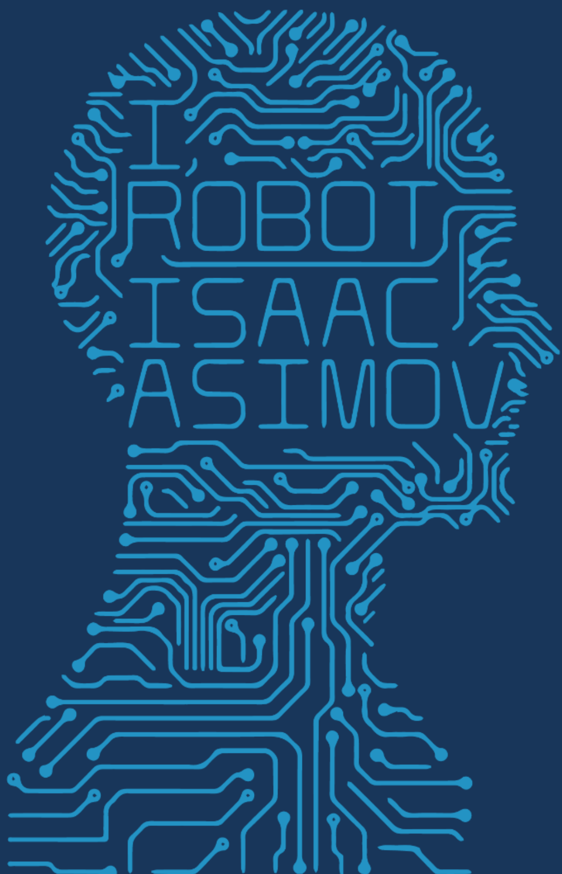
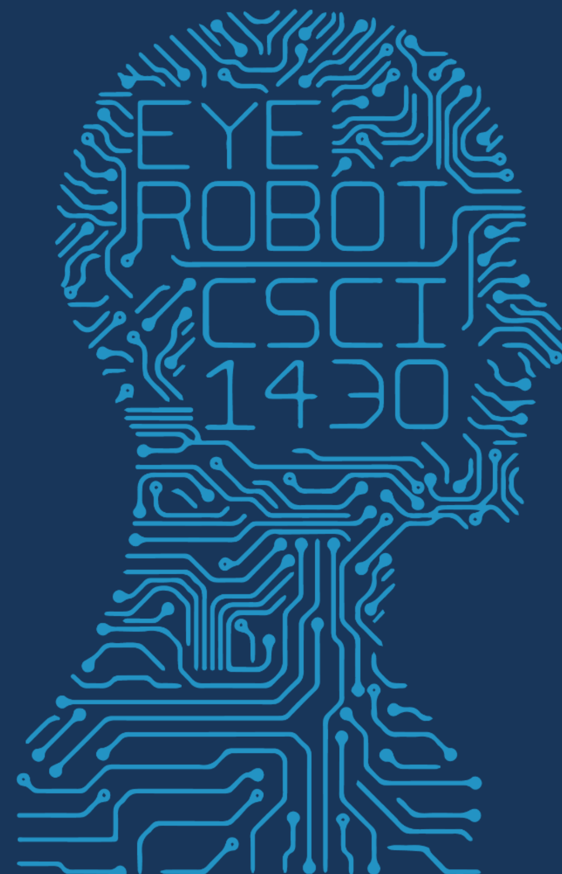


1950

Future Vision



2017 MWF 1pm 368

Computer Vision

Photo: CMU Machine Learning Department Protests G20

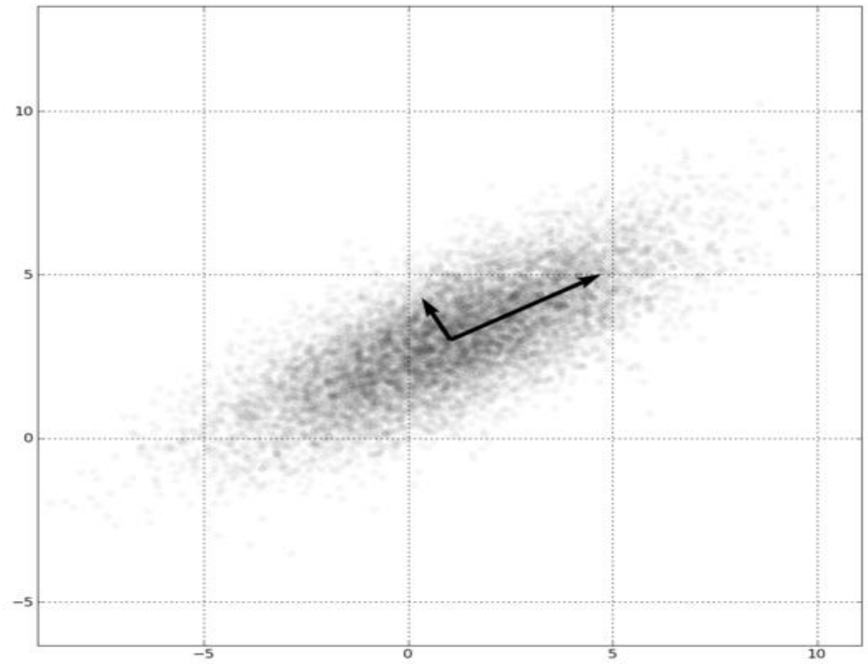
Slides: James Hays, Isabelle Guyon, Erik Sudderth, Mark Johnson, Derek Hoiem

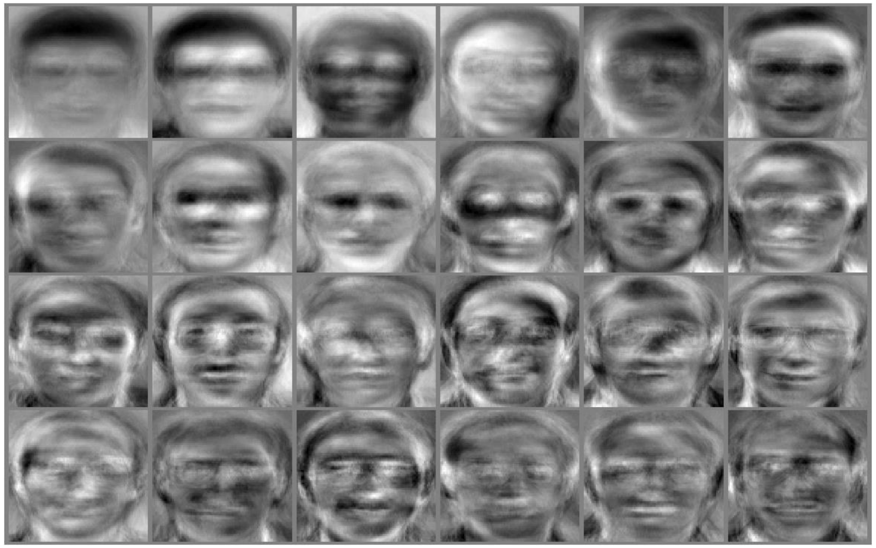
Machine Learning Problems

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

PCA: Principal Component Analysis

- The best possible lower dimensional representation based on linear projections.
- An basis of directions of variance ordered by their significance.
- Throw away least variance dimensions to reduce data representation.





R.P.W. Duin

Machine Learning Problems

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

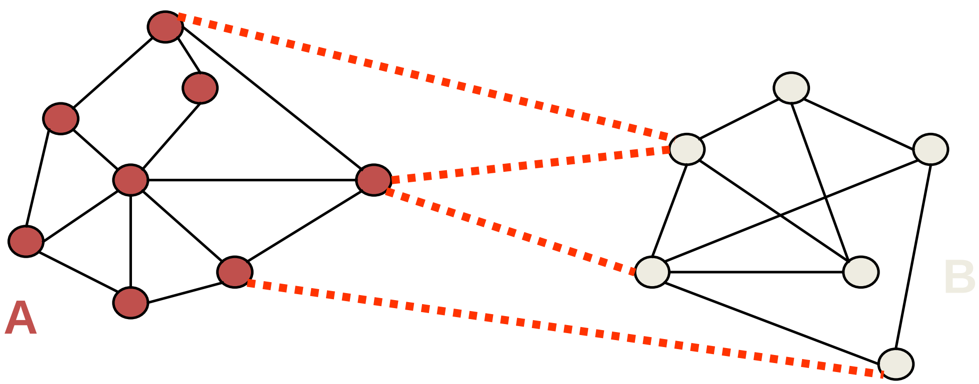
How do we cluster?

- K-means
  - Iteratively re-assign points to the nearest cluster center.
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters.
- Mean-shift clustering
  - Estimate modes of probability density function.
- Spectral clustering
  - Split the nodes in a graph based on assigned links with similarity weights.

