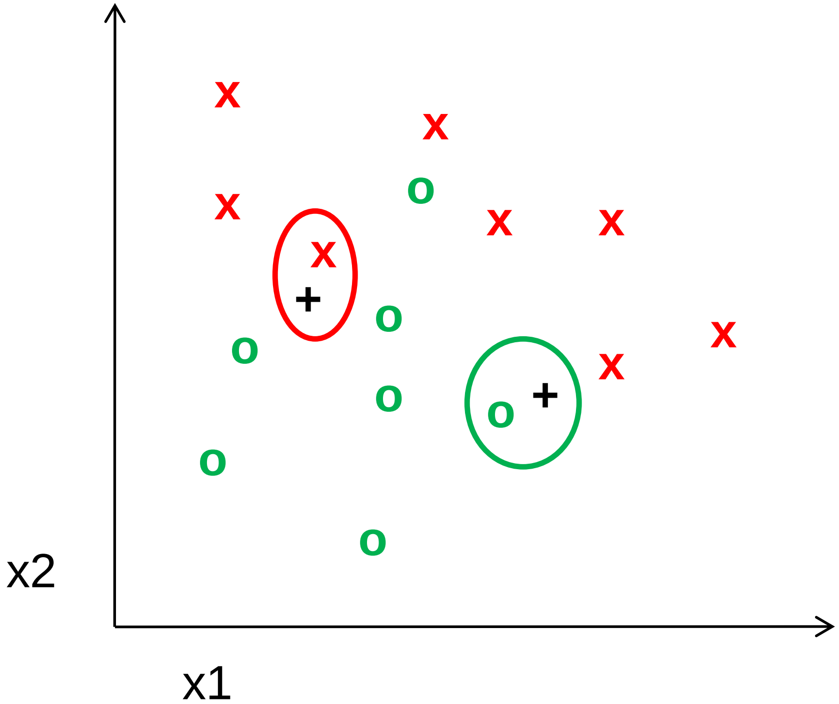
Exemplar-based Models

- Transfer the label(s) of the most similar training examples

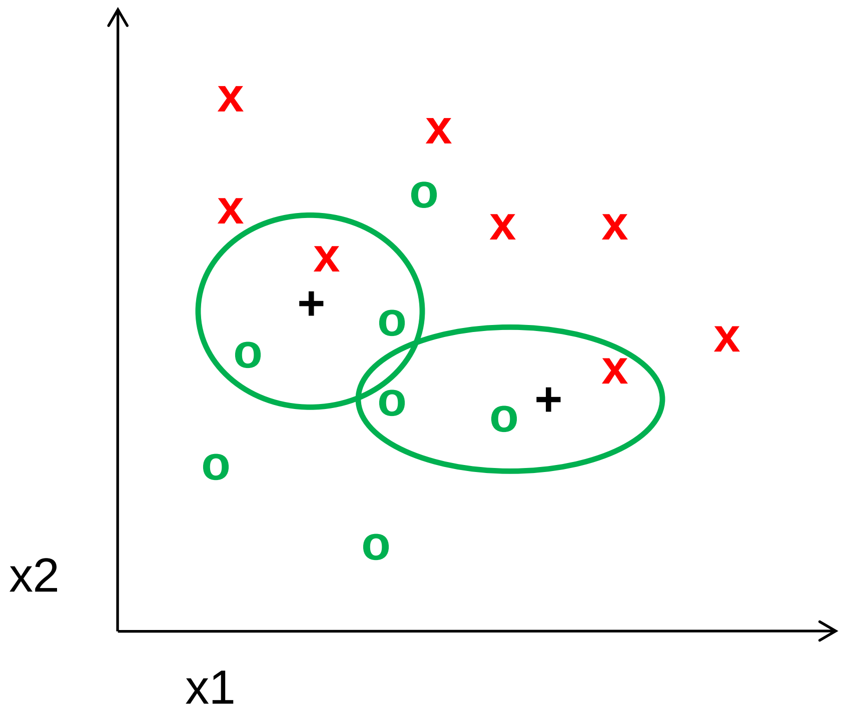
K-nearest neighbor classifier



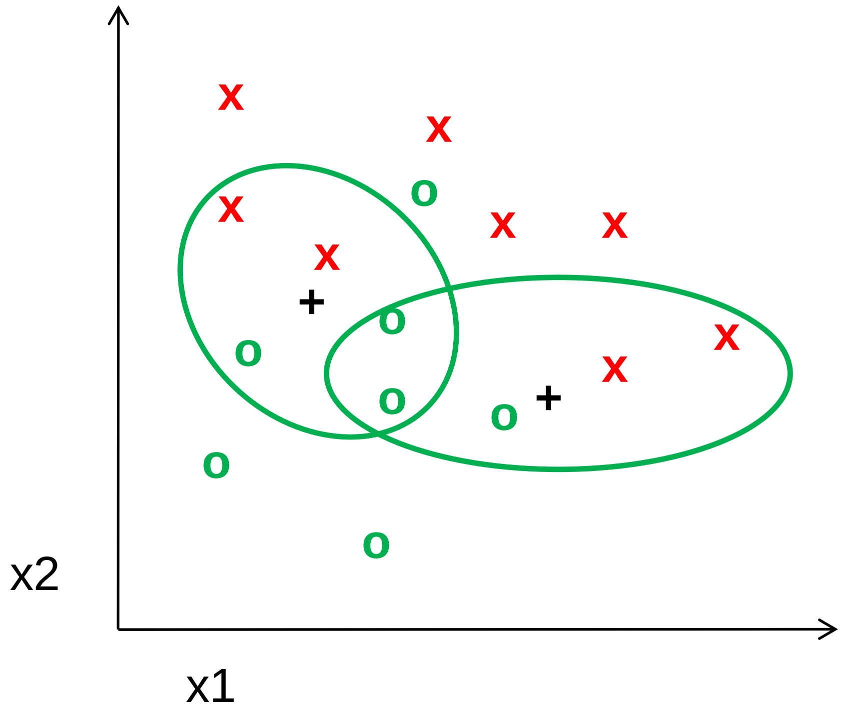
1-nearest neighbor



3-nearest neighbor



5-nearest neighbor



Using K-NN

- Simple, a good one to try first
- Higher K gives smoother functions
- No training time (unless you want to learn a distance function)
- With infinite examples, 1-NN provably has error that is at most twice Bayes optimal error

Discriminative classifiers

