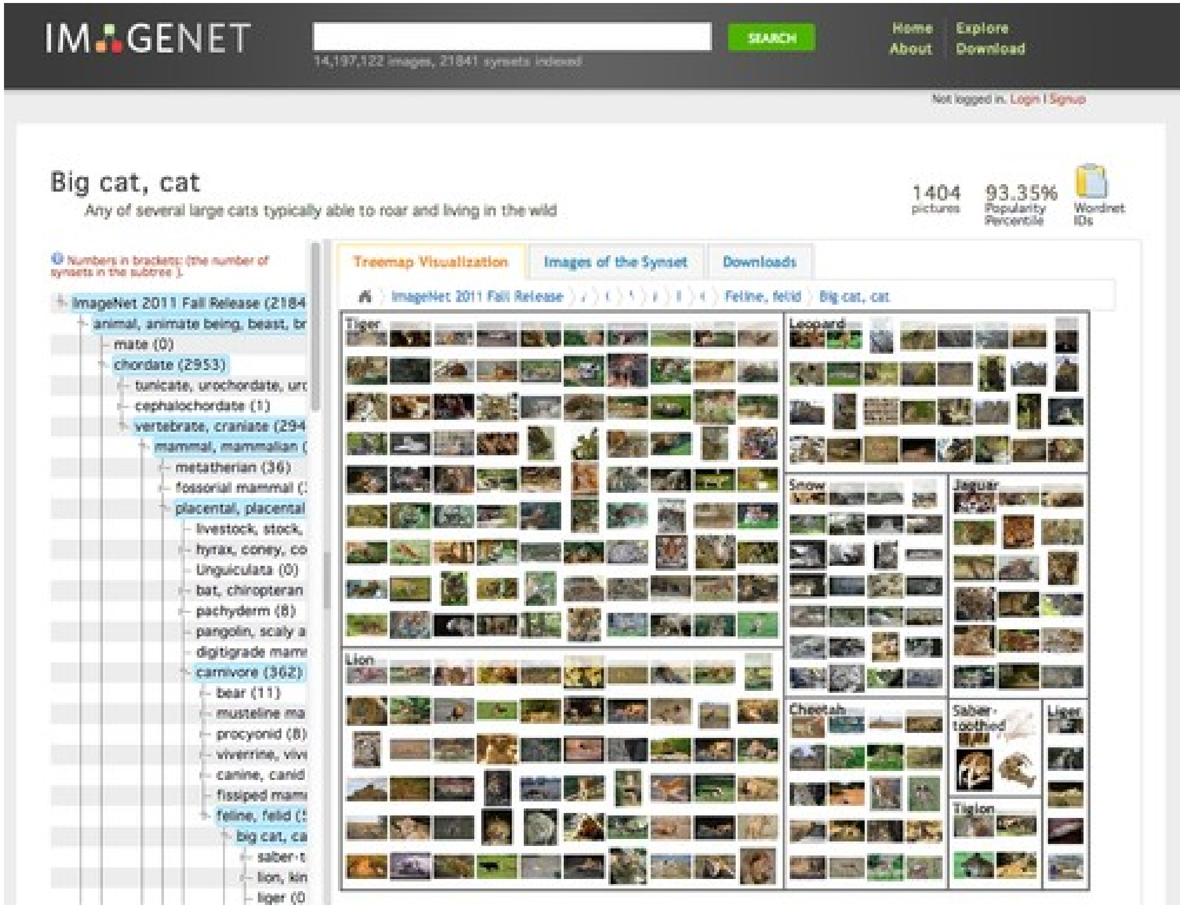
Advanced Image Processing

ML: Supervised Learning

SYDE 671

ImageNet

- Images for each category of WordNet
- 1000 classes
- 1.2mil images
- 100k test



- Top 5 error

Dataset split

Training Images

Validation Images

Testing Images

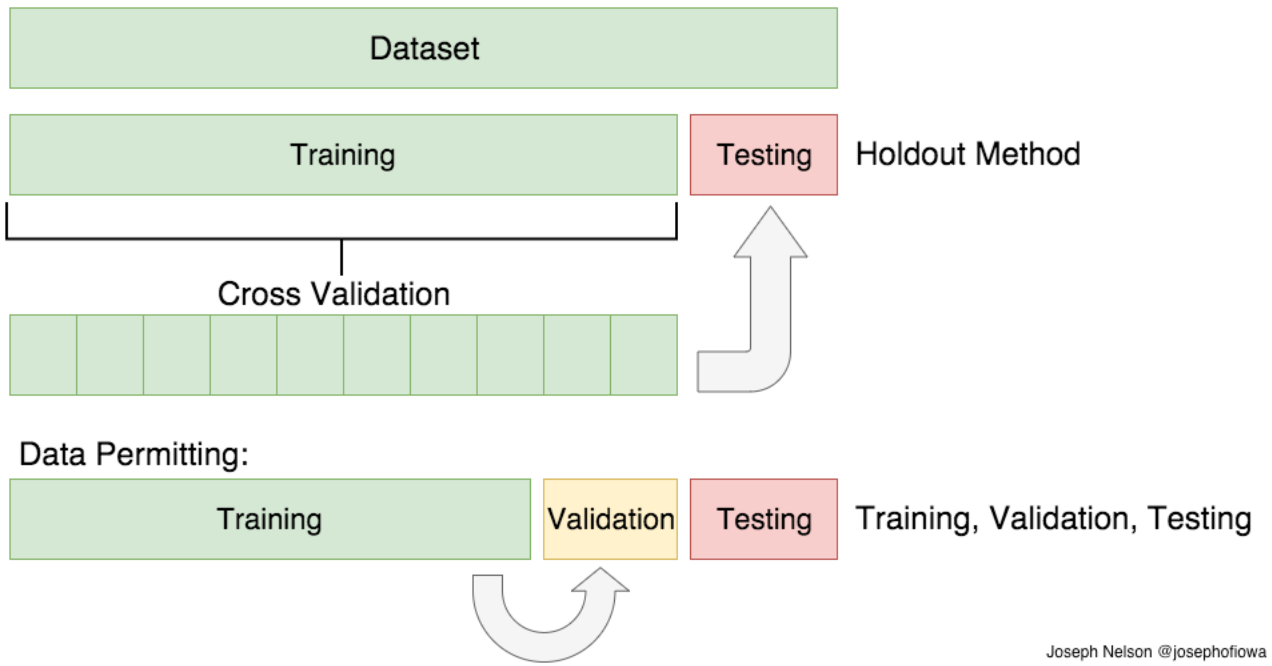
- Train classifier

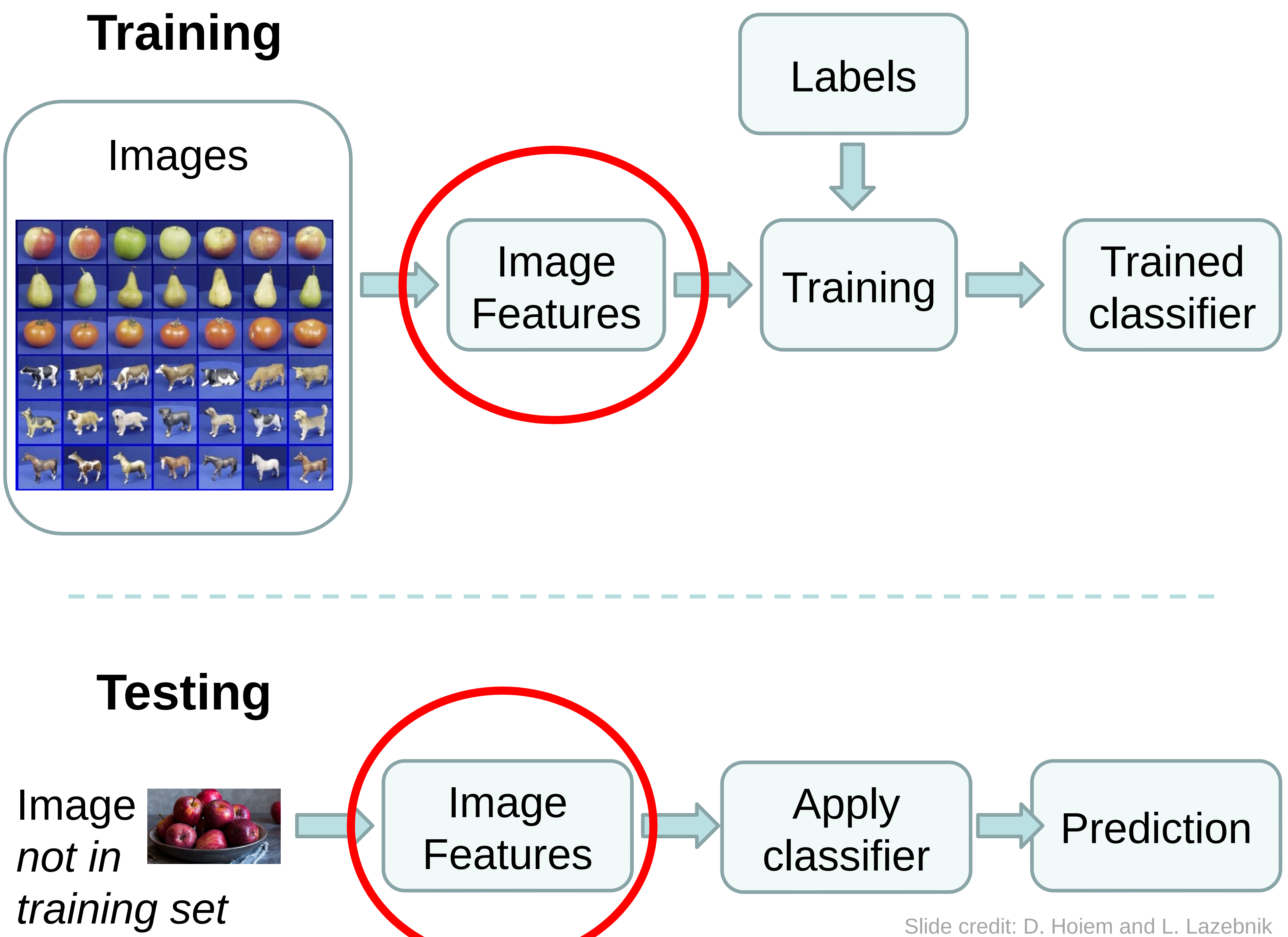
- Measure error
- Tune model hyperparameters

- Secret labels
- Measure error

Cycle through different train/validate splits = cross validation

Dataset split (Cross Validation)





Slide credit: D. Hoiem and L. Lazebnik

Features

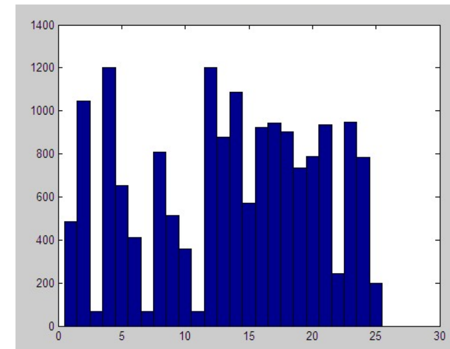
Raw pixels

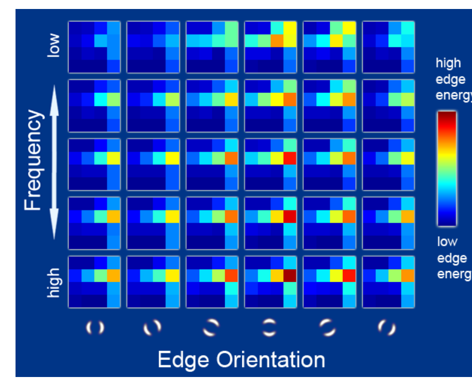
Histograms

Templates

Descriptors

- GIST
- SIFT
- ORB
- HOG….