Spectral clustering

Group points based on graph structure & edge costs.

Captures "neighborhood-ness" or local smoothness.



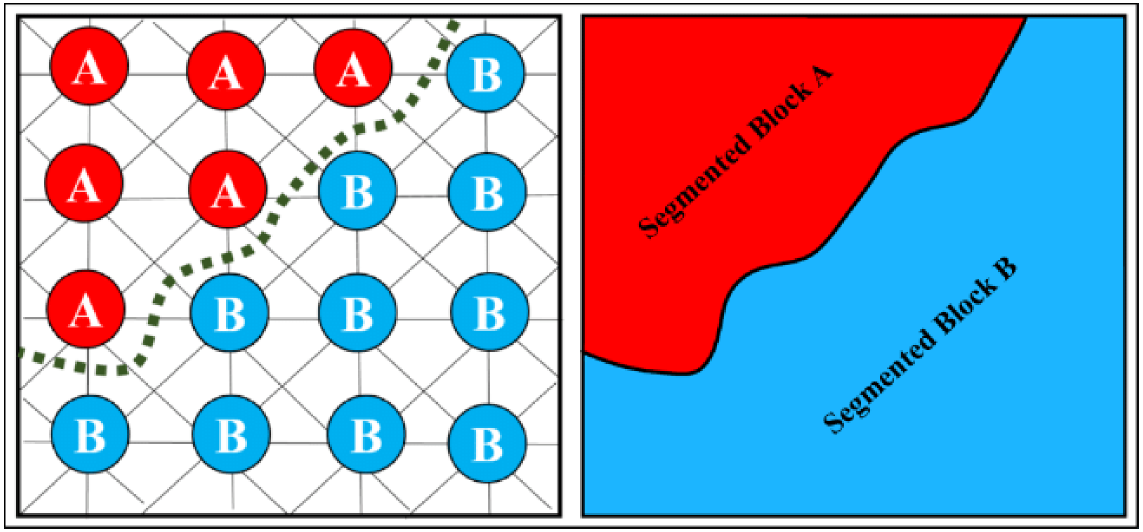


Image: Hassan et al.

Machine Learning Problems

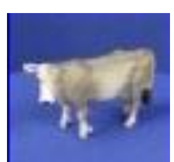
	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

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”

Slide credit: L. Lazebnik

The machine learning framework

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

Prediction function — Image feature — Output (label)

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

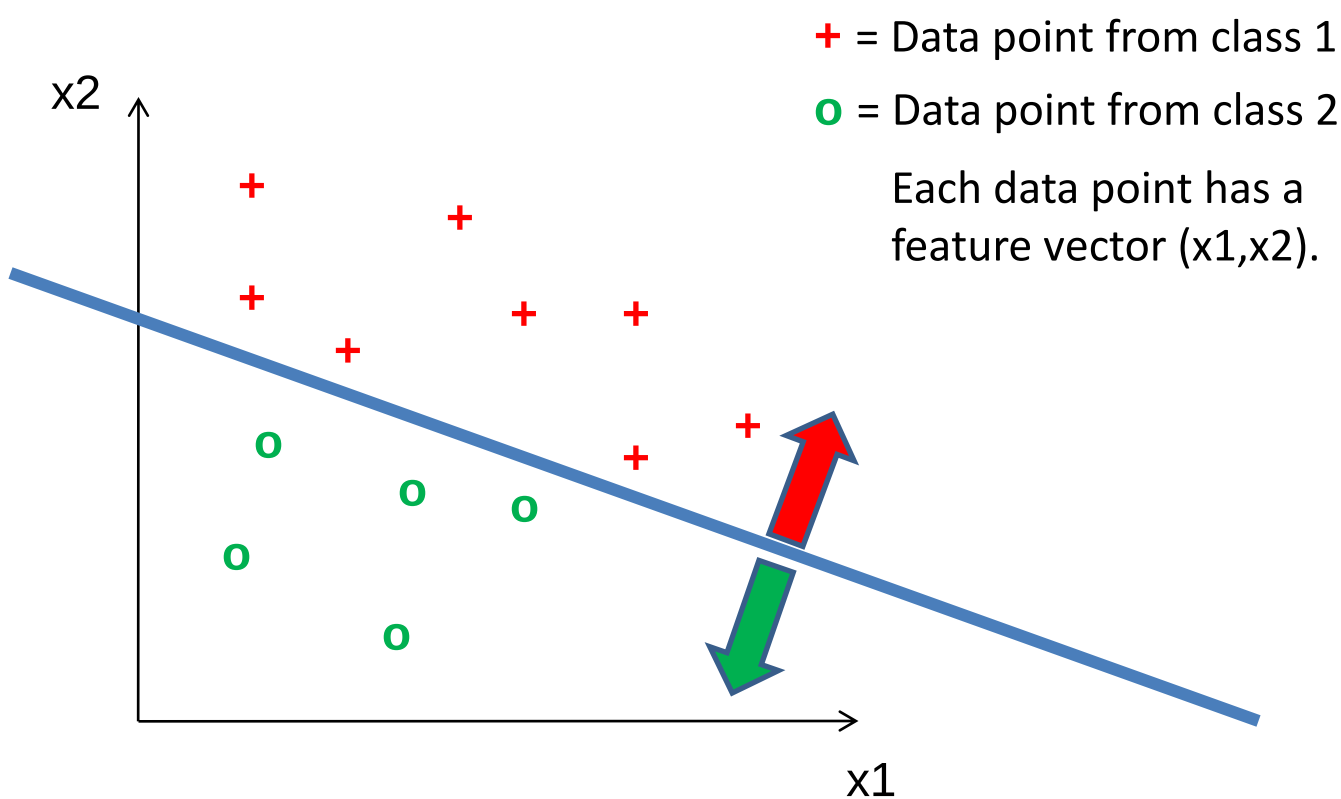
$$\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$$

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

Testing: Apply f to a unseen *test example* **x** and output the predicted value $y = f(\mathbf{x})$ to *classify* **x**.

Learning a classifier

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



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

Random train/validate splits = cross validation

Steps

Training

Training Images

Image Features → Training → Learned classifier

Training Labels → Training

Testing

Test Image

Image Features → Apply classifier → Prediction

Slide credit: D. Hoiem and L. Lazebnik

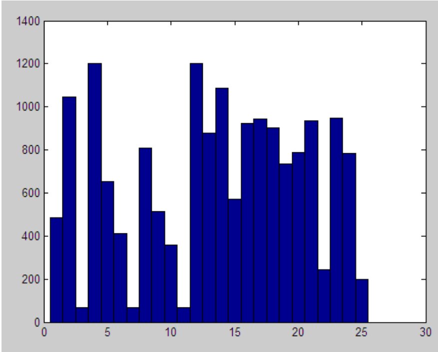
Features

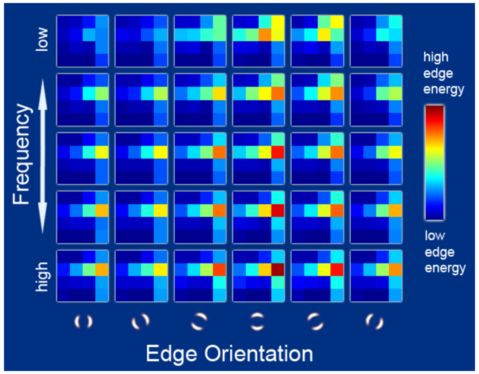
- Raw pixels

- Histograms

- GIST descriptors

- ...





Slide credit: L. Lazebnik

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
- Restricted Boltzmann Machines
- 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.

*Deep learning seems to be an exception to this, currently, because it learns the feature representation.