Learn a simple function of the input features that confidently predicts the true labels on the training set

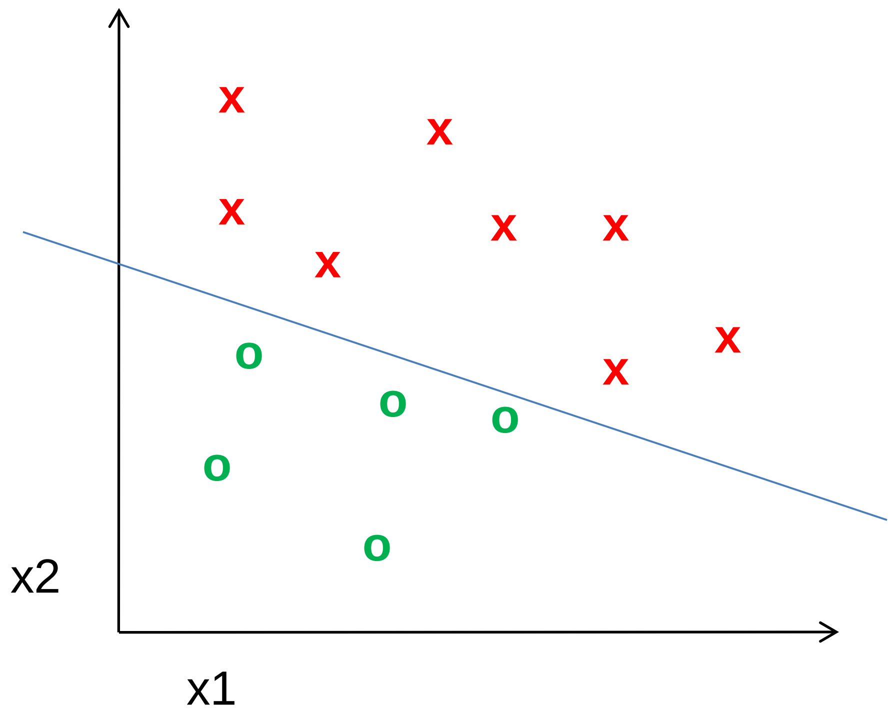
$$y = f(x)$$

Training Goals

1. Accurate classification of training data
2. Correct classifications are confident
3. Classification function is simple

Classifiers: Logistic Regression

- Objective
- Parameterization
- Regularization
- Training
- Inference

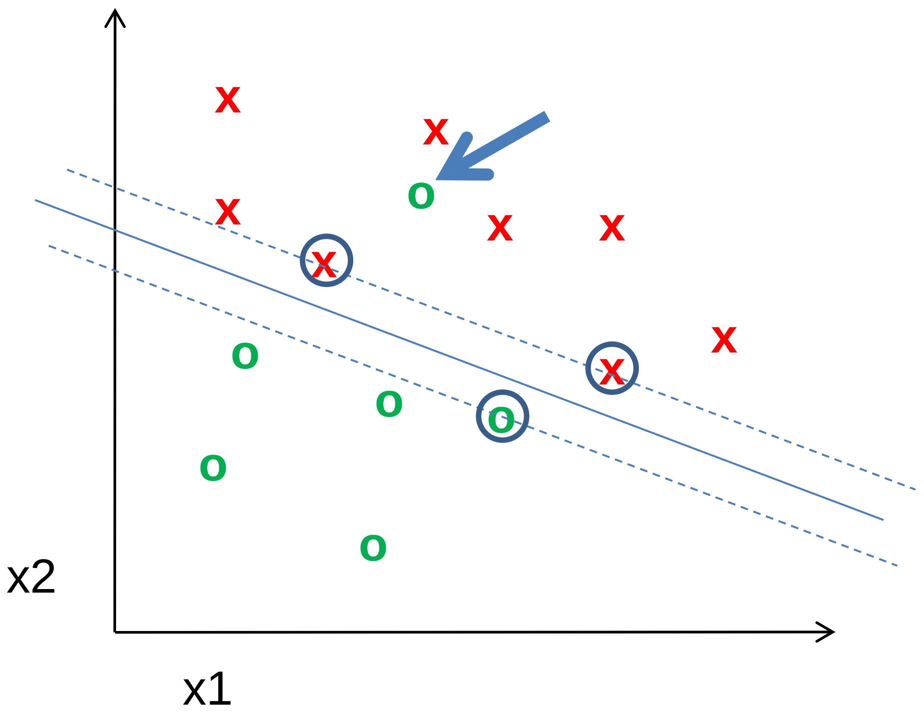


The **objective function** of most discriminative classifiers includes a **loss term** and a **regularization term**.