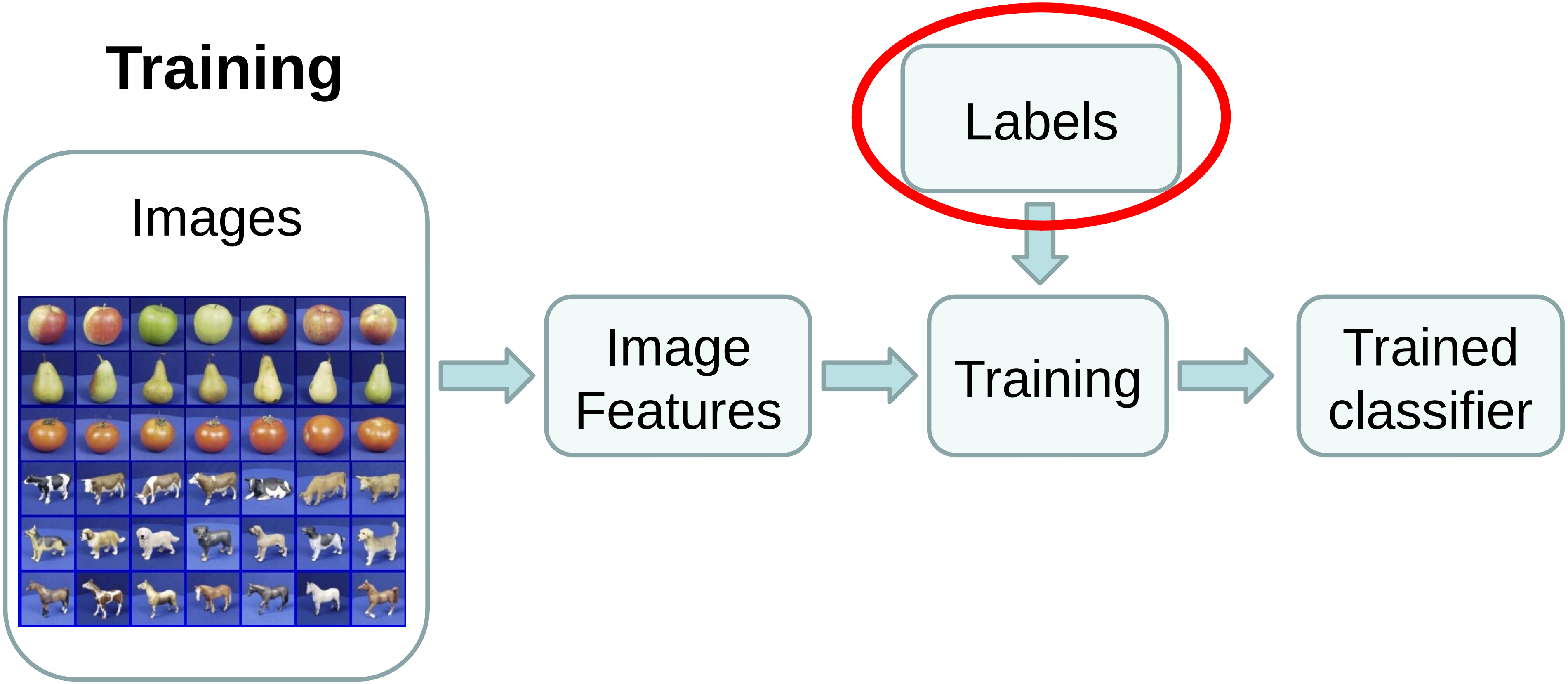


L. Lazebnik

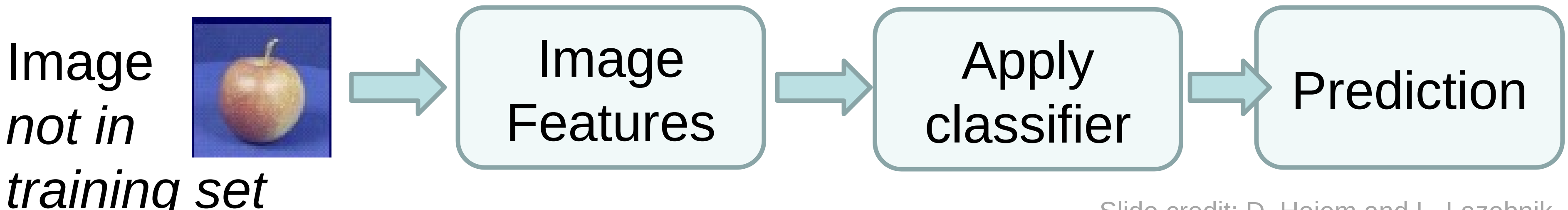
General Principles of Representation

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

Training



Testing



Slide credit: D. Hoiem and L. Lazebnik

Recognition task and supervision

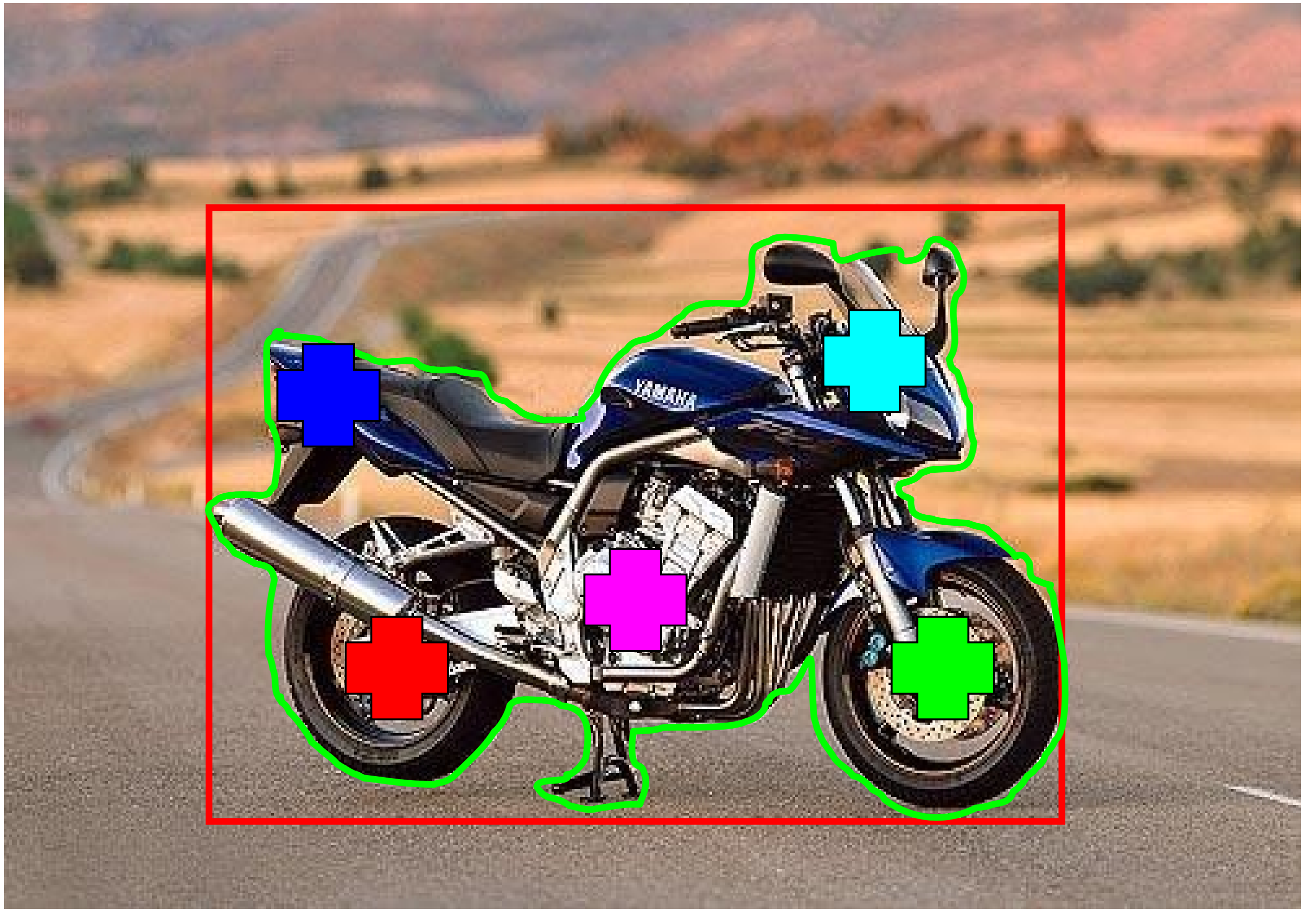
What are all the possible supervision ('label') *types* to consider?

L. Lazebnik

Recognition task and supervision

Images in the training set must be annotated with the "correct answer" that the model is expected to produce

Contains a motorbike

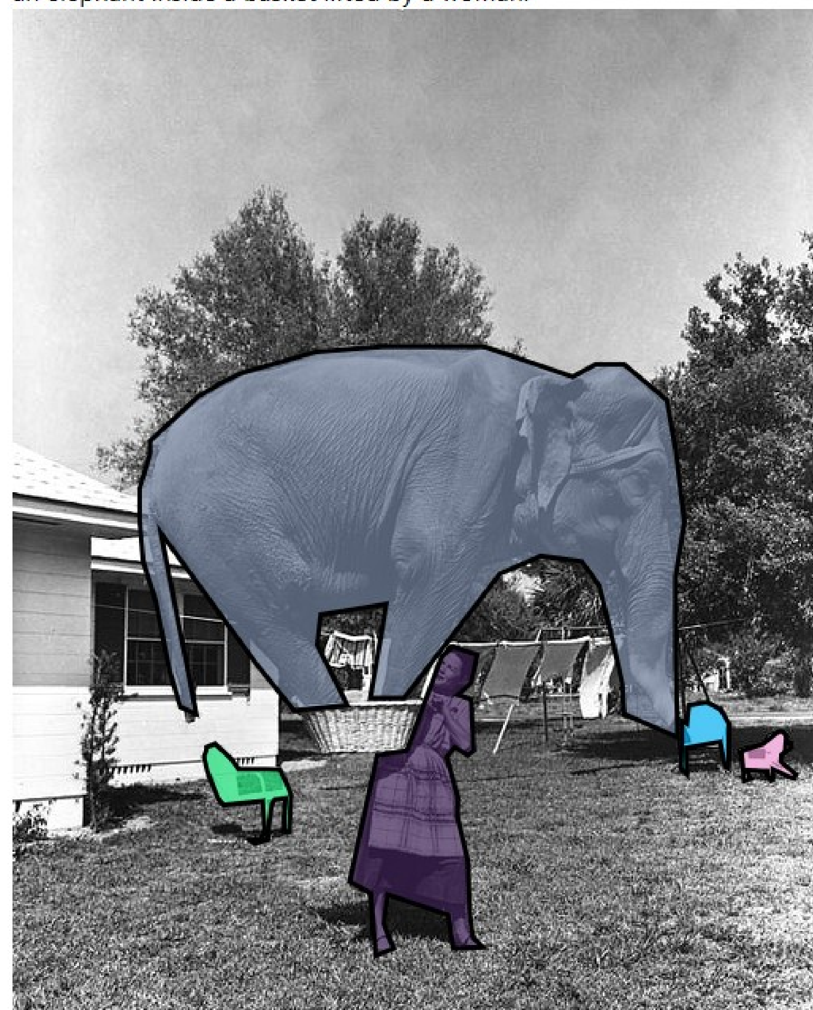


L. Lazebnik

Good training example?