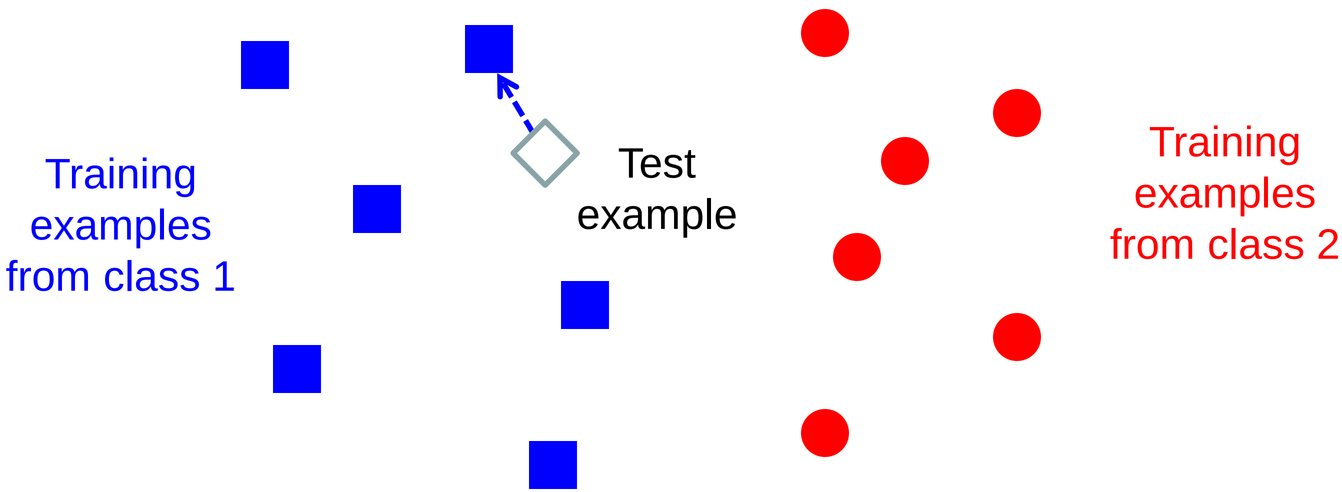
Claim:

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

*Again, deep learning may be an exception here for the same reason, but deep learning _needs_ a lot of labeled data in the first place.

"The Unreasonable Effectiveness of Data" - Norvig

Classifiers: Nearest neighbor

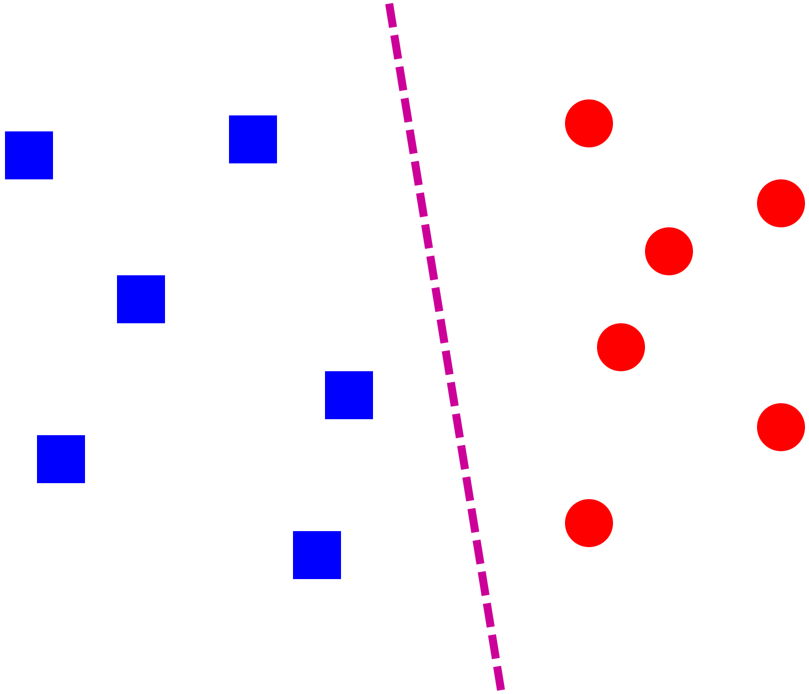


f($\mathbf{x}$) = label of the training example nearest to $\mathbf{x}$

- All we need is a distance function for our inputs
- No training required!

Slide credit: L. Lazebnik

Classifiers: Linear



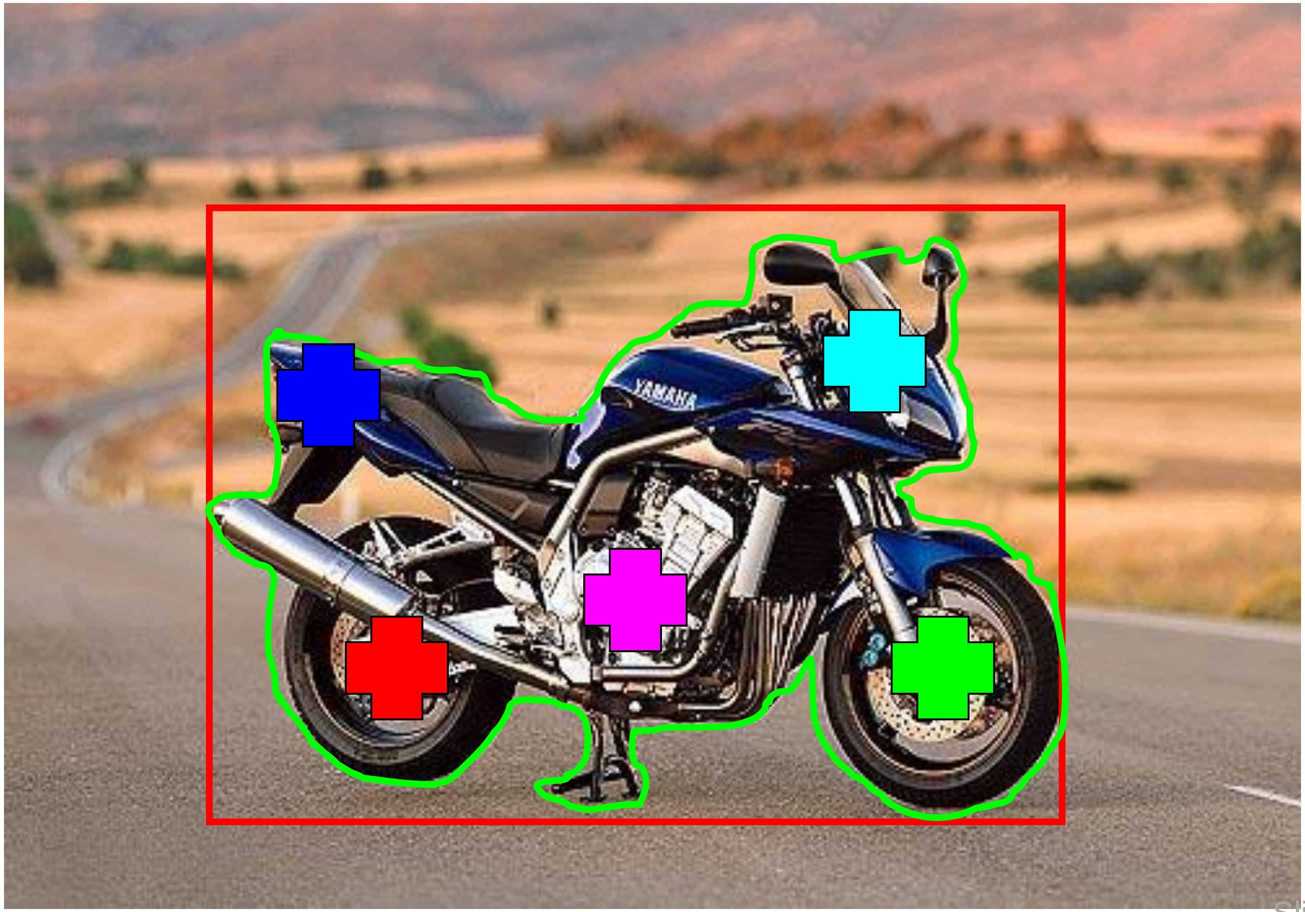
- Find a *linear function* to separate the classes:

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

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



Slide credit: L. Lazebnik

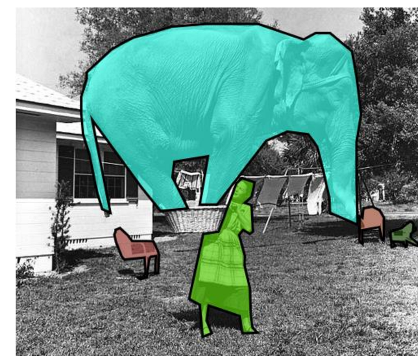
Spectrum of supervision

Less

More

E.G., ImageNet

E.G., MS Coco



Unsupervised

“Weakly” supervised

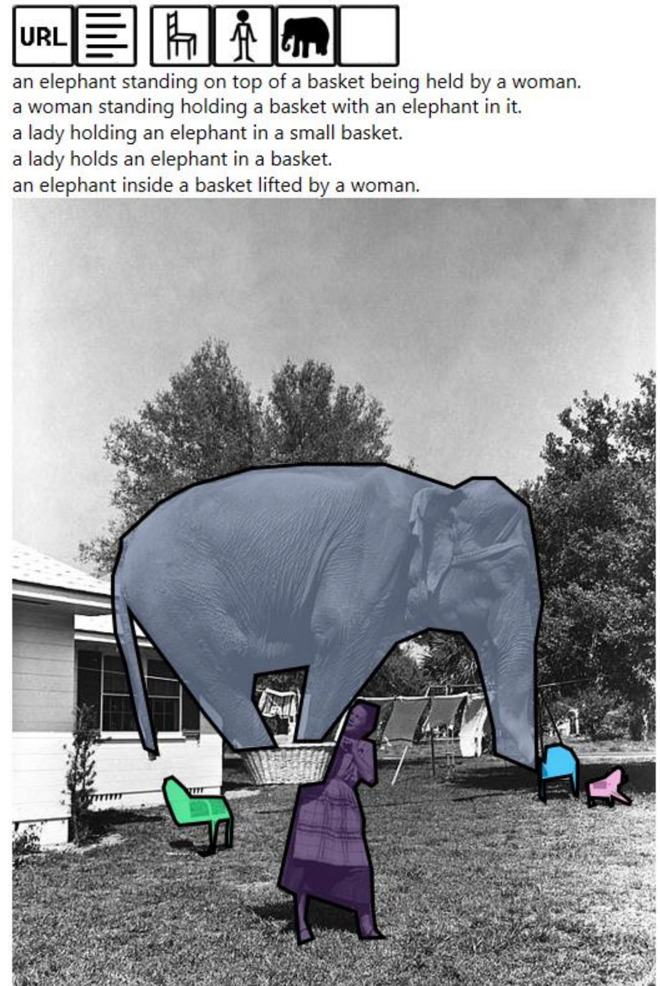
Fully supervised

Fuzzy; definition depends on task

Lazebnik

Good training data?

Good training data?



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

Google guesses from the 1st caption

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?

Slide credit: L. Lazebnik

Generalization Error

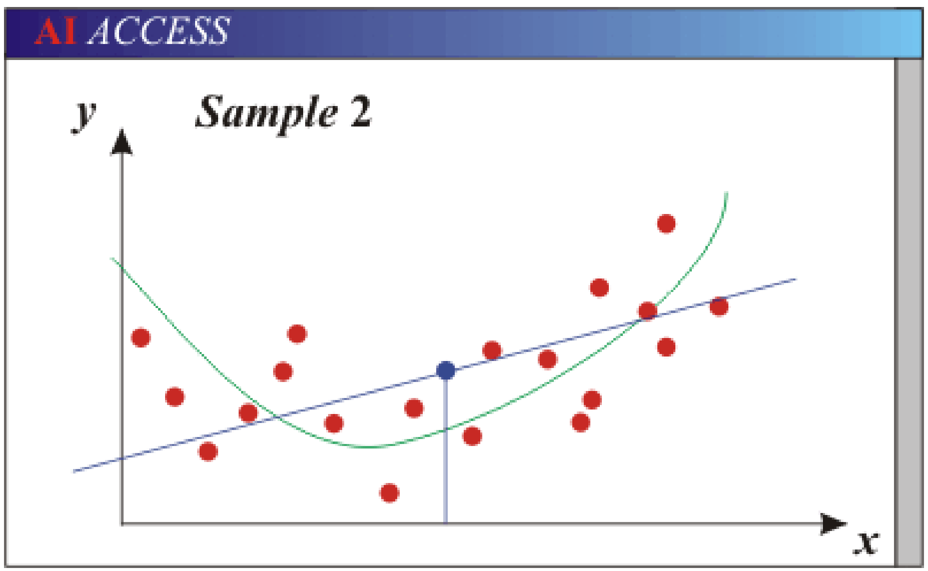
- **Bias:** how much the average model over all training sets differs from the true model.
  - Error due to inaccurate assumptions/simplifications made by the model.
- **Variance:** how much models estimated from different training sets differ from each other.

- **Underfitting:** model is too “simple” to represent all the relevant class characteristics
  - High bias (few degrees of freedom) 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 (many degrees of freedom) and high variance
  - Low training error and high test error

Slide credit: L. Lazebnik

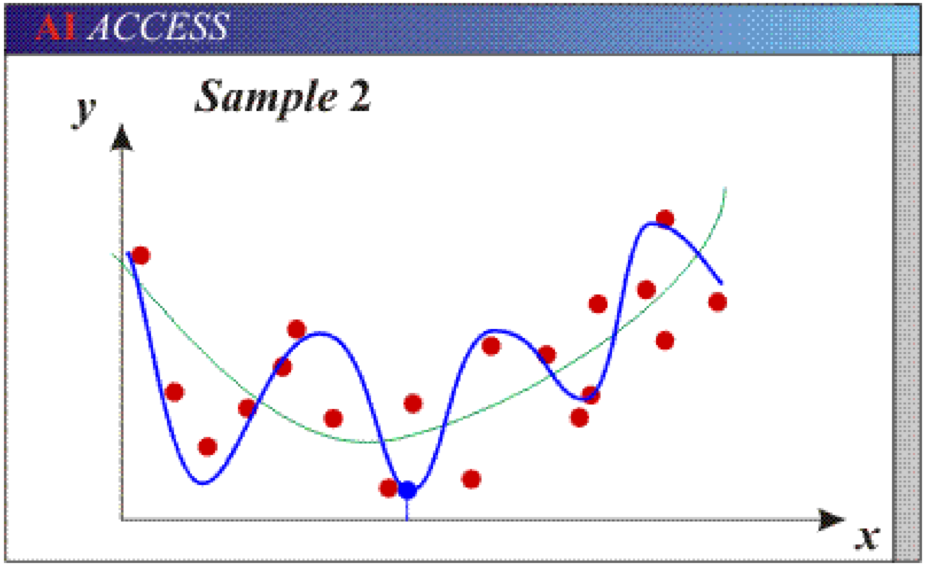
Generalization Error Effects

- **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

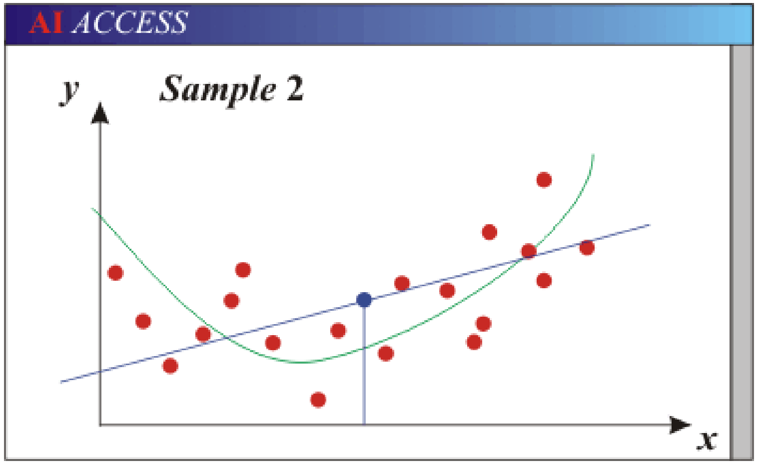


Generalization Error Effects

- **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

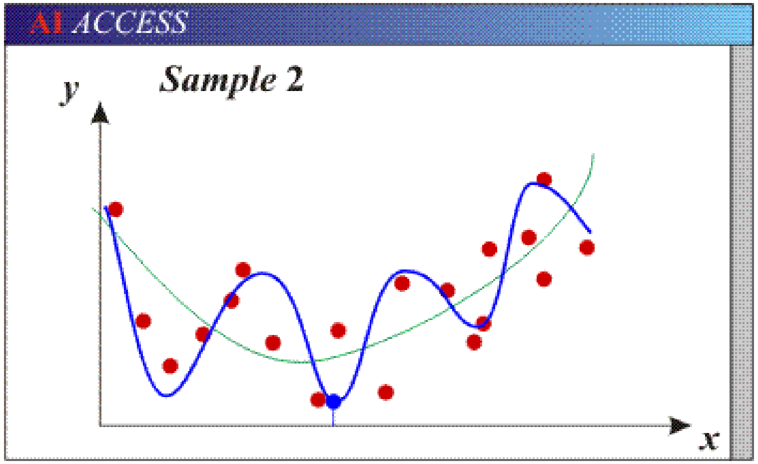


Bias-Variance Trade-off



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

- Not enough flexibility!