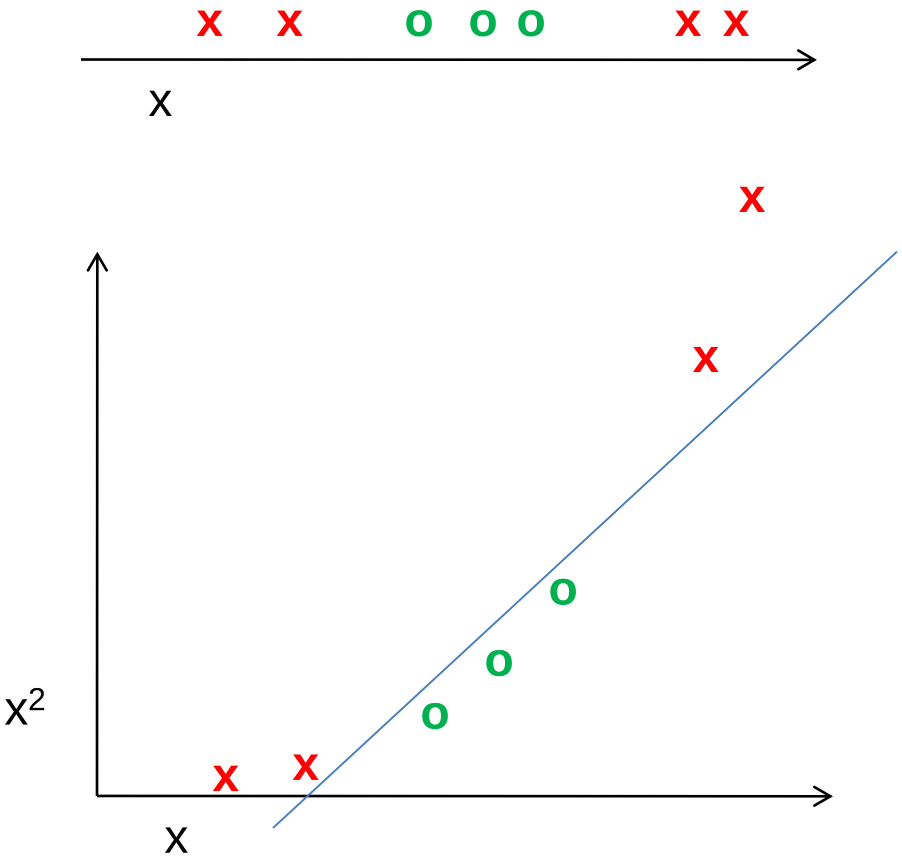
Using Logistic Regression

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

Classifiers: Linear SVM



Classifiers: Kernelized SVM

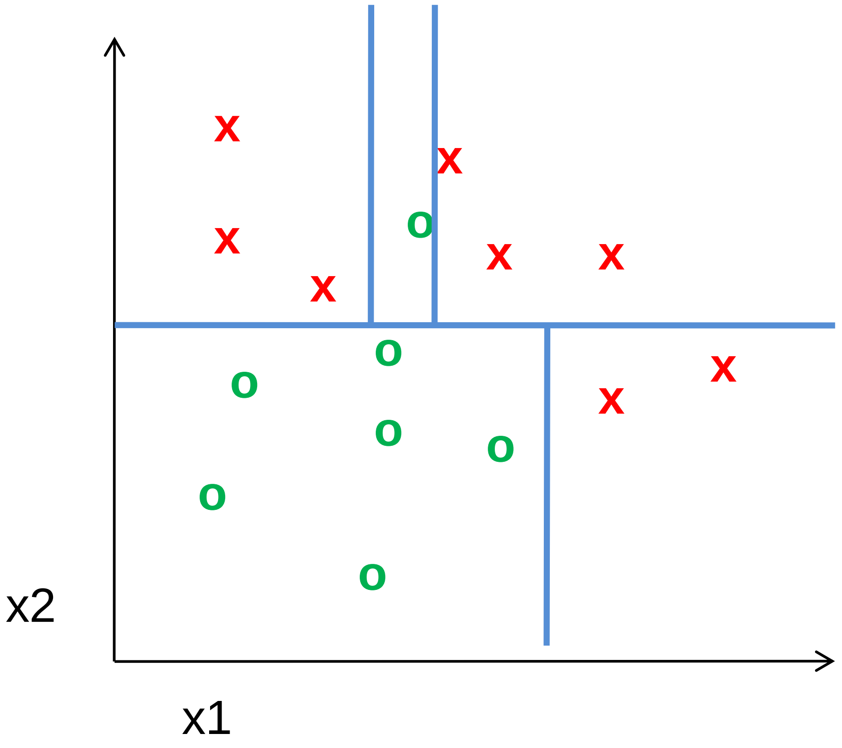


Using SVMs

- 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



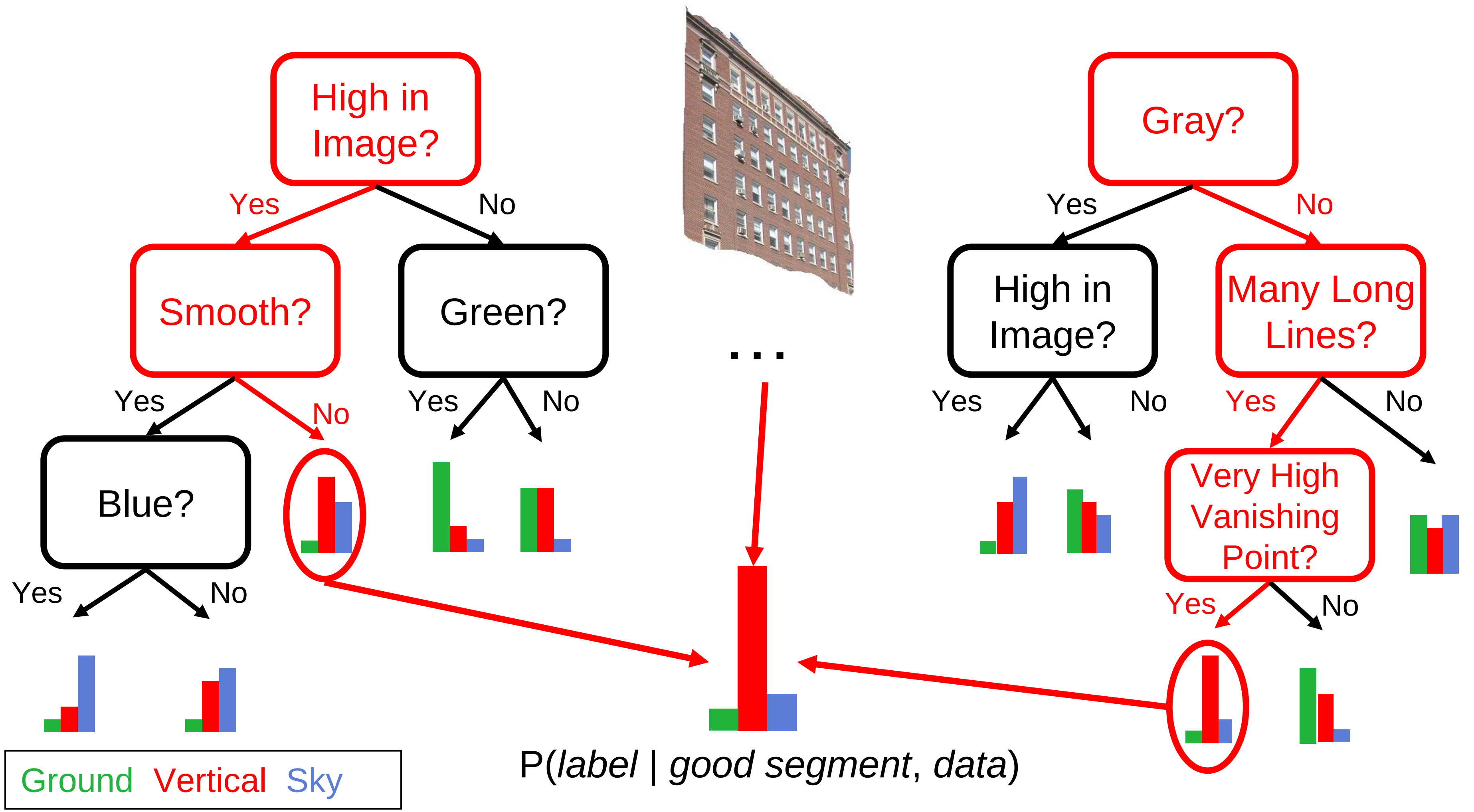
Ensemble Methods: Boosting

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

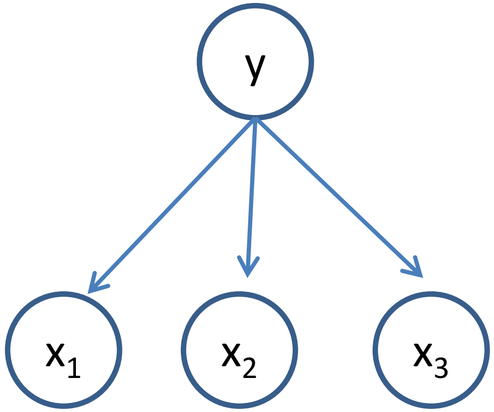
- Flexible: can deal with both continuous and categorical variables
- How to control bias/variance trade-off
  - Size of trees
  - Number of trees
- Boosting trees often works best with a small number of well-designed features
- Boosting “stubs” can give a fast classifier

Generative classifiers

- Model the joint probability of the features and the labels
  - Allows direct control of independence assumptions
  - Can incorporate priors
  - Often simple to train (depending on the model)
- Examples
  - Naïve Bayes
  - Mixture of Gaussians for each class