Good labels?

an elephant standing on top of a basket being held by a woman.
a woman standing holding a basket with an elephant in it.
a lady holding an elephant in a small basket.
a lady holds an elephant in a basket.
an elephant inside a basket lifted by a woman.



http://mscoco.org/explore/?id=134918

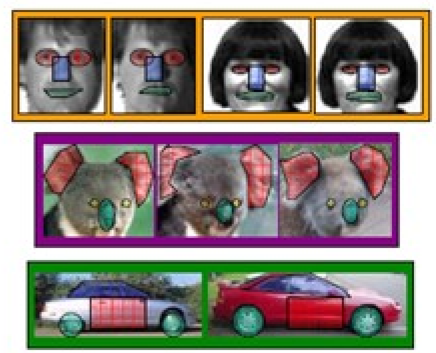
Spectrum of supervision

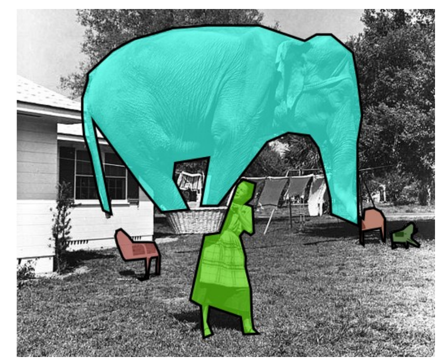
Less

More

E.G., ImageNet

E.G., MS Coco





Unsupervised

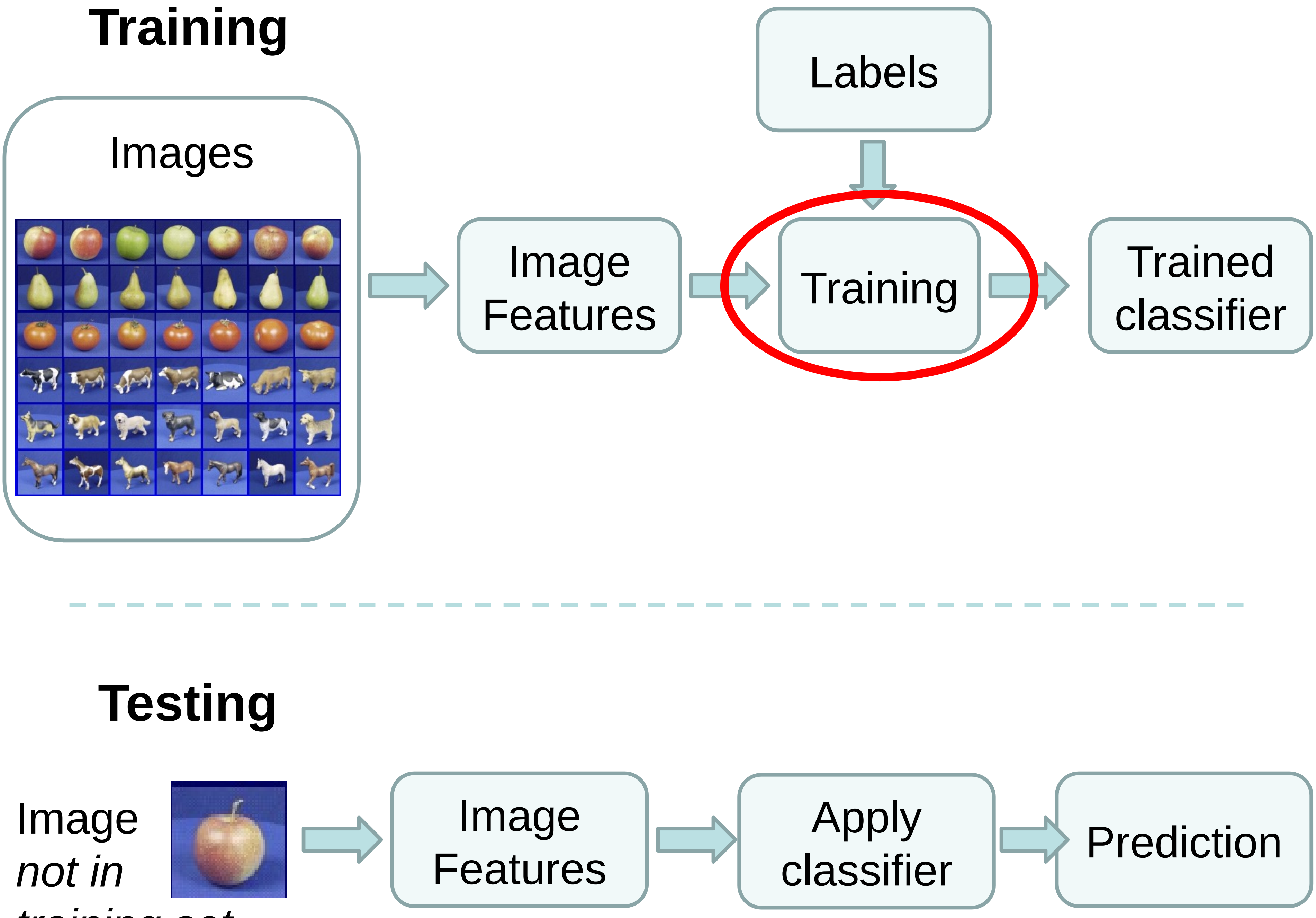
“Weakly” supervised

Fully supervised

Fuzzy; definition depends on task

‘Semi-supervised’: small partial labeling

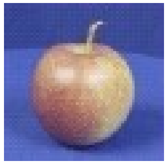
Lazebnik

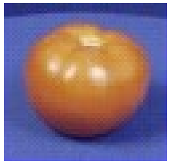


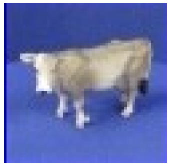
Slide credit: D. Hoiem and L. Lazebnik

The machine learning framework

Apply a prediction function to a feature representation of the image to get the desired output:

f() = "apple"

f() = "tomato"

f() = "cow"

The machine learning framework

$$f(\mathbf{x}) = y$$

Prediction function or *classifier*

Image feature

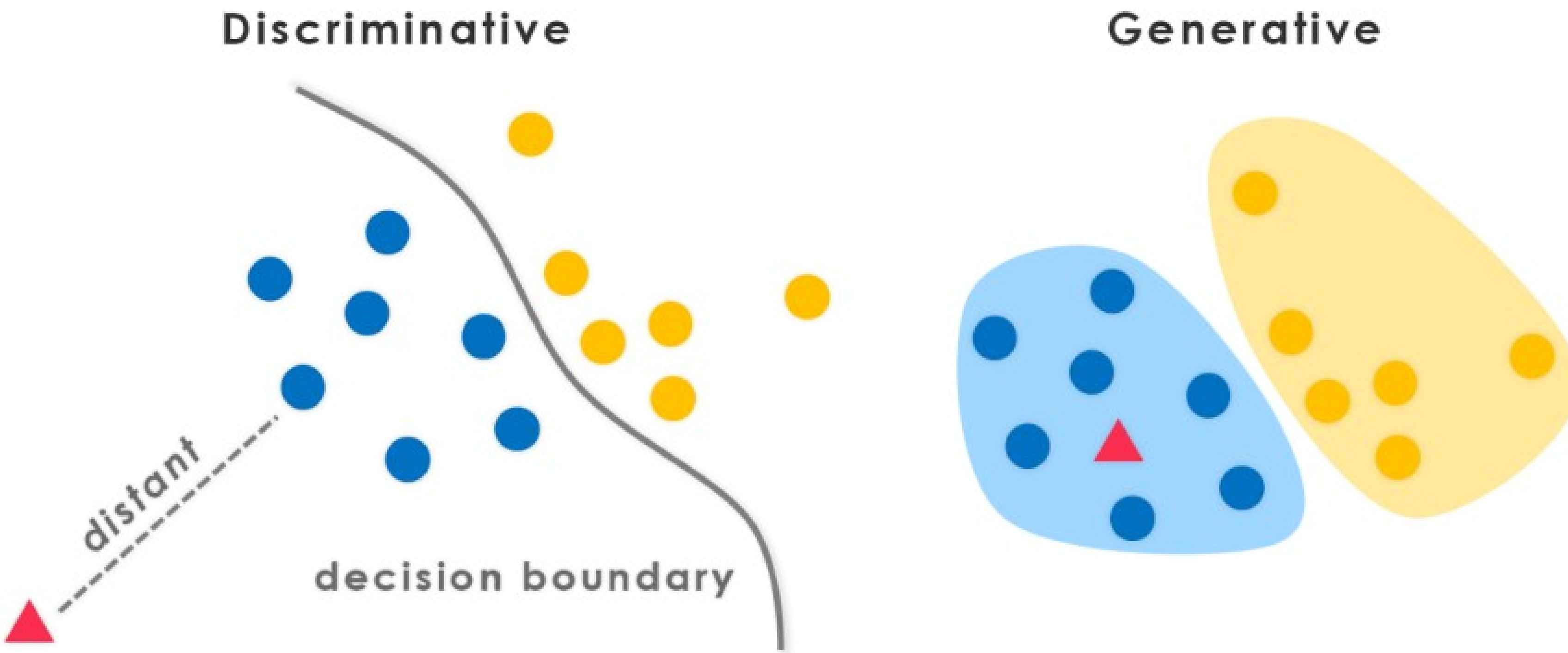
Output (label)

Training: Given a *training set* of labeled examples:

$\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$

Estimate the prediction function f by minimizing the prediction error on the training set

[evolvingai.org]



“Learn the data boundary”

Given:
Observations X
Targets Y

Learn conditional distribution:
P(Y|X=x)

“Represent the data and then define boundary”

Given:
Observations X
Targets Y

Learn joint distribution:
P(X,Y)

Machine Learning Problems

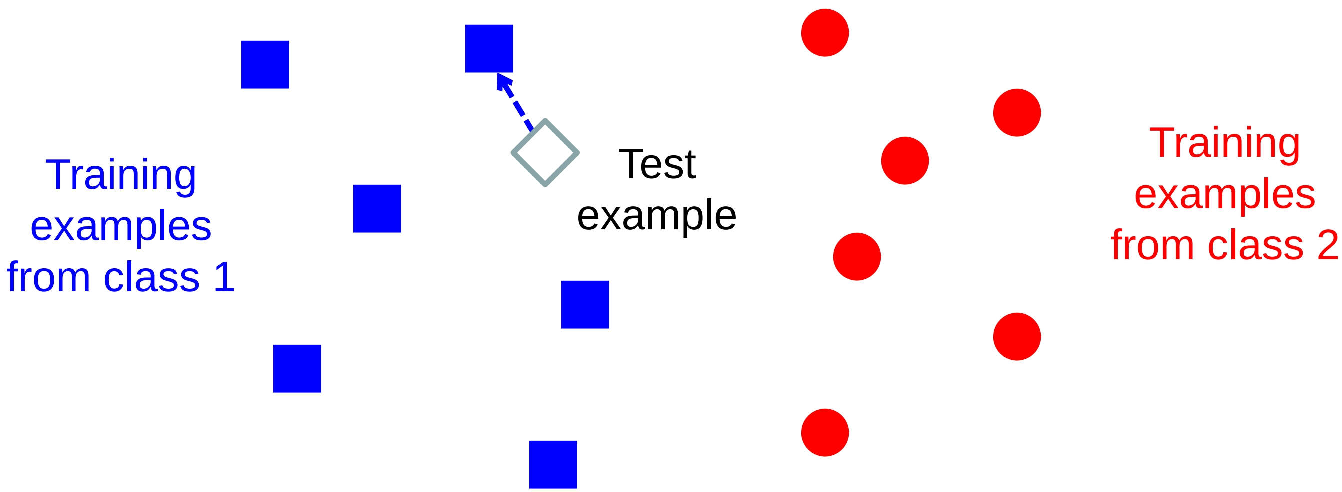
	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Classification

Assign **x** to one of two (or more) classes.

A decision rule divides input space into *decision regions* separated by *decision boundaries* – literally boundaries in the space of the features.

Classifiers: Nearest neighbor



f($\mathbf{x}$) = label of the training example nearest to $\mathbf{x}$

All we need is a distance function for our inputs

No training required!