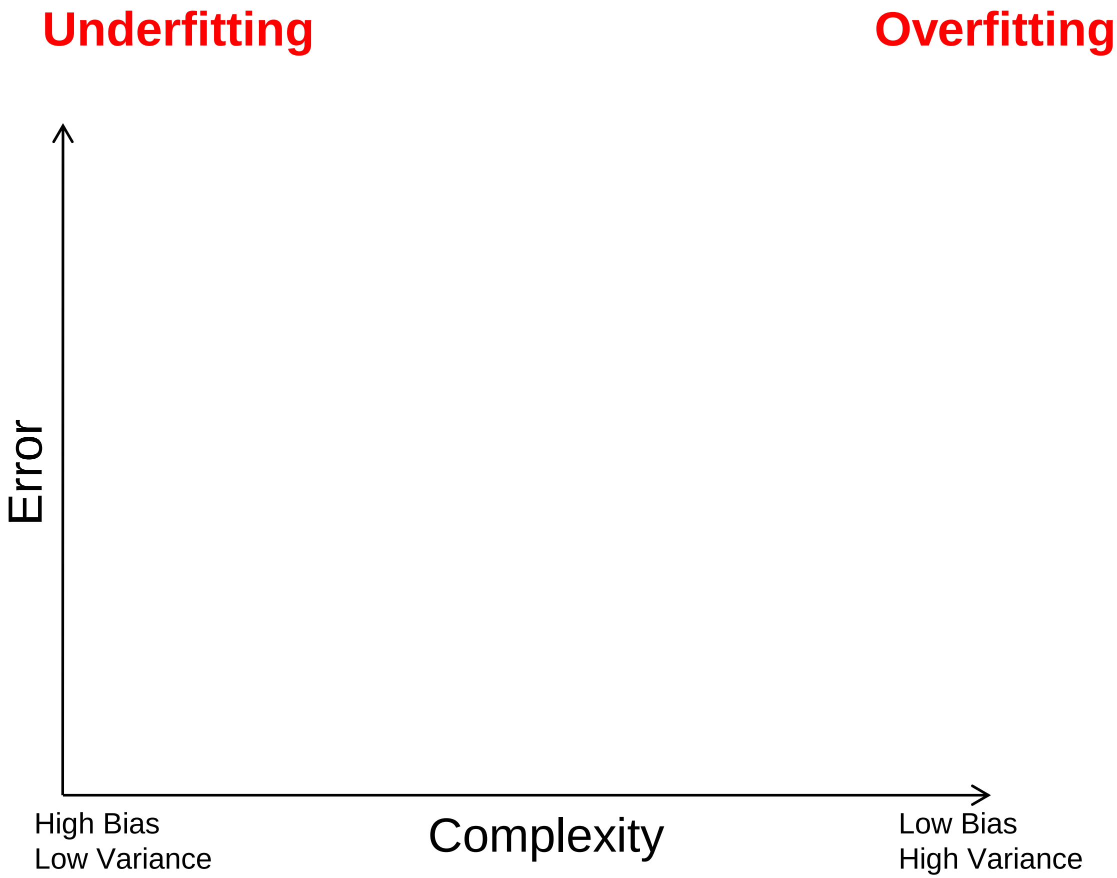


Models with too many parameters are inaccurate because of a large variance.

- Too much sensitivity to the sample.

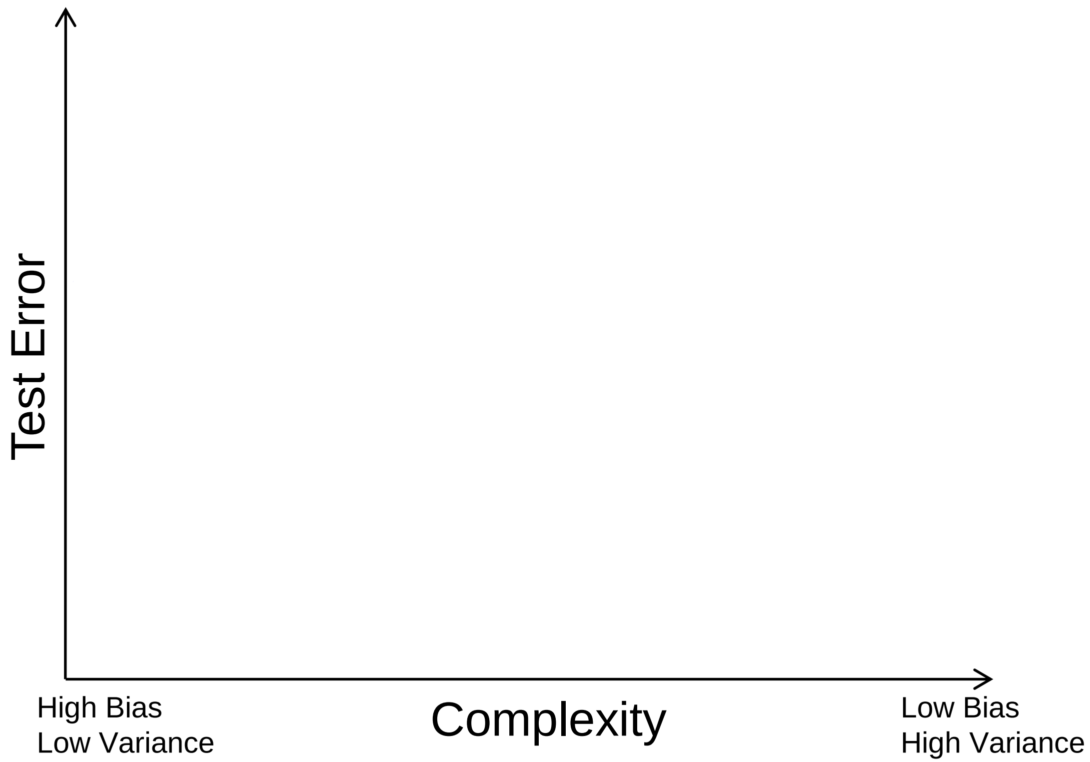
Slide credit: D. Hoiem

Bias-variance tradeoff



Slide credit: D. Hoiem

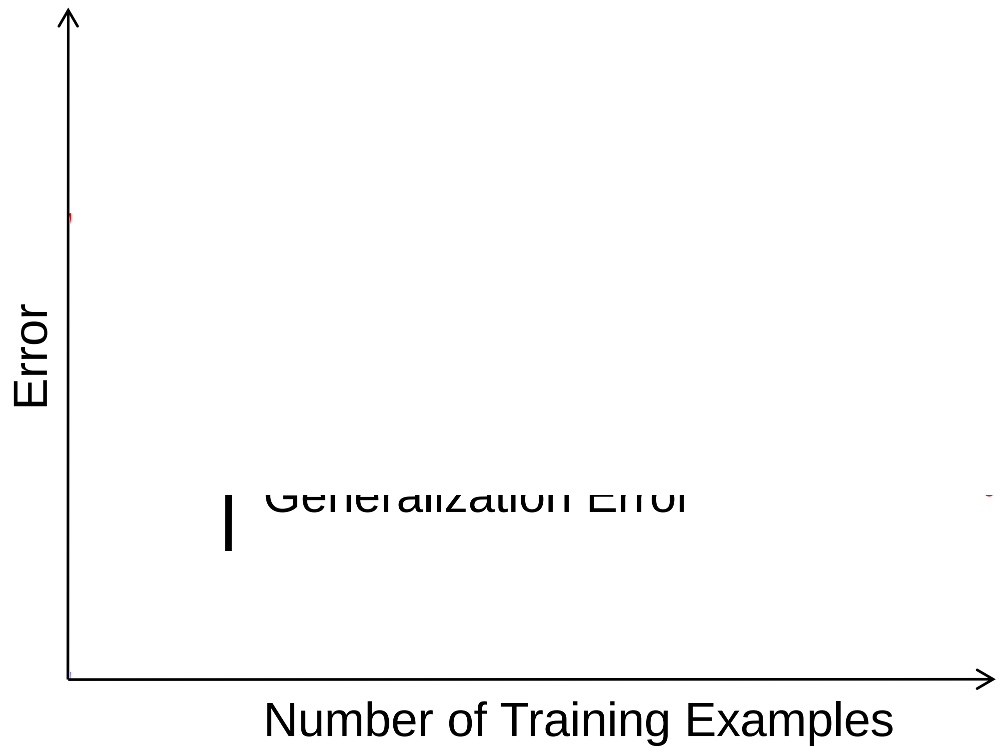
Bias-variance tradeoff



Slide credit: D. Hoiem

Effect of Training Size

Fixed prediction model



Remember…

- 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

How to reduce variance?