Naïve Bayes

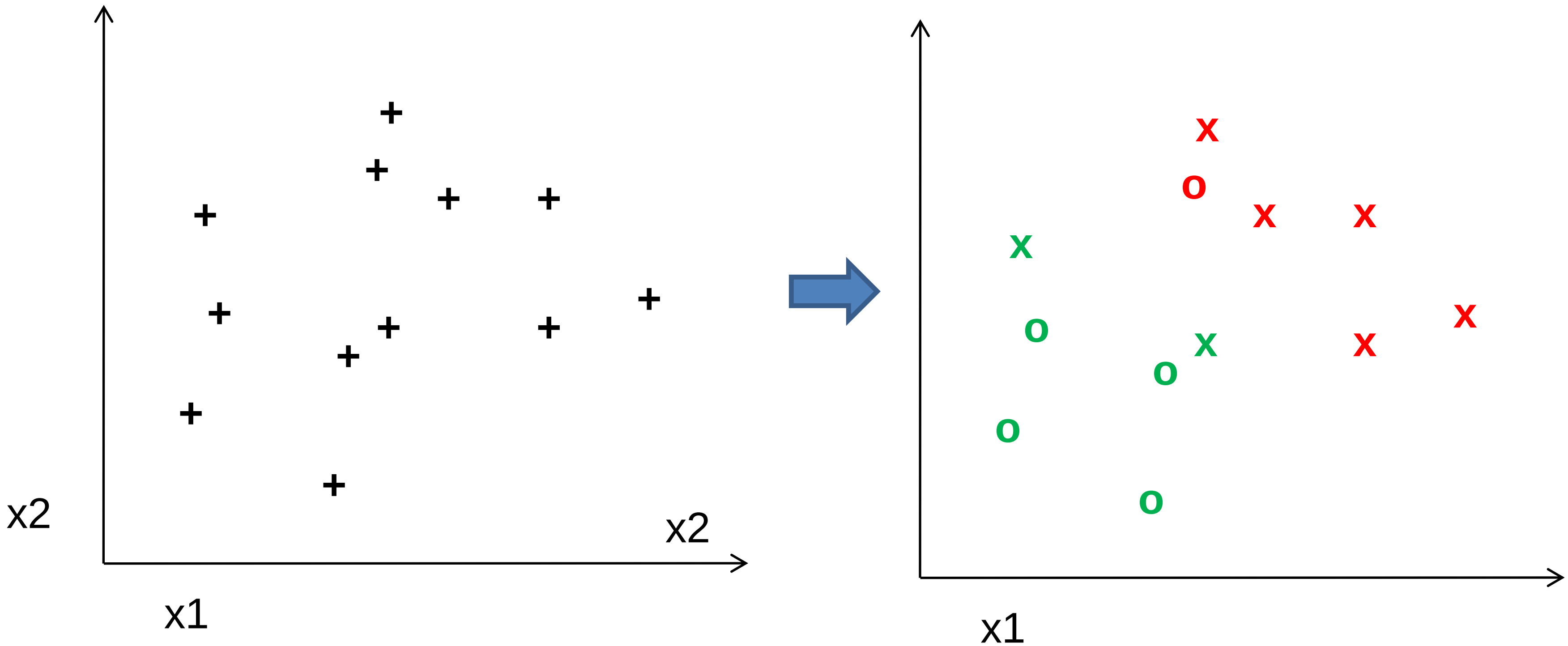
- Objective
- Parameterization
- Regularization
- Training
- Inference



Using Naïve Bayes

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

Clustering (unsupervised)



Many classifiers to choose from

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

Which is the best one?

No Free Lunch Theorem

Generalization Theory

- It’s not enough to do well on the training set: we want to also make good predictions for new examples

Bias-Variance Trade-off

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

Unavoidable error

Error due to incorrect assumptions

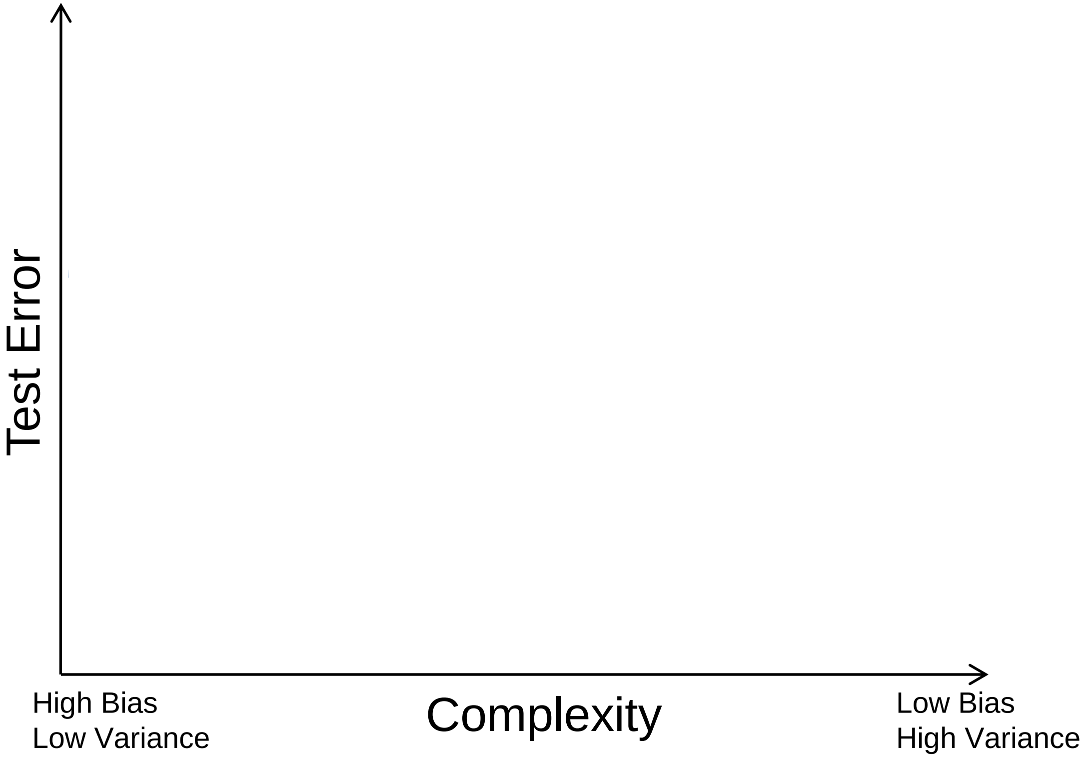
Error due to variance parameter estimates from training samples