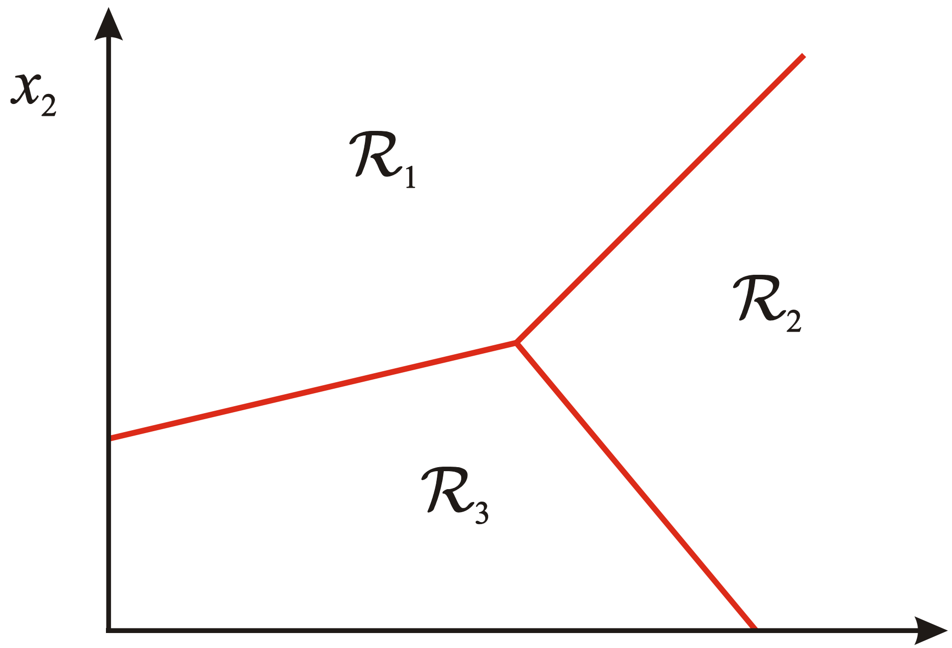
What does the decision boundary look like?

Slide credit: L. Lazebnik

Classification

Assign **x** to one of two (or more) classes.

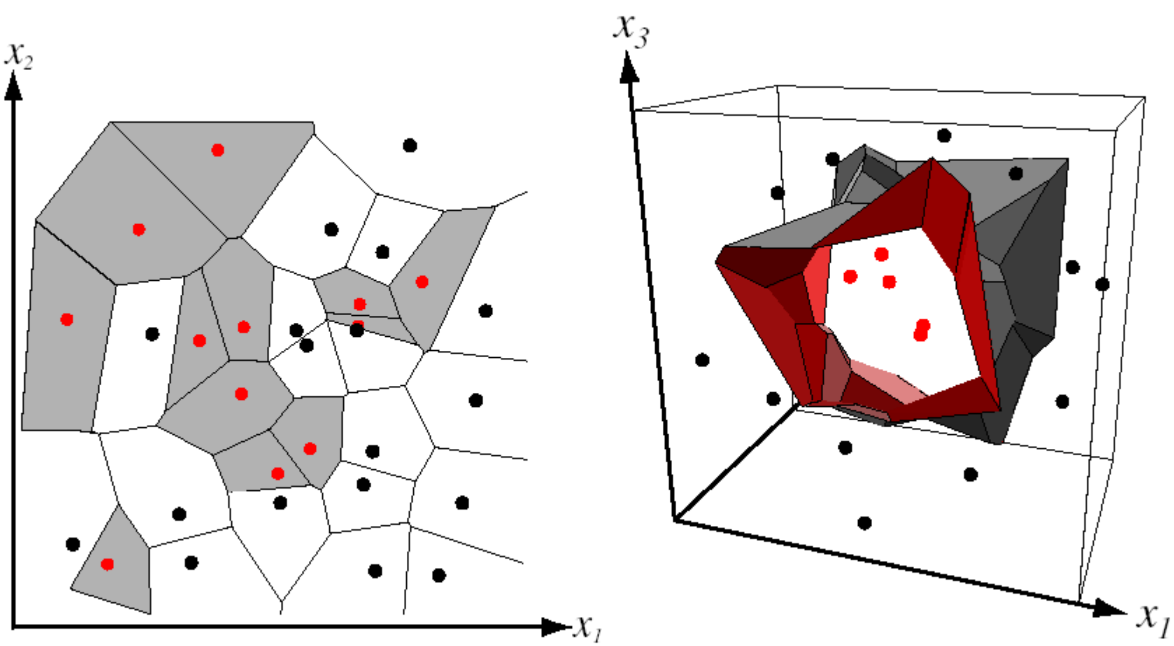
A decision rule divides input space into *decision regions* separated by *decision boundaries* – literally boundaries in the space of the features.



L. Lazebnik

Decision boundary for Nearest Neighbor Classifier

Divides input space into *decision regions* separated by *decision boundaries – Voronoi.*

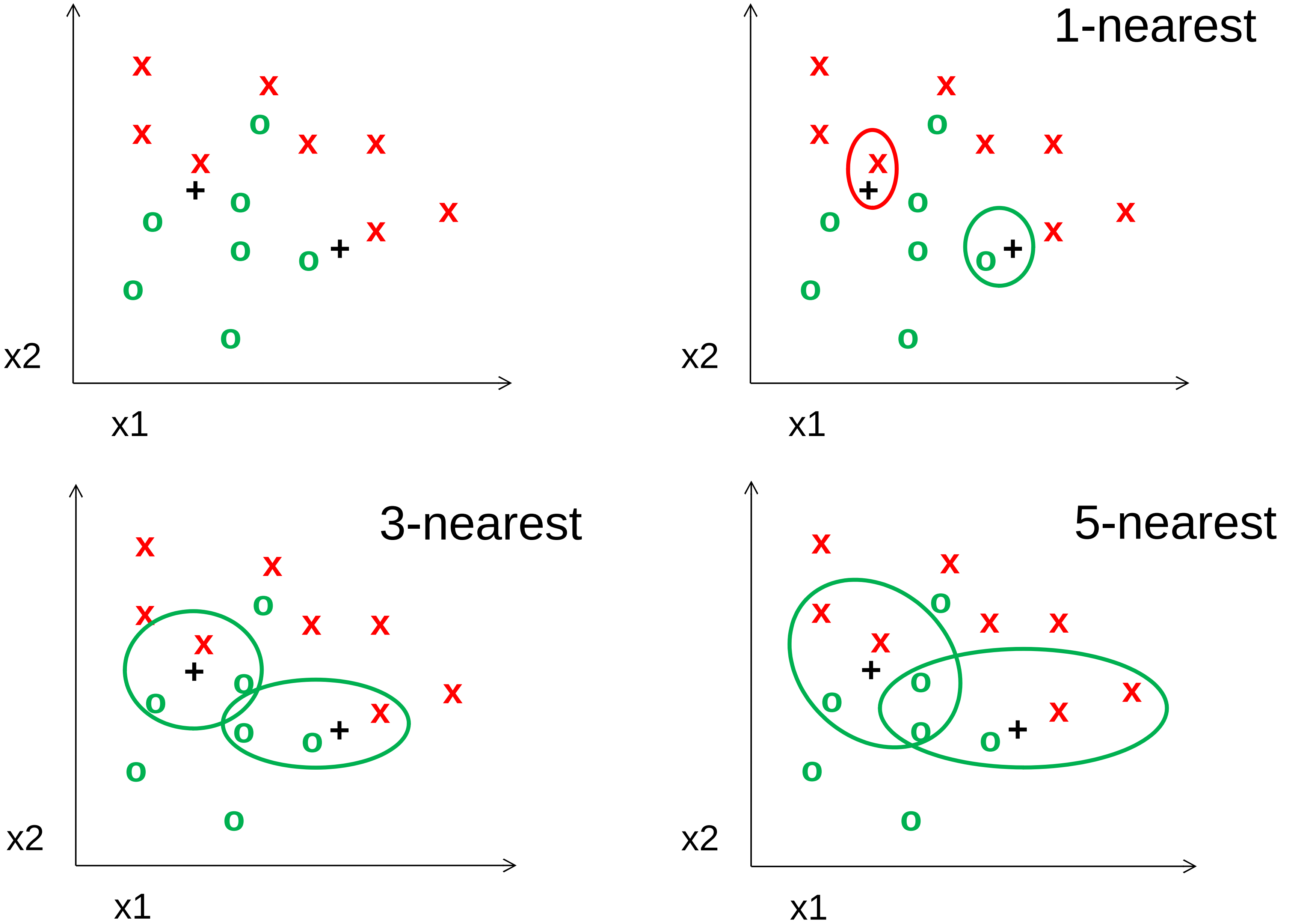


Voronoi partitioning of feature space for two-category 2D and 3D data

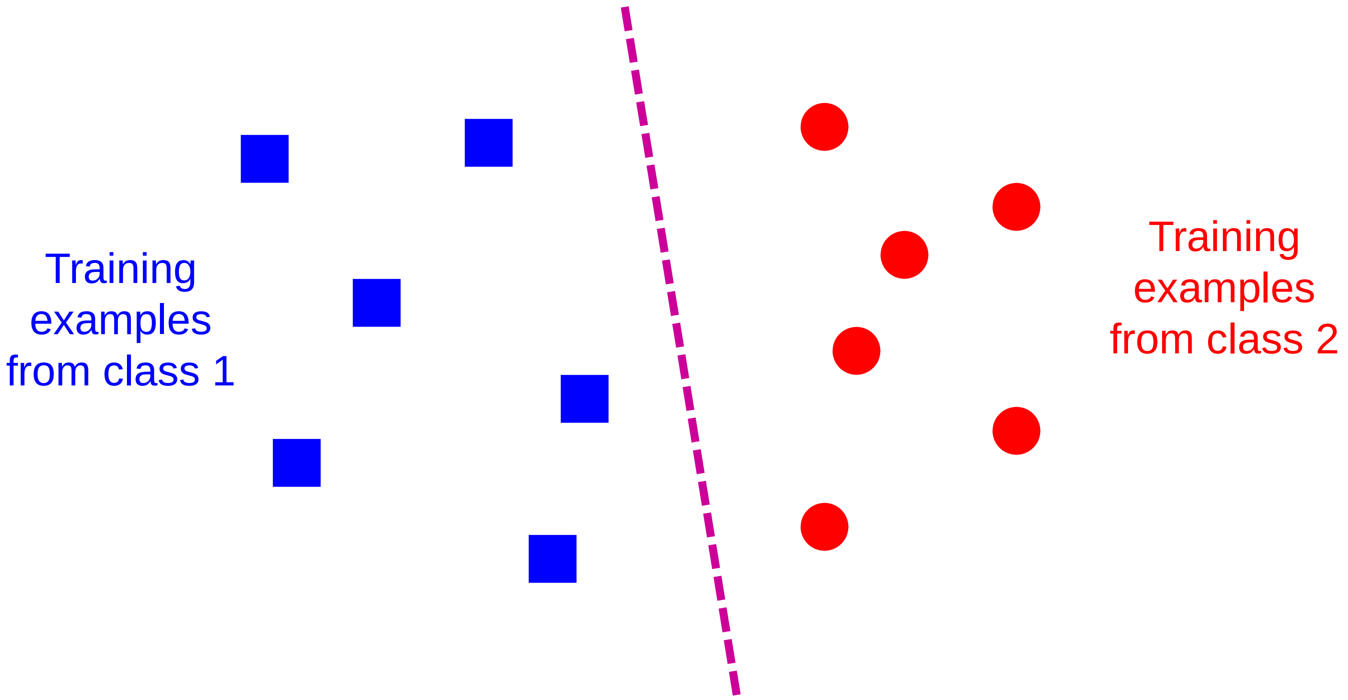
from Duda *et al.*

Source: D. Lowe

k-nearest neighbor



Classifiers: Linear

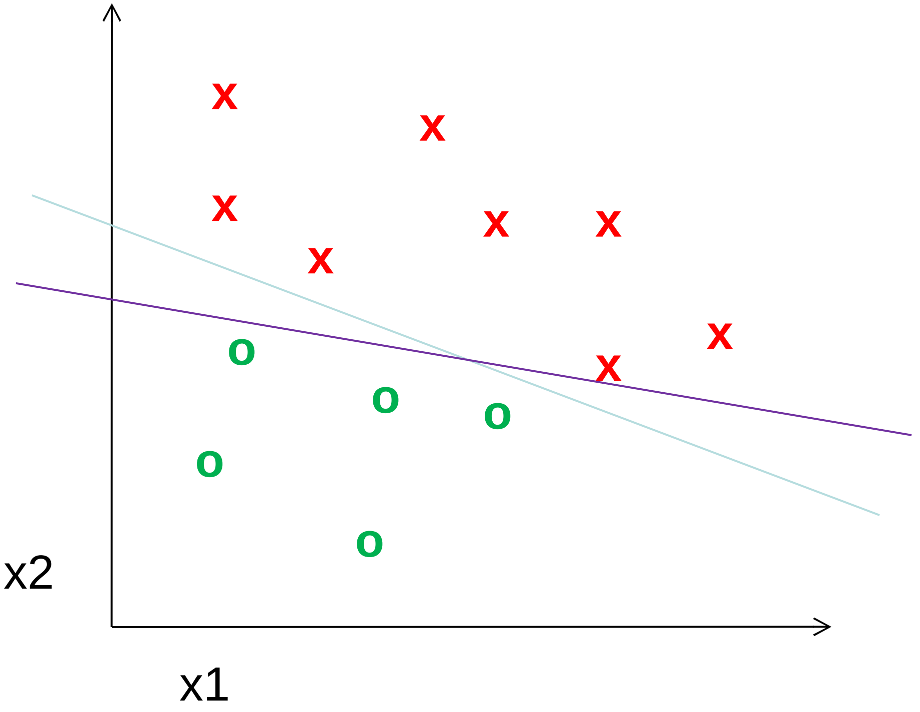


Find a *linear function* to separate the classes

Classifiers: Linear classifier

Find a *linear function*
to separate the classes:

$f(\mathbf{x}) = \mathrm{sgn}(w^T\mathbf{x} - b)$

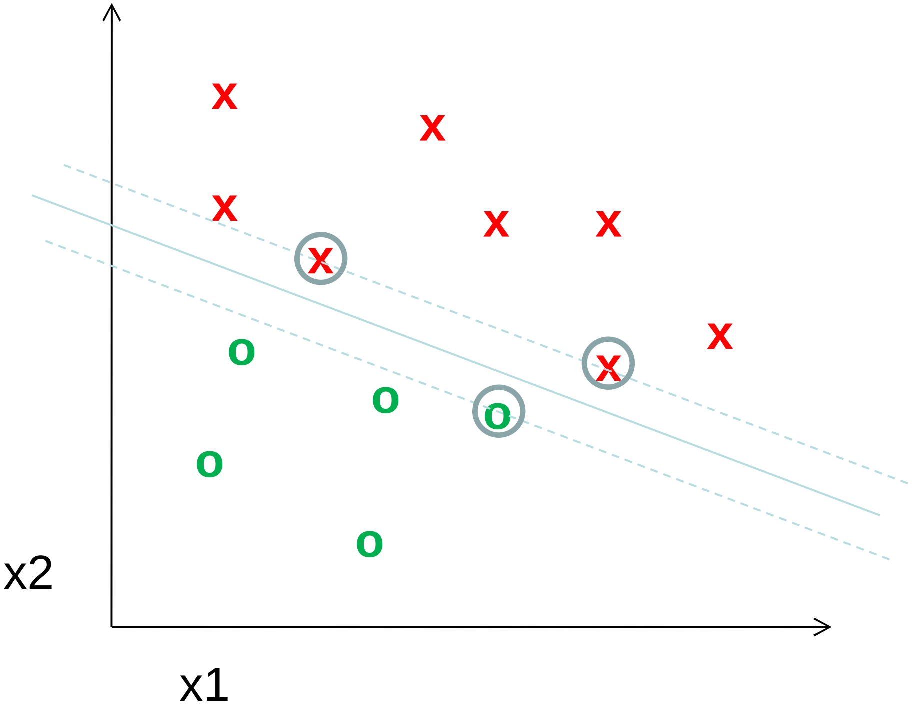


linear 2-class classifier, a point belongs to :
class 1 if $f(x) \geq 0$ i.e. $w^Tx \geq b$
class 2 if $f(x) < 0$ i.e. $w^Tx < b$

Classifiers: Linear SVM

Find a *linear function*
to separate the classes:

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



How?

$\mathbf{X}$ = all data points

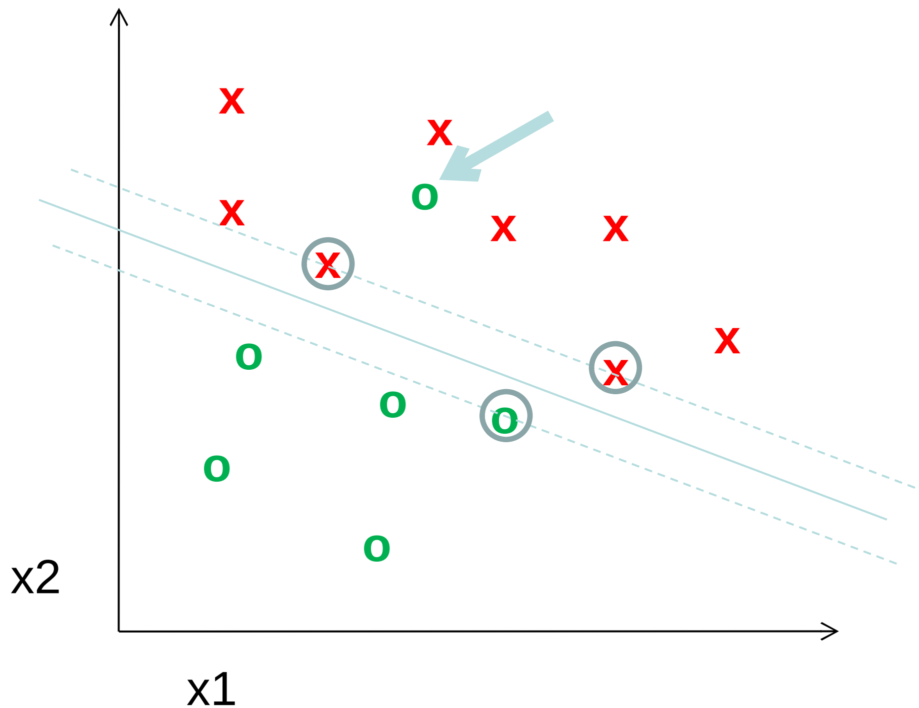
Define *hyperplane* **tX-b = 0**, where **t** is tangent to hyperplane.

Minimize **||t||** s.t. **tX-b** produces correct label for all **X**

Classifiers: Linear SVM

Find a *linear function* to separate the classes:

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

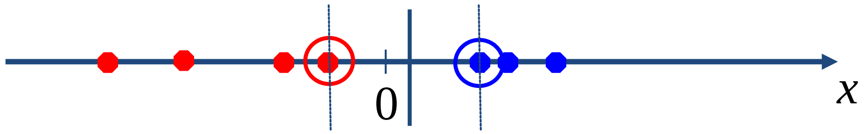


What if my data are not linearly separable?

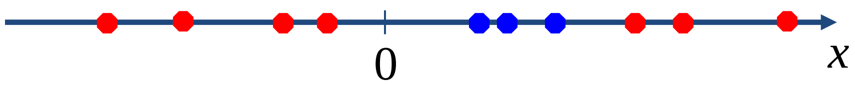
Introduce flexible ‘hinge’ loss (or ‘soft-margin’)

Nonlinear SVMs

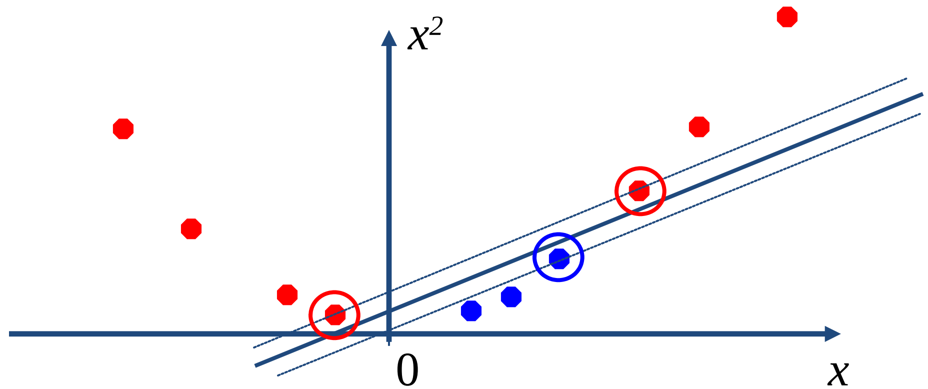
Datasets that are linearly separable work out great:



But what if the dataset is just too hard?

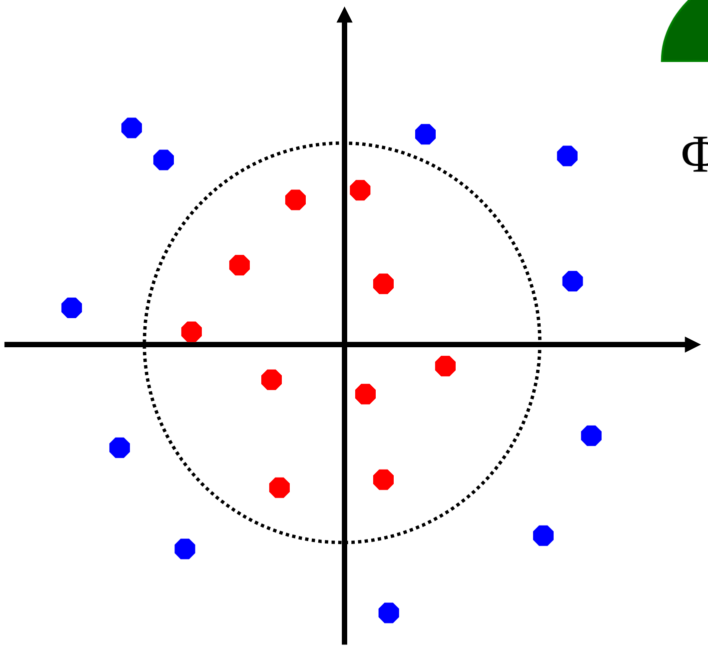


We can map it to a higher-dimensional space:

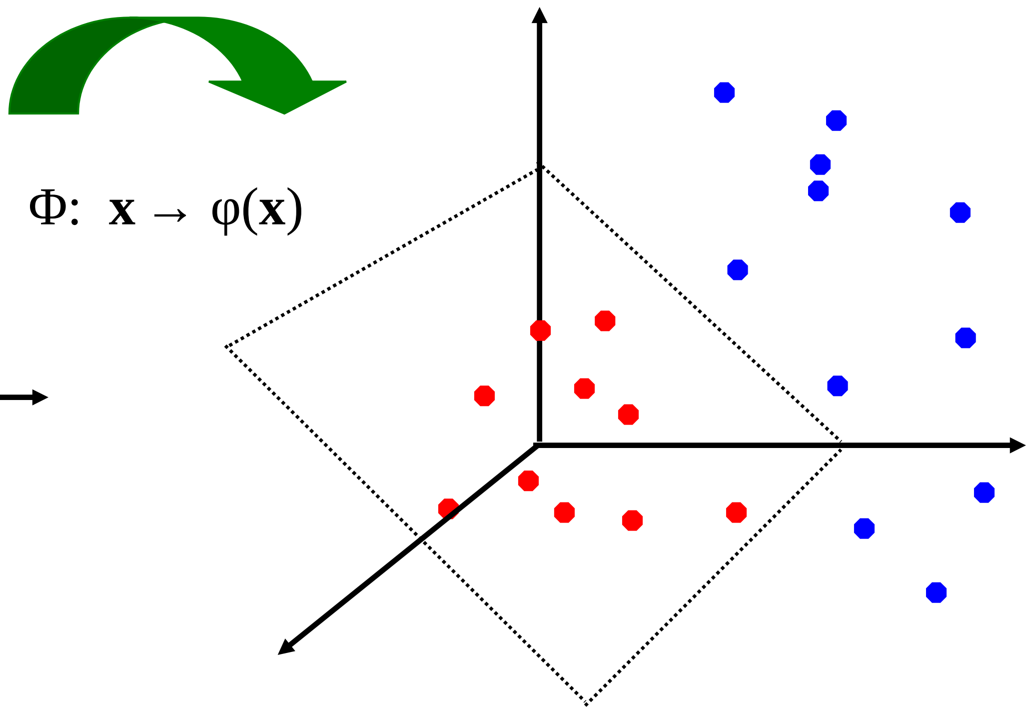


Nonlinear SVMs

Map the original input space to some higher-dimensional feature space where the training set is separable:



Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$



Slide credit: Andrew Moore

Nonlinear SVMs

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

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

This gives a *non-linear* decision boundary in the original feature space:

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

But…we only transformed the distance function *K!*

Common kernel function: Radial basis function kernel

K must satisfy mercer's conditions to be a valid kernel

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

Nonlinear SVMs

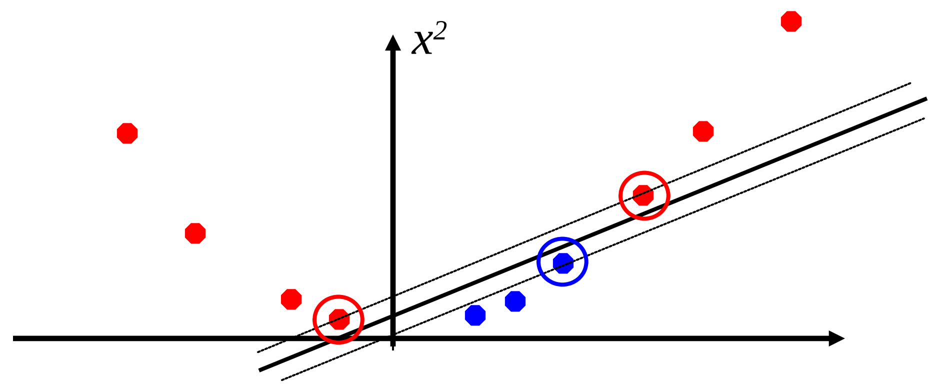
A kernel, K(xi , xj), is a dot product in some feature space

A kernel function is a function that can be applied to pairs of input examples to evaluate dot products in some corresponding (possibly infinite dimensional) feature space

We do not need to compute Φ explicitly

Nonlinear kernel: Example

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



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

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

Note y here is just another point, not a label

Kernels for SVM

- Histogram intersection kernel:

$$\mathrm{K}(h_1, h_2) = \sum_{i=1}^{N} \min(h_1(i), h_2(i))$$

- Polynomial of degree d:

$$\mathrm{K}(x_i, x_j) = ({x_i}^T . x_j + 1)^d$$

- Generalized Gaussian kernel:

$$K(h_1, h_2) = \exp\left(-\frac{1}{A} D(h_1, h_2)^2\right)$$

D can be (inverse) L1 distance, Euclidean distance, χ^2 distance, etc.

J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, IJCV 2007
Local Features and Kernels for Classifcation of Texture and Object Categories: A Comprehensive Study

Algorithm for SVM

- Input: N examples with labels (x_k, y_k) where y_k = +1 / -1
- Compute N x N matrix Q by computing $y_k y_l K(x_k, x_l)$ between all pairs of training points
- Solve optimization problem to compute αi for i = 1, …, N

$$\text{Maximize} \sum_{k=1}^{N} \alpha_k - \frac{1}{2} \sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_k \alpha_l Q_{kl} \text{ where } Q_{kl} = y_k y_l (\mathbf{\Phi}(\mathbf{x}_k)^{\mathrm{T}} \mathbf{\Phi}(\mathbf{x}_l))$$

st. $0 < \alpha_i < C$ and $\sum_i \alpha_i y_i = 0$

- Each non-zero α_i indicates that example x_i is a support vector
- Compute w and b:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \mathbf{\Phi}(\mathbf{x}_k)$$

$$b = y_K (1 - \varepsilon_K) - \mathbf{x}_K . \mathbf{w}_K$$

$$\text{where } K = \arg\max_k \alpha_k$$

- Classify test example x using: f(x) = sign($W^T x - b$)