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

Slide credit: D. Hoiem

Very brief tour of some classifiers

- **K-nearest neighbor**
- **SVM**
- Boosted Decision Trees
- Neural networks (+CNNs)
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Restricted Boltzmann Machines
- …

Generative vs. Discriminative Classifiers

Discriminative Models

- Learn to directly predict the labels from the data
- Often, assume a simple boundary (e.g., linear)
- Examples
  - Logistic regression
  - SVM
  - Boosted decision trees

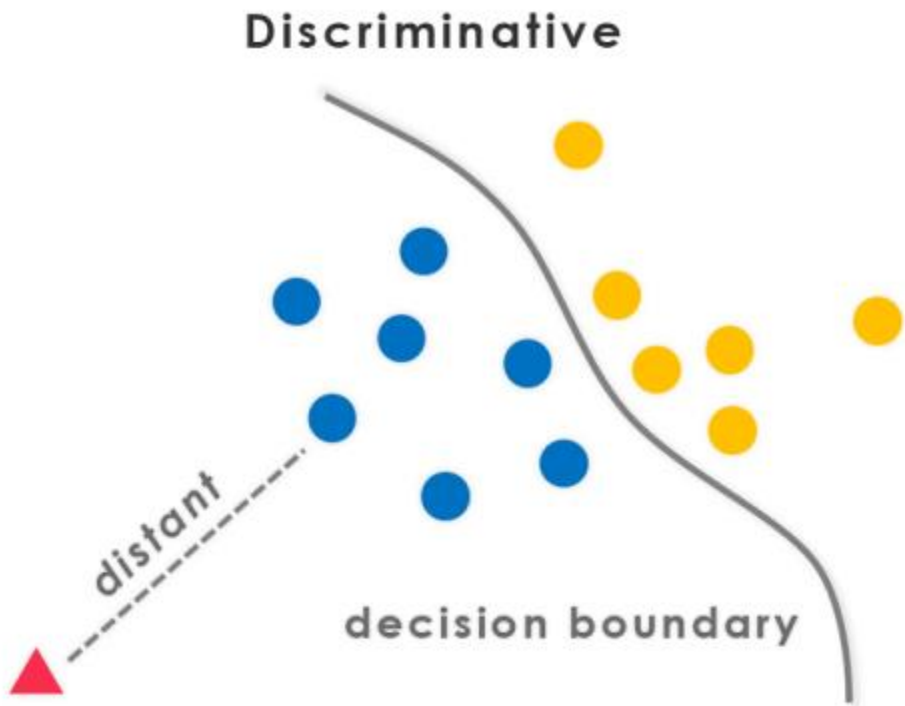
- Often easier to predict a label from the data than to model the data

Generative Models

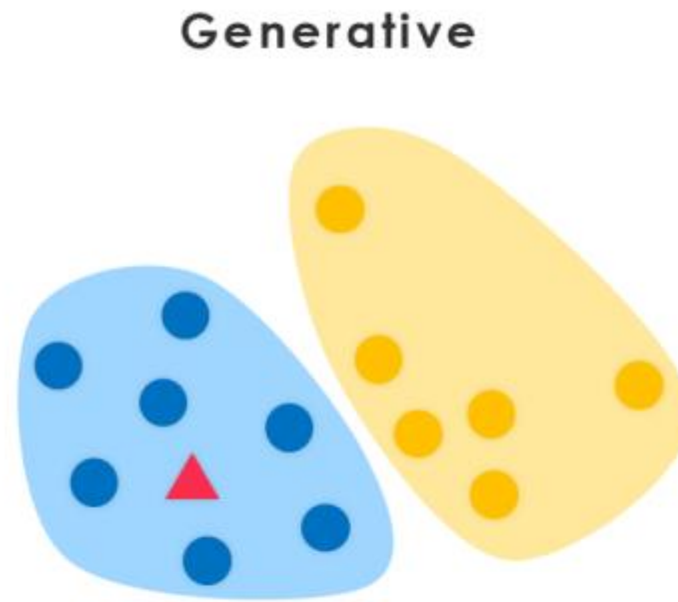
- Represent both the data and the labels
- Often, makes use of conditional independence and priors
- Examples
  - Naïve Bayes classifier
  - Bayesian network

- Models of data may apply to future prediction problems

Slide credit: D. Hoiem



"Learn the data boundary"



"Represent the data + boundary"

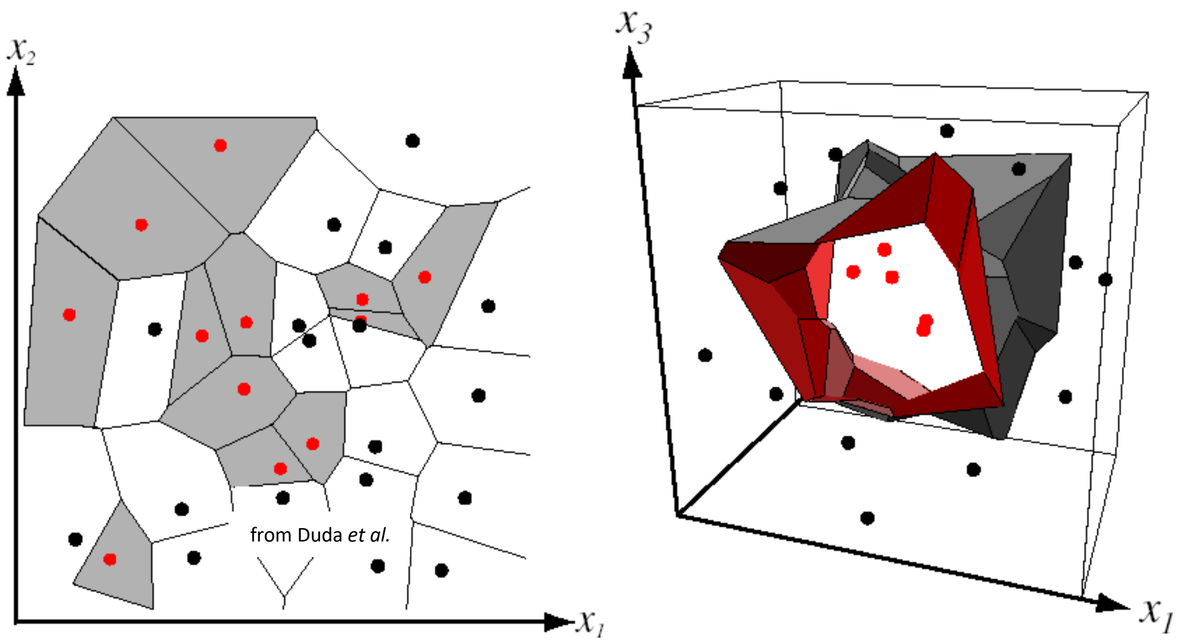
Photo: CMU Machine Learning Department Protests G20

Slides: James Hays, Isabelle Guyon, Erik Sudderth, Mark Johnson, Derek Hoiem

Nearest Neighbor Classifier

Assign label of nearest training data point to each test data point.

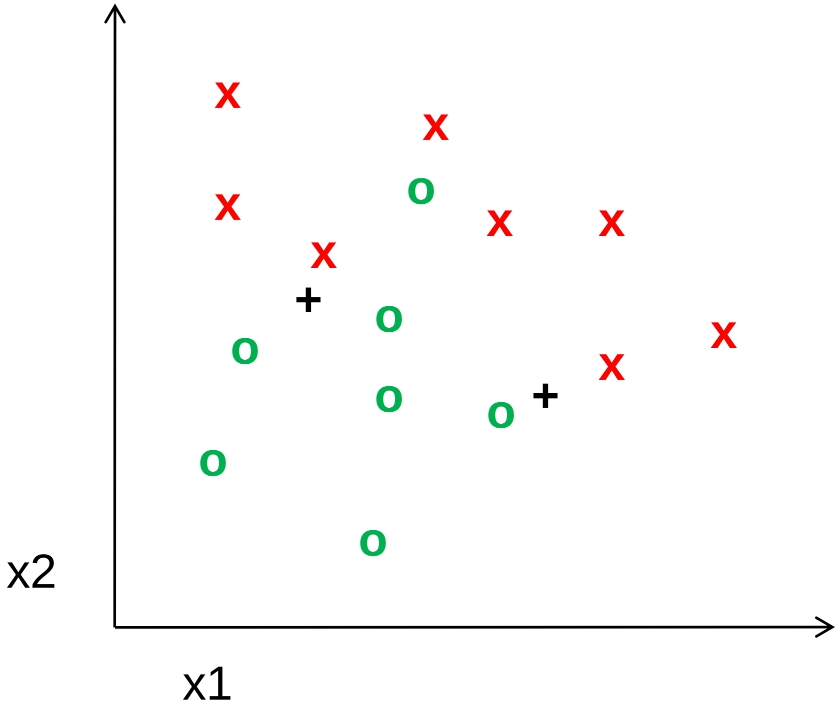
Divides input space into *decision regions* separated by *decision boundaries – Voronoi.*