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

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

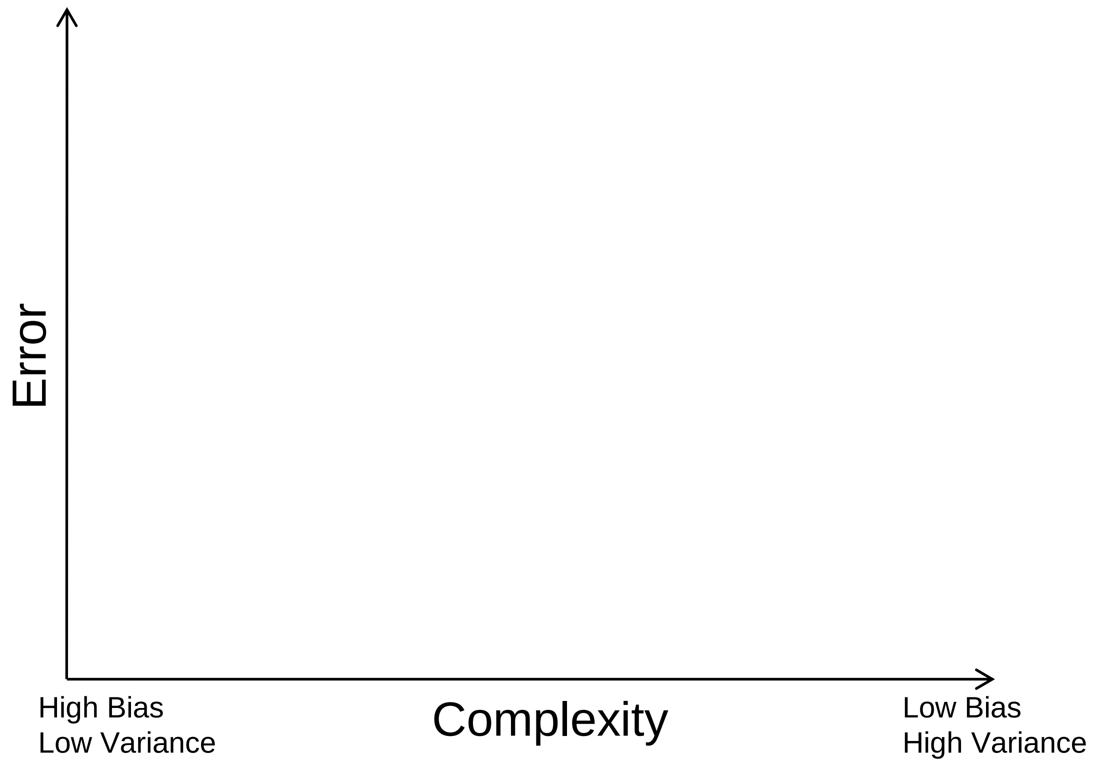
Bias and Variance

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

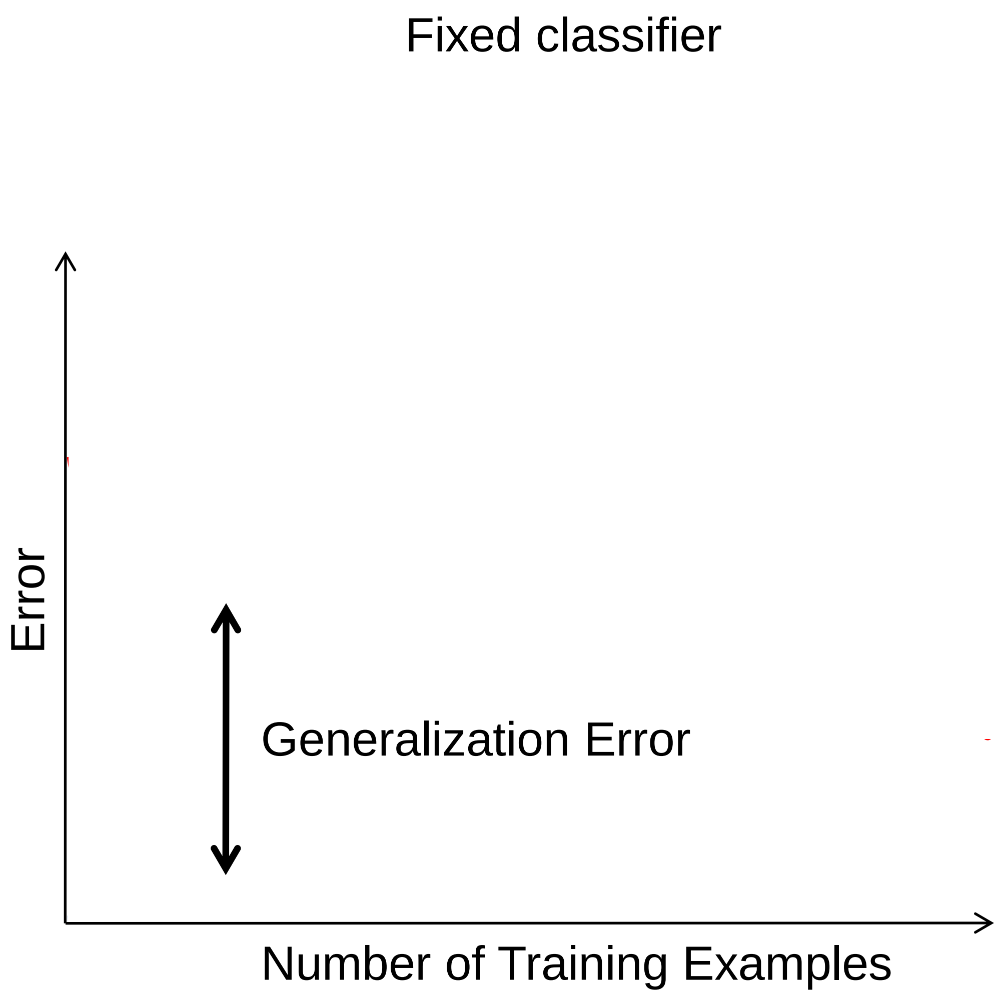


Choosing the trade-off

- Need validation set
- Validation set is separate from the 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

Test 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.

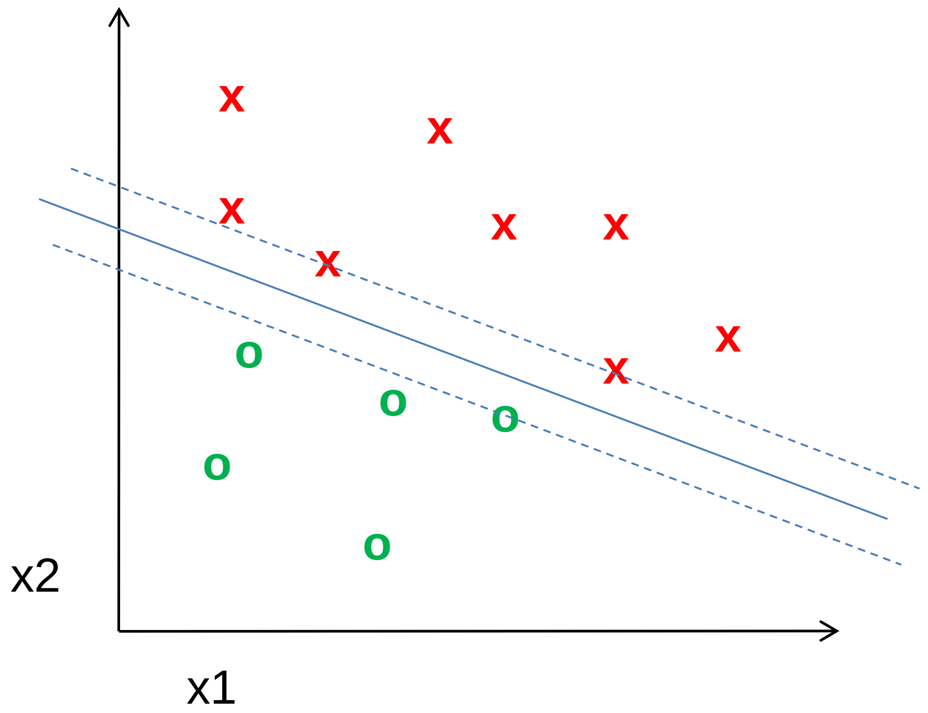
How to reduce variance?

- Choose a simpler classifier
- Regularize the parameters
- Use fewer features
- 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

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	$\boldsymbol{\theta}^T \mathbf{x} > t$
Linear SVM	$\text{minimize } \lambda \sum_i \xi_i + \frac{1}{2} \lVert \boldsymbol{\theta} \rVert$ such that $y_i \boldsymbol{\theta}^T \mathbf{x} \geq 1 - \xi_i \quad \forall i, \ \xi_i \geq 0$	Quadratic programming or subgradient opt.	$\boldsymbol{\theta}^T \mathbf{x} > t$
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 $i = \underset{i}{\operatorname{argmin}} \ K(\hat{\mathbf{x}}_i, \mathbf{x})$