SVM from library

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear') #kernel='rbf' or 'poly'
svclassifier.fit(X_train, y_train)
y_pred = svclassifier.predict(X_test)
```

What about multi-class SVMs?

Unfortunately, there is no “definitive” multi-class SVM.

In practice, we combine multiple two-class SVMs

One vs. others

- Training: learn an SVM for each class vs. the others
- Testing: apply each SVM to test example and assign to it the class of the SVM that returns the highest decision value

One vs. one

- Training: learn an SVM for each pair of classes
- Testing: each learned SVM “votes” for a class to assign to the test example

Slide credit: L. Lazebnik

SVMs: Pros and cons

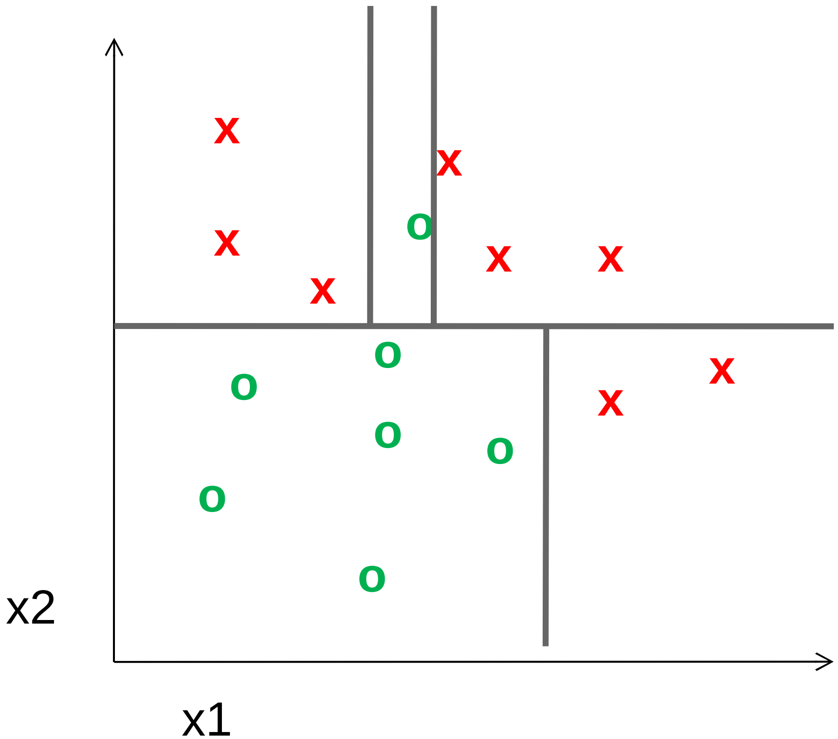
Pros

- Many publicly available SVM packages: http://www.kernel-machines.org/software
- Kernel-based framework is very powerful, flexible
- SVMs work very well in practice, even with very small training sample sizes

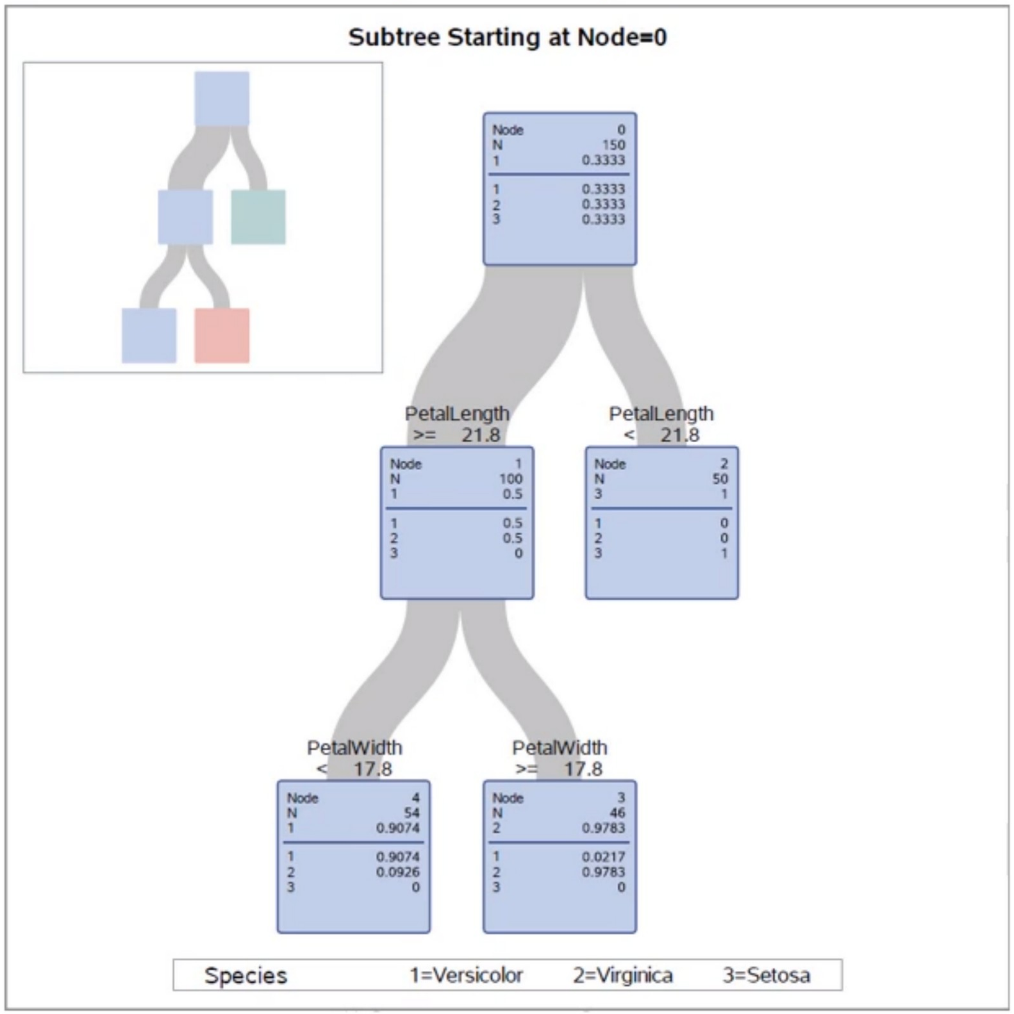
Cons

- No “direct” multi-class SVM, must combine two-class SVMs
- Computation, memory
  - During training time, must compute matrix of kernel values for every pair of examples
  - Learning can take a very long time for large-scale problems

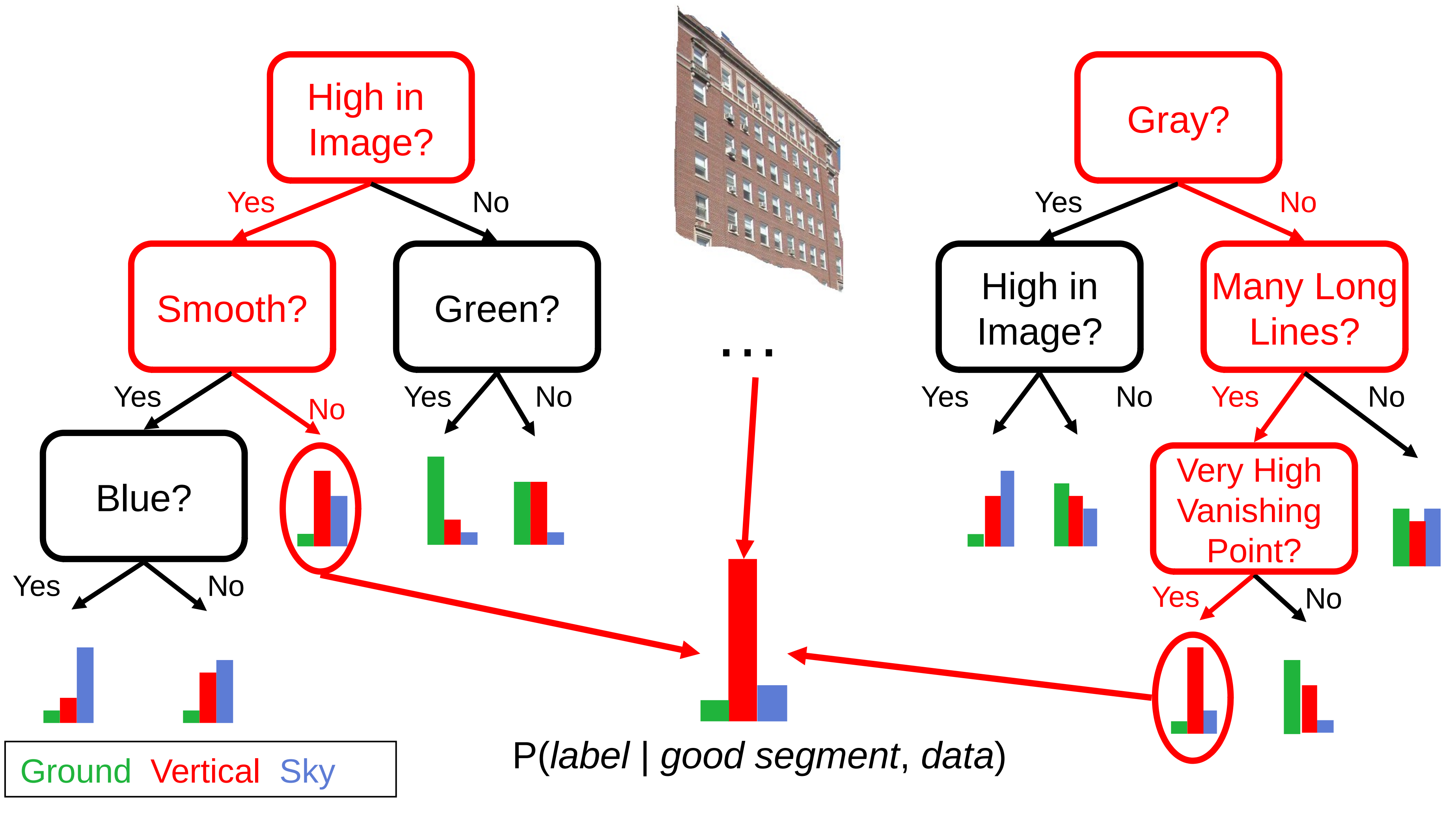
Classifiers: Decision Trees



Classifiers: Decision Trees



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

Summary: Classifiers

- Nearest-neighbor and k-nearest-neighbor classifiers
  - L1 distance, χ^2 distance, quadratic distance, histogram intersection
- Support vector machines
  - Linear classifiers
  - Margin maximization
  - The kernel trick
  - Kernel functions: histogram intersection, generalized Gaussian
  - Multi-class
- Of course, there are many other classifiers out there
  - Decision Trees, Neural networks, etc. …

Training

Training Images

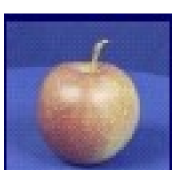
Image Features

Training Labels

Training

Learned classifier

Testing



Test Image

Image Features

Apply classifier

Prediction

Slide credit: D. Hoiem and L. Lazebnik

Design Considerations

Features and **distance measures**

define visual similarity.

Training labels

dictate that examples are the same or different.

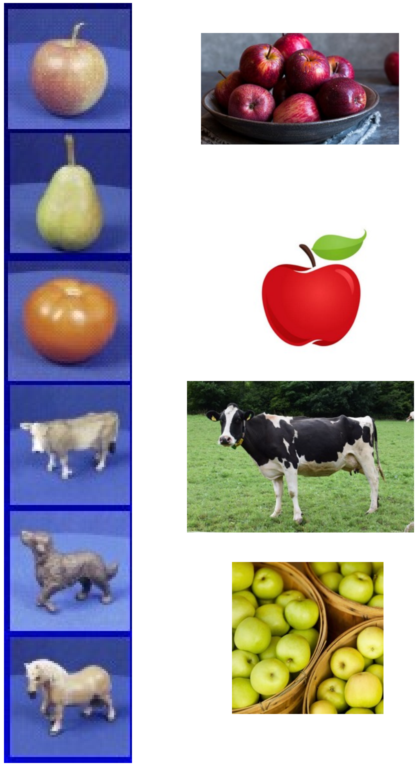
Classifiers

learn weights (or parameters) of features and distance measures…

so that visual similarity predicts label similarity.

Generalization

Training set (labels known)



Test set (labels unknown)

How well does a learned model generalize from the data it was trained on to a new test set?

Generalization Error

Bias:

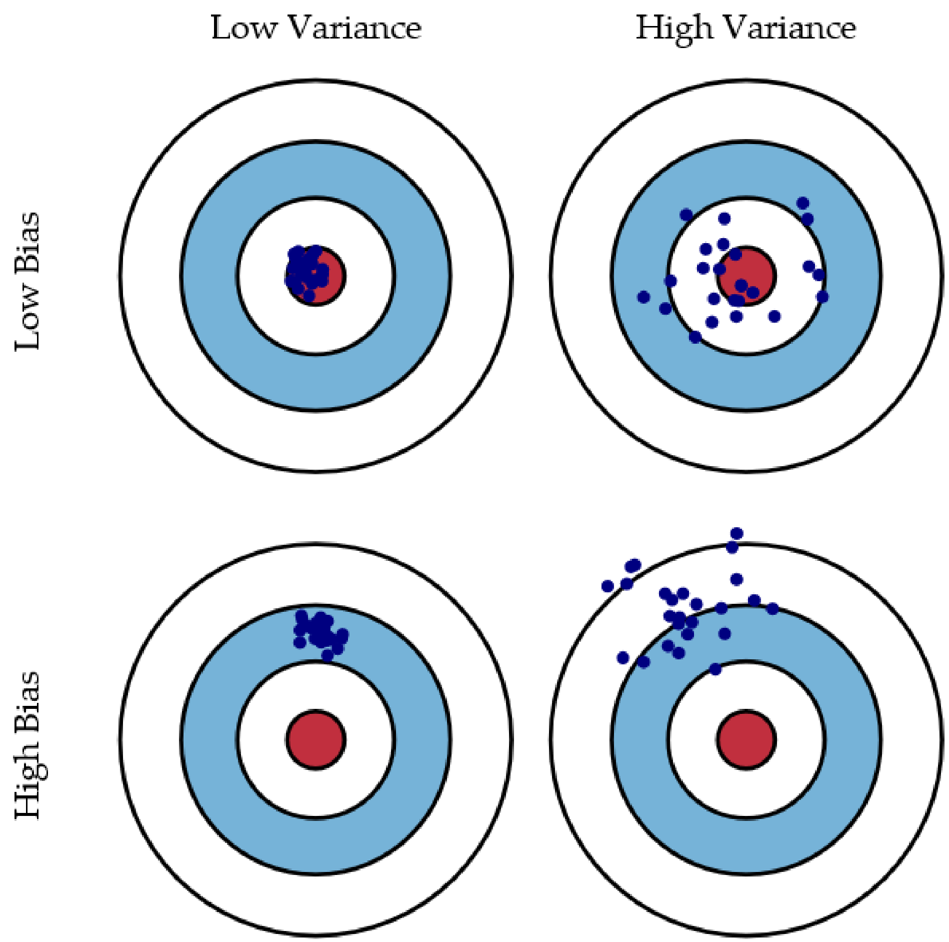