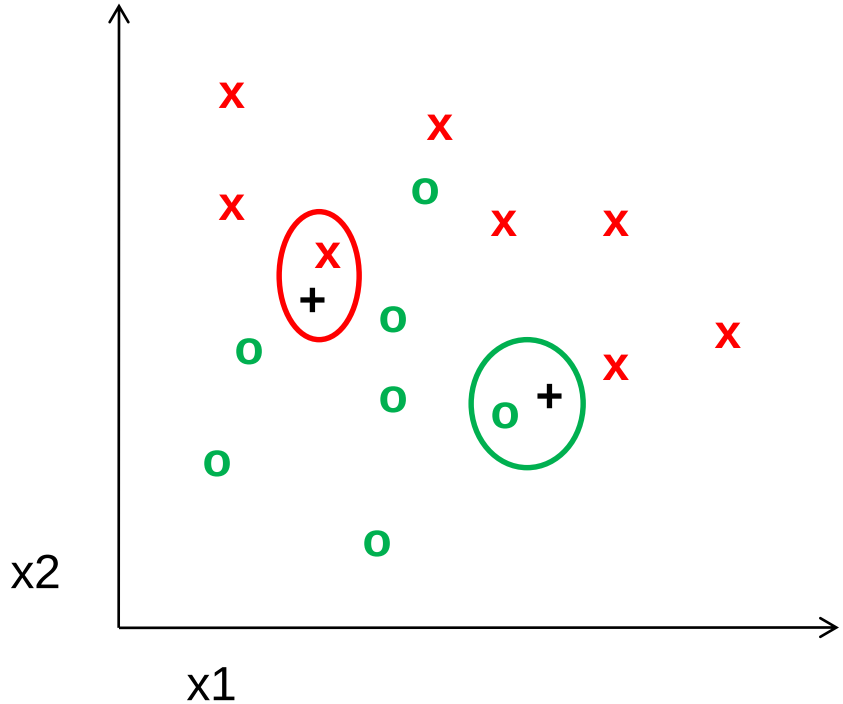
Voronoi partitioning of feature space for two-category 2D and 3D data

