

# 2020 COMPUTER VISION



Escher's Circle Limit III



Escher's Circle Limit III

## **Machine Learning Problems**



#### ImageNet

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

• Top 5 error



#### Dataset split



- Train classifier



Measure errorTune modelhyperparameters



- Secret labels
- Measure error

Random train/validate splits = cross validation





#### Features

- Raw pixels
- Histograms
- Templates
- SIFT descriptors
  - GIST
  - ORB
  - HOG....











#### **Testing** Image Apply Image Prediction Features classifier not in training set

#### Slide credit: D. Hoiem and L. Lazebnik

#### Recognition task and supervision

Think-Pair-Share

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?



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

### 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.



#### Google guesses from the 1<sup>st</sup> caption



#### An elephant standing on top of a basket being held by a woman



MS COCO

wordseye.com

Thank you Trent Green



#### **Testing** Image Apply Image Prediction classifier **Features** not in training set

#### 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:



## The machine learning framework f(x) = y f(x) = y

feature

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

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

or classifier

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

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

#### 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!

Quickie Think-Pair-Share: What does the decision boundary look like?

#### 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.



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.

k-nearest neighbor





#### Find a *linear function* to separate the classes

## **Classifiers: Linear SVM**

Find a *linear function* to separate the classes:

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



## **Classifiers: Linear SVM**



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

How?

**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:

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



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:



Andrew Moore

## Nonlinear SVMs

Map the original input space to some higherdimensional feature space where the training set is separable:



## Nonlinear SVMs

The kernel trick: instead of explicitly computing the lifting transformation  $\varphi(\mathbf{x})$ , 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

C. Burges, <u>A Tutorial on Support Vector Machines for Pattern Recognition</u> Data Mining and Knowledge Discovery, 1998 [Additional info]

#### Nonlinear kernel: Example

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



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

## Kernels for bags of features

• Histogram intersection kernel:

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

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,  $\chi^2$  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

## 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

## SVMs: Pros and cons

- Pros
  - Many publicly available SVM packages: <u>http://www.kernel-machines.org/software</u>
  - 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



#### 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



[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

## **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 tradeoff**

Fixed number of training examples

Underfitting

Overfitting



Slide credit: D. Hoiem

#### **Bias-variance tradeoff**

Overfitting



## Effect of Training Size

Fixed complexity prediction model



[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)



Photo: CMU Machine Learning Department Protests G20

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

#### 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.

\*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

#### 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)