SVM wrap-up and Neural Networks

Tues April 25
Kristen Grauman
UT Austin

Last time

• Supervised classification continued
  • Nearest neighbors (wrap up)
  • Support vector machines
    • HoG pedestrians example
    • Understanding classifier mistakes with iHoG
  • Kernels
  • Multi-class from binary classifiers

Today

• Support vector machines (wrap-up)
  • Pyramid match kernels
• Evaluation
  • Scoring an object detector
  • Scoring a multi-class recognition system
• Intro to (deep) neural networks
Recall: Linear classifiers

• Find linear function to separate positive and negative examples

\[ x_\text{positive}: \mathbf{x} \cdot \mathbf{w} + b \geq 0 \]

\[ x_\text{negative}: \mathbf{x} \cdot \mathbf{w} + b < 0 \]

Which line is best?

Recall: Support Vector Machines (SVMs)

• Discriminative classifier based on optimal separating line (for 2d case)

• Maximize the margin between the positive and negative training examples
Recall: Form of SVM solution

- Solution:
  \[ w = \sum a_i y_i x_i \]
  \[ b = y_i - w \cdot x_i \] (for any support vector)

- Classification function:
  \[ f(x) = \text{sign} (w \cdot x + b) = \text{sign} \left( \sum a_i y_i (x_i \cdot x) + b \right) \]

  - If \( f(x) < 0 \), classify as negative.
  - If \( f(x) > 0 \), classify as positive

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

Nonlinear SVMs

- The kernel trick: instead of explicitly computing the lifting transformation \( \phi(x) \), define a kernel function \( K \) such that
  \[ K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \]

- This gives a nonlinear decision boundary in the original feature space:
  \[ \sum a_i y_i K(x_i, x) + b \]

SVMs: Pros and cons

- Pros
  - Kernel-based framework is very powerful, flexible
  - Often a sparse set of support vectors – compact at test time
  - Work very well in practice, even with small training sample sizes

- Cons
  - No "direct" multi-class SVM, must combine two-class SVMs
  - Can be tricky to select best kernel function for a problem
  - 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

Adapted from Lana Lazebnik
Review questions

- What are tradeoffs between the one vs. one and one vs. all paradigms for multi-class classification?
- What roles do kernels play within support vector machines?
- What can we expect the training images associated with support vectors to look like?
- What is hard negative mining?

Scoring a sliding window detector

If prediction and ground truth are **bounding boxes**, when do we have a correct detection?

Kristen Grauman

\[
\frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} > 0.5 \Rightarrow \text{correct}
\]

Kristen Grauman

We'll say the detection is correct (a “true positive”) if the intersection of the bounding boxes, divided by their union, is > 50%.
Scoring an object detector

- If the detector can produce a confidence score on the detections, then we can plot its precision vs. recall as a threshold on the confidence is varied.
- Average Precision (AP): mean precision across recall levels.

Recall: Examples of kernel functions

- Linear: $K(x_i, x_j) = x_i^T x_j$
- Gaussian RBF: $K(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$
- Histogram intersection: $K(x_i, x_j) = \sum \min(x_i(k), x_j(k))$

- Kernels go beyond vector space data
- Kernels also exist for "structured" input spaces like sets, graphs, trees...

Discriminative classification with sets of features?

- Each instance is unordered set of vectors
- Varying number of vectors per instance
Partially matching sets of features

We introduce an approximate matching kernel that makes it practical to compare large sets of features based on their partial correspondences.

Optimal match: $O(m^3)$
Greedy match: $O(m^2 \log m)$
Pyramid match: $O(m)$

($m =$ num pts)

(Previous work: Indyk & Thaper, Bartal, Charikar, Agarwal & Varadarajan, ...)

Pyramid match: main idea

Feature space partitions serve to "match" the local descriptors within successively wider regions.