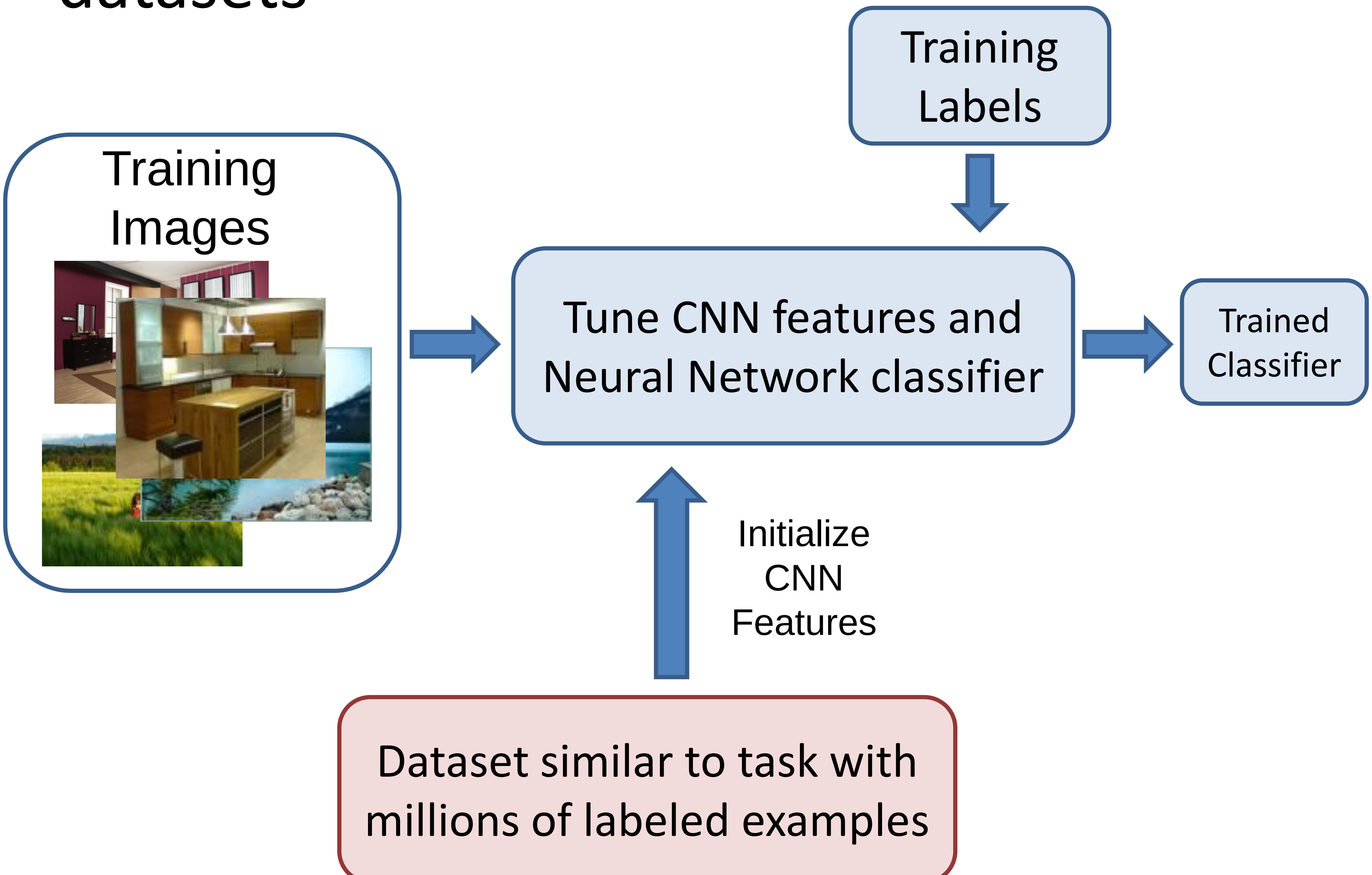
Characteristics of vision learning problems

- Lots of continuous features
  - E.g., HOG template may have 1000 features
  - Spatial pyramid may have ~15,000 features
- Imbalanced classes
  - often limited positive examples, practically infinite negative examples
- Difficult prediction tasks

When a massive training set is available

- Relatively new phenomenon
  - MNIST (handwritten letters) in 1990s, LabelMe in 2000s, ImageNet (object images) in 2009, …
- Want classifiers with low bias (high variance ok) and reasonably efficient training
- Very complex classifiers with simple features are often effective
  - Random forests
  - Deep convolutional networks

New training setup with moderate sized datasets



Practical tips

- Preparing features for linear classifiers
  - Often helps to make zero-mean, unit-dev
  - For non-ordinal features, convert to a set of binary features
- Selecting classifier meta-parameters (e.g., regularization weight)
  - Cross-validation: split data into subsets; train on all but one subset, test on remaining; repeat holding out each subset
    - Leave-one-out, 5-fold, etc.
- Most popular classifiers in vision
  - *SVM*: linear for when fast training/classification is needed; performs well with lots of weak features
  - *Logistic Regression*: outputs a probability; easy to train and apply
  - *Nearest neighbor*: hard to beat if there is tons of data (e.g., character recognition)
  - *Boosted stumps or decision trees*: applies to flexible features, incorporates feature selection, powerful classifiers
  - *Random forests*: outputs probability; good for simple features, tons of data
  - *Deep networks / CNNs*: flexible output; learns features; adapt existing network (which is trained with tons of data) or train new with tons of data
- Always try at least two types of classifiers

What to remember about classifiers

- No free lunch: machine learning algorithms are tools
- Try simple classifiers first
- Better to have smart features and simple classifiers than simple features and smart classifiers
  - Though with enough data, smart features can be learned
- Use increasingly powerful classifiers with more training data (bias-variance tradeoff)

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
- Random forests
  - http://research.microsoft.com/pubs/155552/decisionForests_MSR_TR_2011_114.pdf

Next class

- Detection using sliding windows and region proposals