Source: D. Lowe

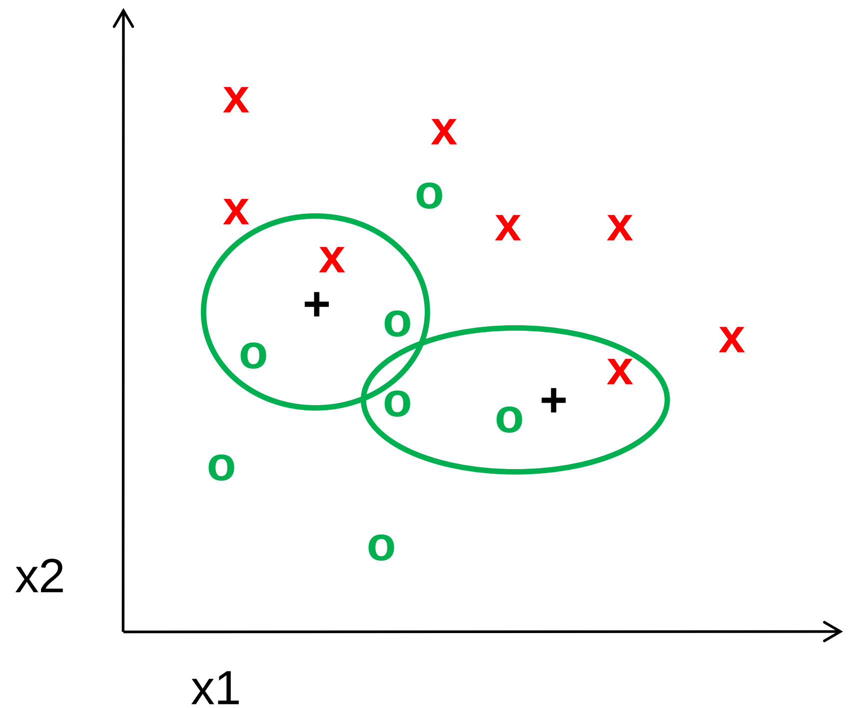
K-nearest neighbor



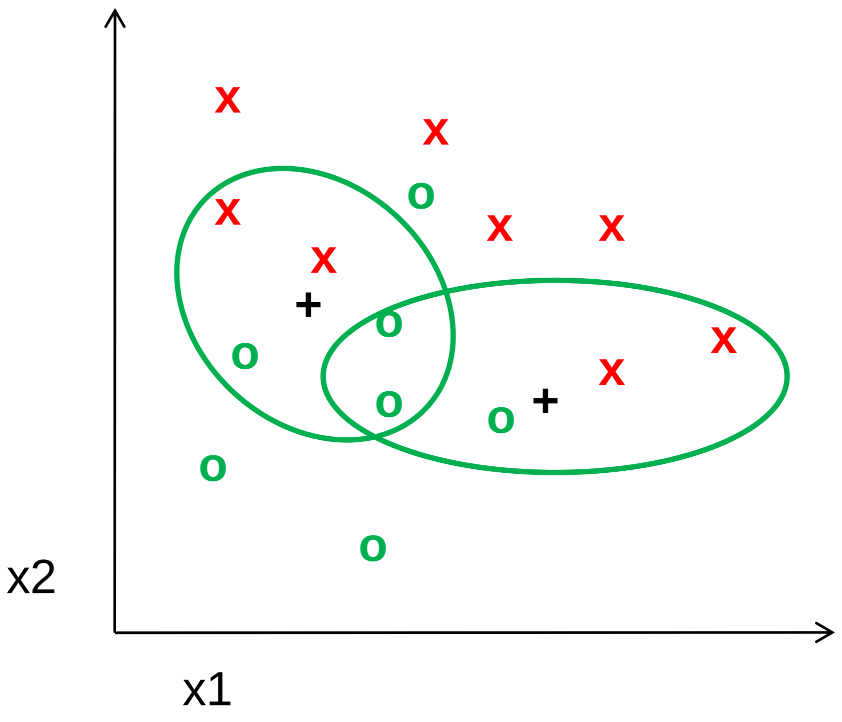
1-nearest neighbor



3-nearest neighbor



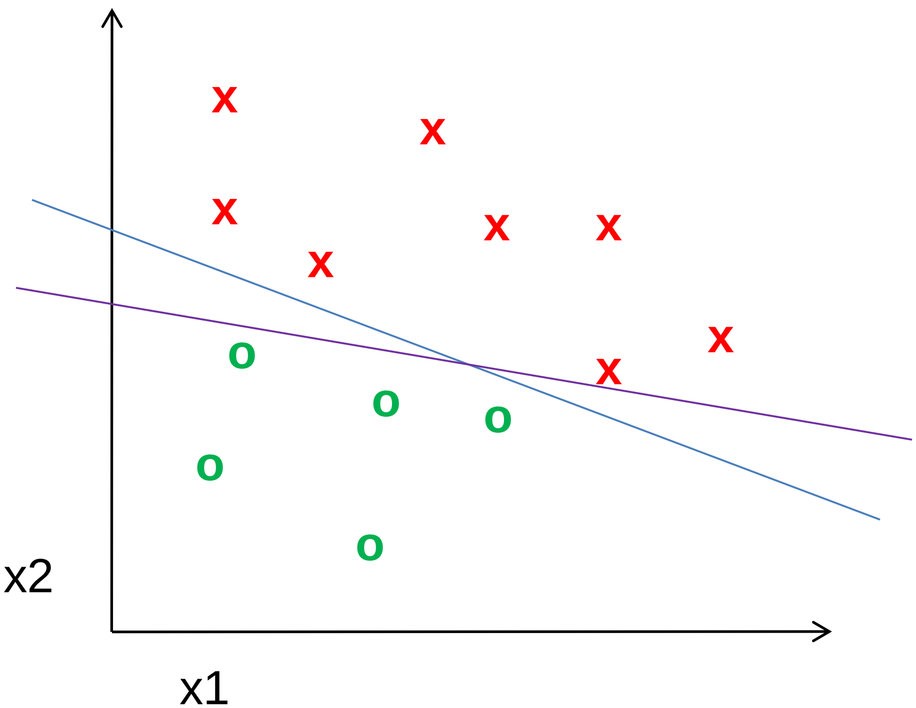
5-nearest neighbor



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

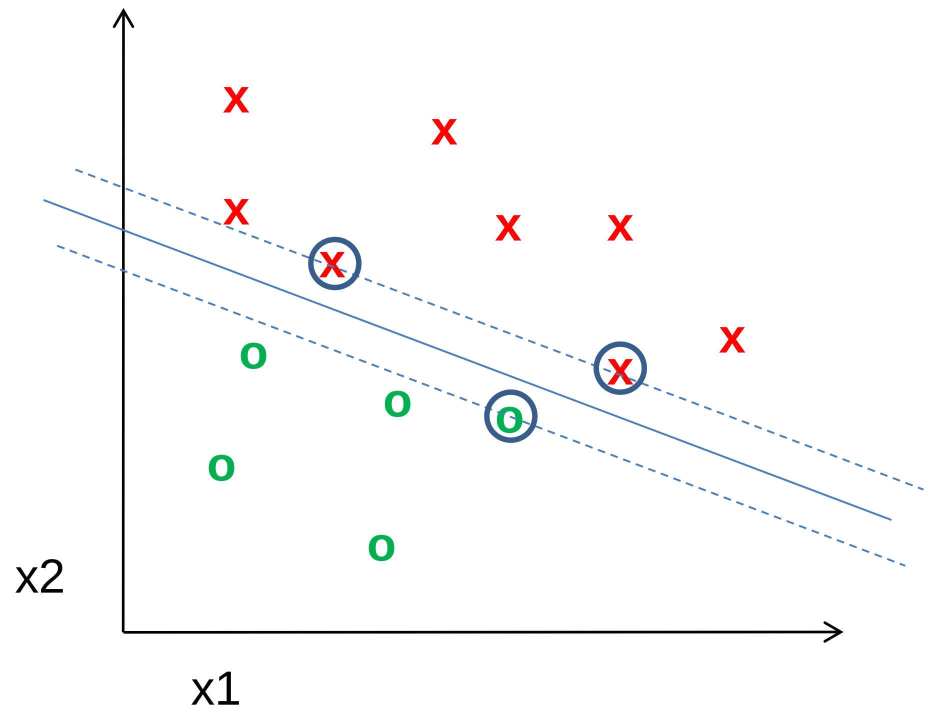
Classifiers: Linear SVM



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

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

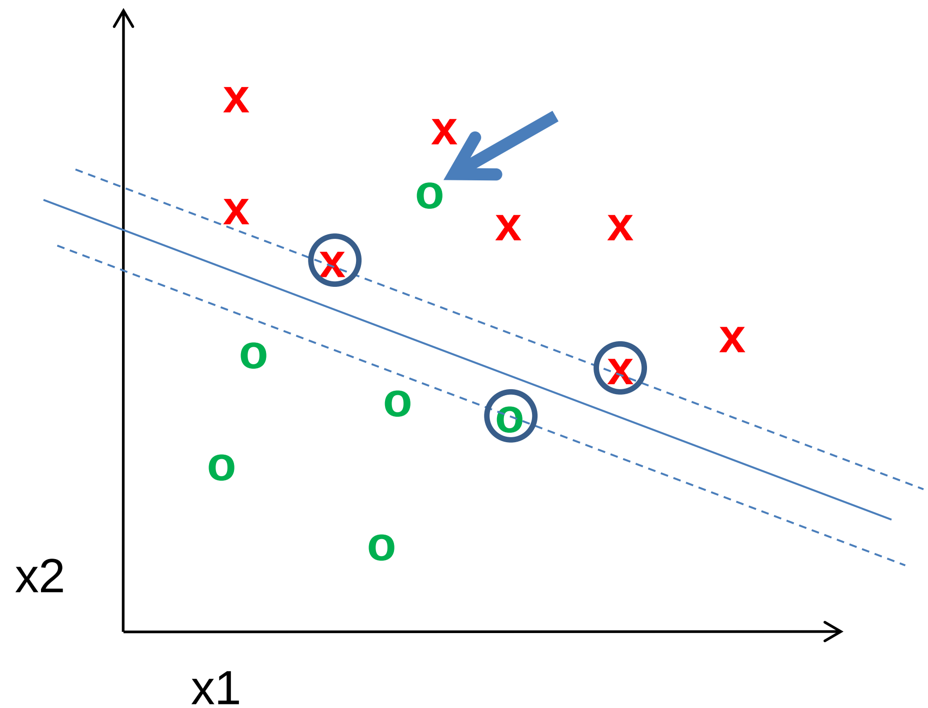
Classifiers: Linear SVM



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

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

Classifiers: Linear SVM



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

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

What about multi-class SVMs?

- Unfortunately, there is no “definitive” multi-class SVM formulation
- In practice, we have to obtain a multi-class SVM by combining multiple two-class SVMs
- One vs. others
  - Traning: 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

What to remember about classifiers

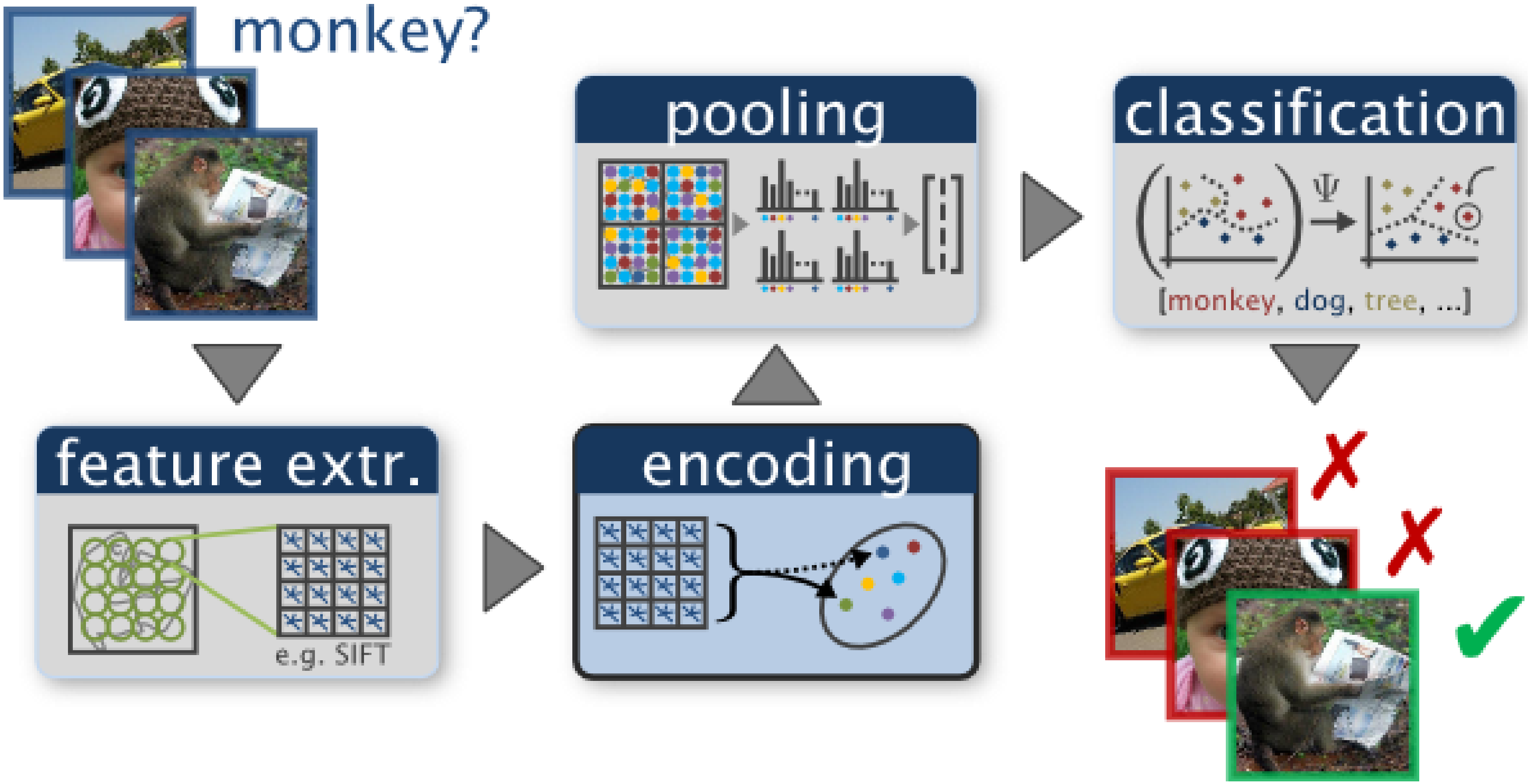
- No free lunch: machine learning algorithms are tools, not dogmas
- Try simple classifiers first
- Better to have smart features and simple classifiers than simple features and smart classifiers
- Use increasingly powerful classifiers with more training data (bias-variance tradeoff)

Slide credit: D. Hoiem

Making decisions about data

- 3 important design decisions:
  1) What data do I use?
  2) How do I represent my data (what feature)?
  3) What classifier / regressor / machine learning tool do I use?
- These are in decreasing order of importance
- Deep learning addresses 2 and 3 simultaneously (and blurs the boundary between them).
- You can take the representation from deep learning and use it with any classifier.

Project 4



Chatfield et al.

Project 4

[Lazebnik et al. 2006]()