Difference between the expected (or ‘average’) prediction of our model and the correct value.

Error due to inaccurate assumptions/simplifications.

Variance:

- Amount that the estimate of the target function will change if different training data was used.

Bias/variance trade-off



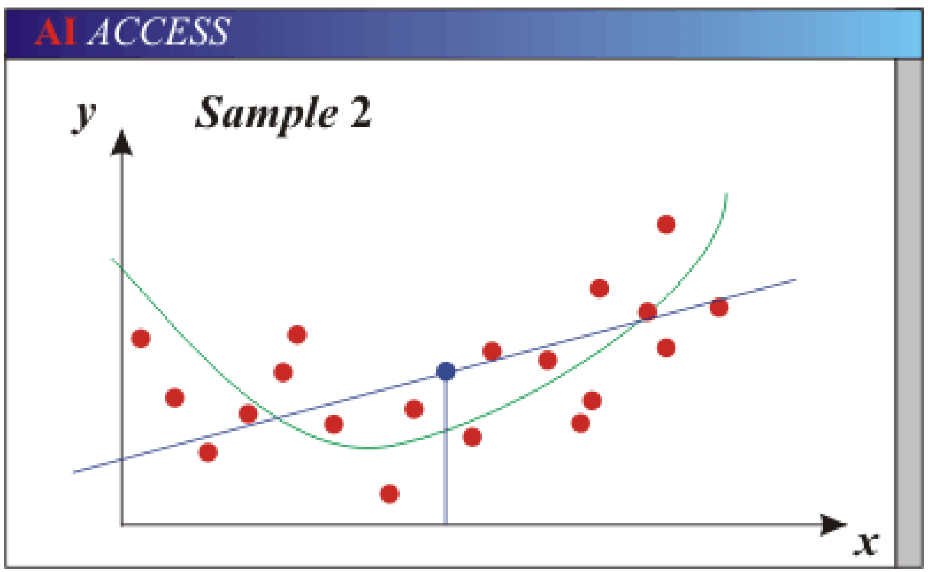
Bias = ‘accuracy’
Variance = ‘precision’

[Scott Fortmann-Roe]

Generalization Error Effects

Underfitting: model is too “simple” to represent all the relevant class characteristics

- High bias (few degrees of freedom) and low variance
- High training error and high test error

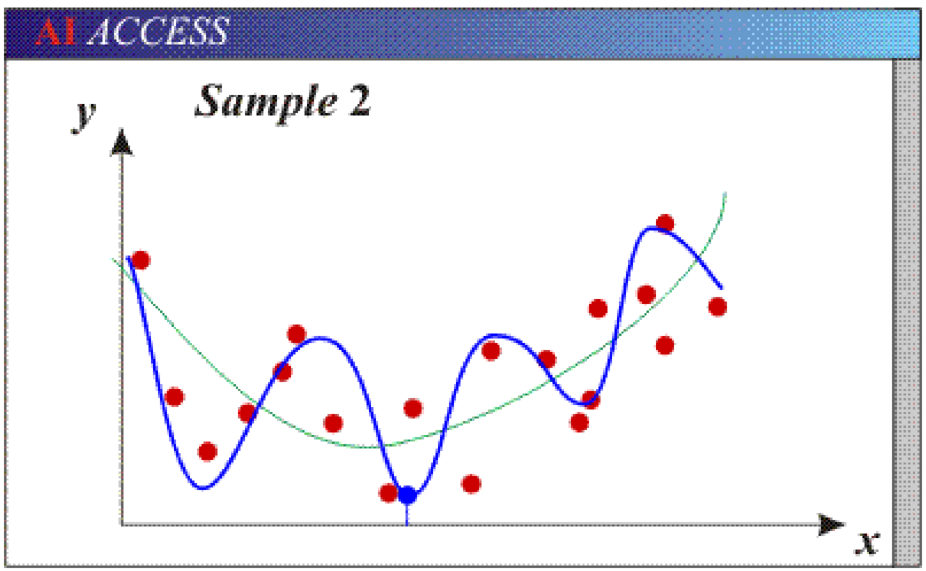


Green line = true data-generating function without noise
Blue line = data model which underfits (low capacity)

Generalization Error Effects

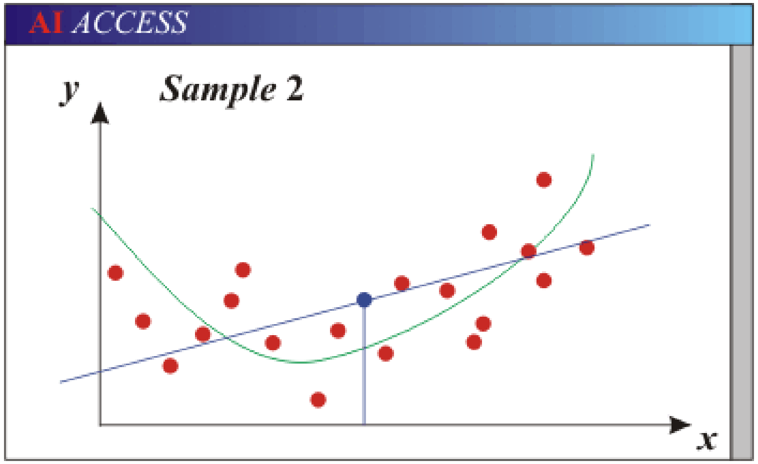
Overfitting: model is too “complex” and fits irrelevant characteristics (noise) in the data

- Low bias (many degrees of freedom) and high variance
- Low training error and high test error



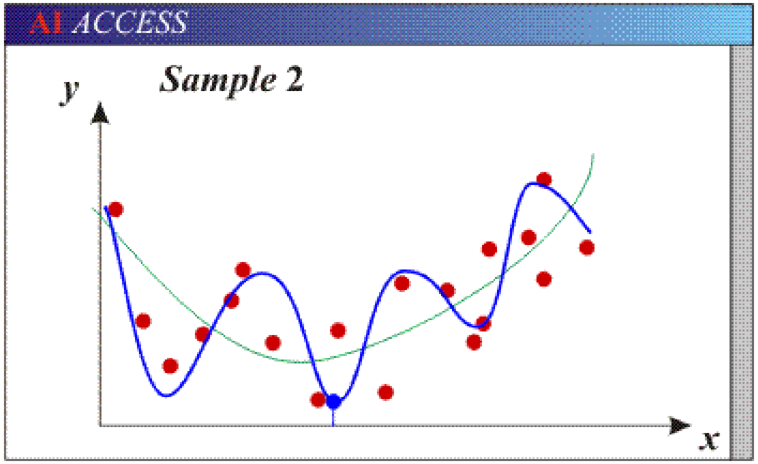
Green line = true data-generating function without noise
Blue line = data model which overfits

Bias-Variance Trade-off



Models with too few parameters are inaccurate because of a large bias.

- Not enough flexibility!
- Too many assumptions



Models with too many parameters are inaccurate because of a large variance.

- Too much sensitivity to the sample.
- Slightly different data ->
  very different function.

Bias-Variance Trade-off

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

Unavoidable error

Error due to incorrect assumptions

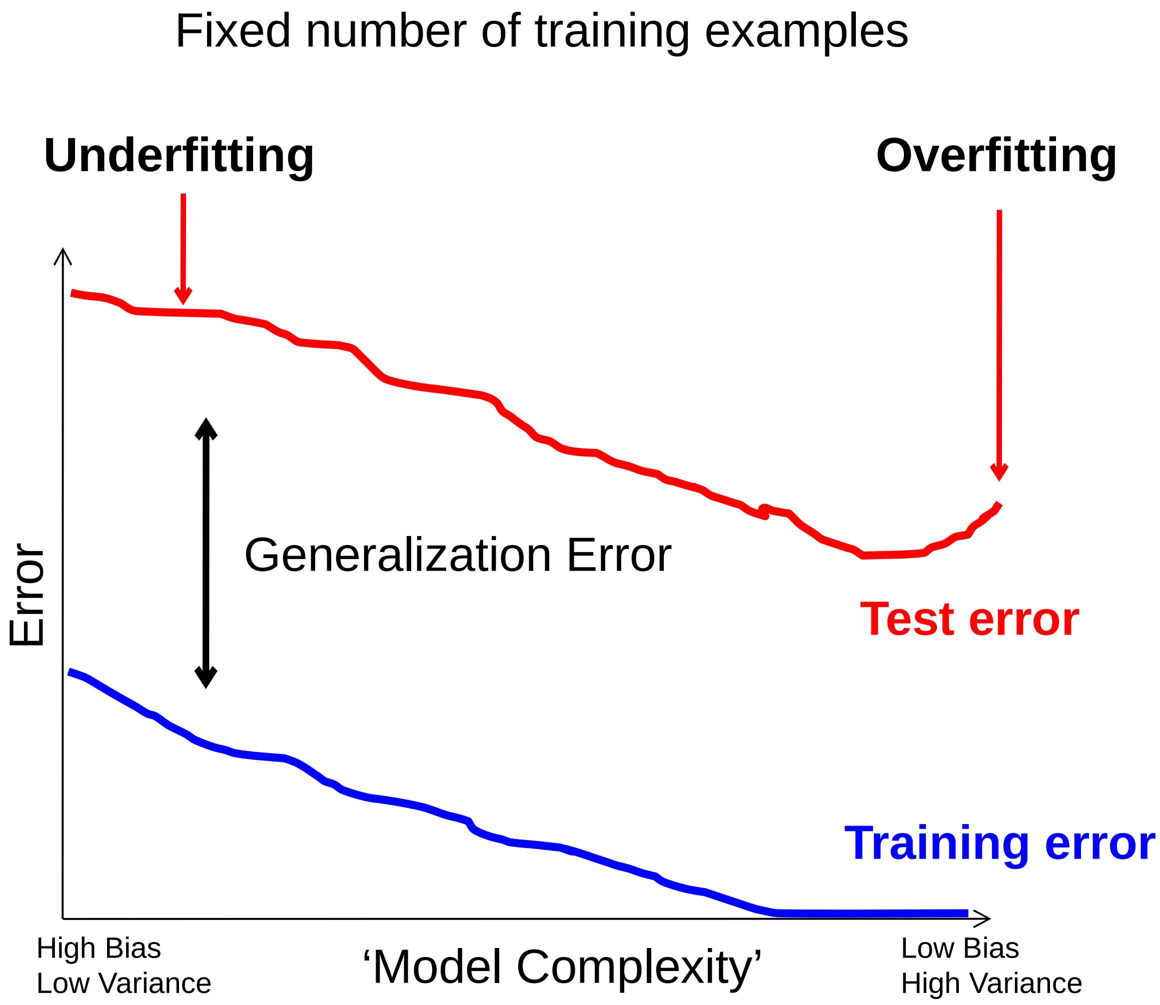
Error due to variance of training samples

For explanations of bias-variance (also Bishop's "Neural Networks" book):

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

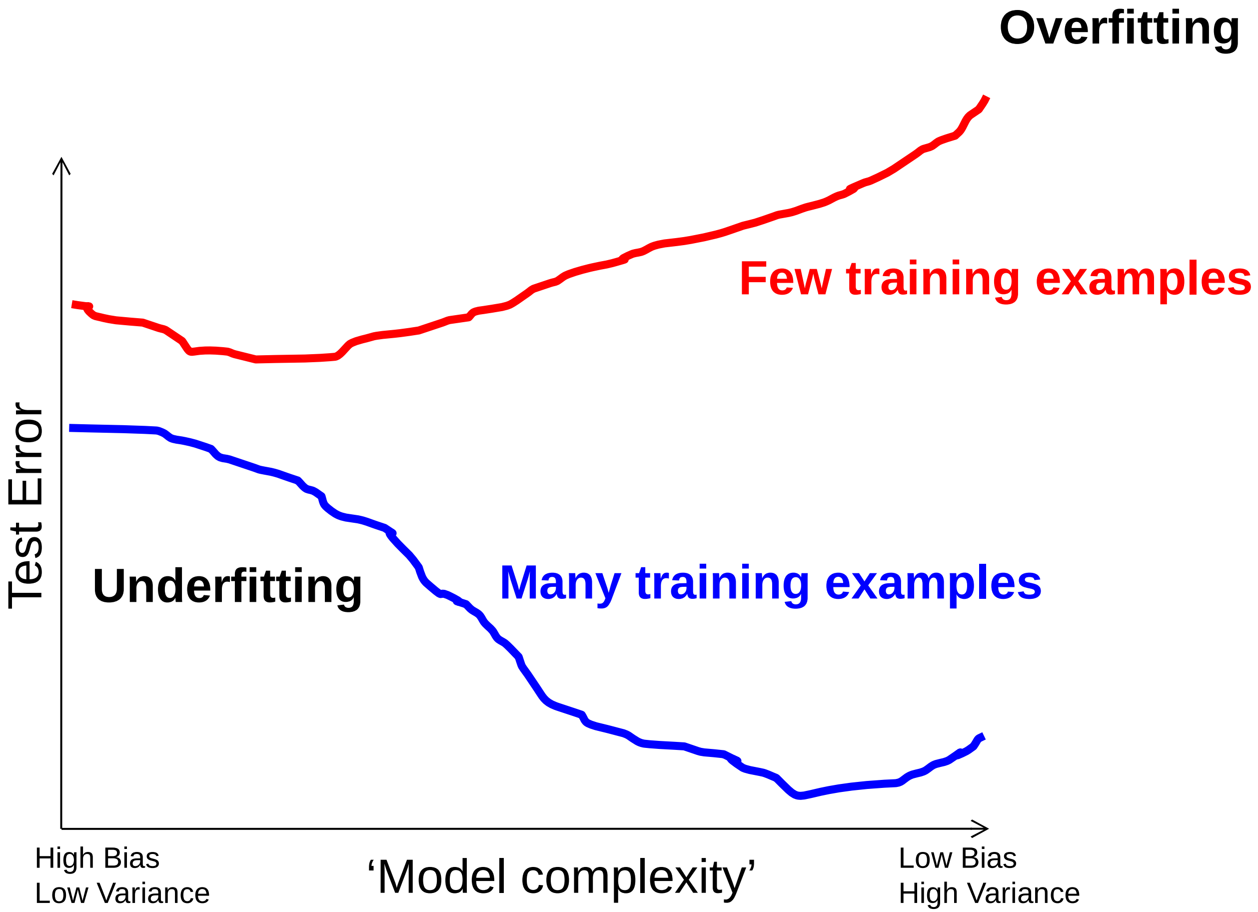
Slide credit: D. Hoiem

Bias-variance tradeoff



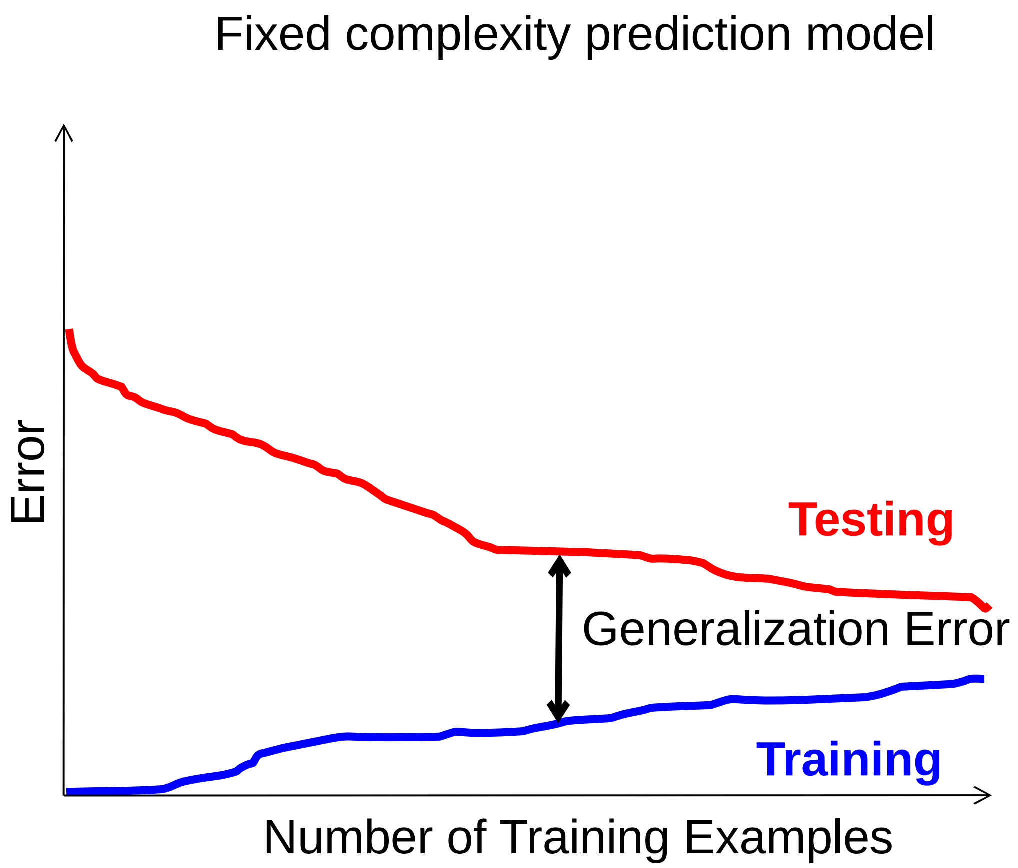
Slide credit: D. Hoiem

Bias-variance tradeoff



Slide credit: D. Hoiem

Effect of Training Size



Many classifiers to choose from…

- K-nearest neighbor
- SVM
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- Restricted Boltzmann Machines
- Neural networks
- Deep Convolutional Network
- …

Which is the best?

Claim:

The decision to use machine learning is more important than the choice of a particular learning method.

It is more important to have more or better labeled data than to use a different supervised learning technique.

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)

Slide credit: D. Hoiem

Finally…

- 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: D. Hoiem