Histogram intersection counts number of possible matches at a given partitioning.
Pyramid match

\[ K_\Delta(X,Y) = \sum_{i=0}^{L} 2^{-i} \left( I(H_X^{(i)}, H_Y^{(i)}) - I(H_X^{(i-1)}, H_Y^{(i-1)}) \right) \]

- Measures difficulty of a match at level \( j \)
- Normalizes these kernel values to avoid favoring large sets

For similarity, weights inversely proportional to bin size (or may be learned)

- Normalize these kernel values to avoid favoring large sets

[Grauman & Darrell, ICCV 2005]

BoW Issue:

No spatial layout preserved!

Too much?

Too little?
Spatial pyramid match
- Make a pyramid of bag-of-words histograms.
- Provides some loose (global) spatial layout information

\[ K^l(X, Y) = \sum_{m=1}^{M} \kappa^l(x_{m}, y_{m}) \]

Sum over PMKs computed in image coordinate space, one per word.

Spatial pyramid match
- Can capture scene categories well---texture-like patterns but with some variability in the positions of all the local

[Lazebnik, Schmid & Ponce, CVPR 2006]
Spatial pyramid match

- Can capture scene categories well—texture-like patterns but with some variability in the positions of all the local pieces.
- Sensitive to global shifts of the view

Summary: Past week

- Object recognition as classification task
- Boosting (face detection ex)
- Support vector machines and HOG (person detection ex)
  - Pyramid match kernels
  - Hoggles visualization for understanding classifier mistakes
- Nearest neighbors and global descriptors (scene rec ex)
- Sliding window search paradigm
  - Pros and cons
  - Speed up with attentional cascade
- Evaluation
  - Detectors: Intersection over union, precision recall
  - Classifiers: Confusion matrix

Today

- Support vector machines (wrap-up)
  - Pyramid match kernels
- Evaluation
  - Scoring an object detector
  - Scoring a multi-class recognition system
- Intro to (deep) neural networks
Traditional Image Categorization:
Training phase

Training Images ➔ Training Labels ➔ Classifier Training ➔ Trained Classifier ➔ Image Features

Training

Traditional Image Categorization:
Testing phase

Training Images ➔ Training Labels ➔ Classifier Training ➔ Trained Classifier ➔ Image Features ➔ Prediction Outdoor

Testing

Features have been key

SIFT [Lowe IJCV 04]

HOG [Dalal and Triggs CVPR 05]

SPM [Lazebnik et al. CVPR 06]

Textons

and many others:
SURF, MSER, LBP, Color-SIFT, Color histogram, GLOH, ....
Learning a Hierarchy of Feature Extractors

- Each layer of hierarchy extracts features from output of previous layer
- All the way from pixels to classifier
- Layers have the (nearly) same structure
- Train all layers jointly

Learning Feature Hierarchy

Goal: Learn useful higher-level features from images

- Better performance
- Other domains (unclear how to hand engineer):
  - Kinect
  - Video
  - Multi spectral
- Feature computation time
  - Dozens of features now regularly used (e.g., MKL)
  - Getting prohibitive for large datasets (10’s sec/image)
Biological neuron and Perceptrons

A biological neuron

An artificial neuron (Perceptron) - a linear classifier

Slide credit: Jia-Bin Huang

Simple, Complex and Hypercomplex cells

David H. Hubel and Torsten Wiesel

Suggested a hierarchy of feature detectors in the visual cortex, with higher level features responding to patterns of activation in lower level cells, and propagating activation upwards to still higher level cells.

David Hubel’s Eye, Brain, and Vision

Hubel/Wiesel Architecture and Multi-layer Neural Network

Hubel and Weisel’s architecture

Multi-layer Neural Network - A non-linear classifier

Slide credit: Jia-Bin Huang
**Neuron: Linear Perceptron**

- Inputs are feature values
- Each feature has a weight
- Sum is the activation

\[ \text{activation}_{w}(x) = \sum_{i} w_i \cdot f_i(x) = w \cdot f(x) \]

- If the activation is:
  - Positive, output +1
  - Negative, output -1

**Multi-layer Neural Network**

- A non-linear classifier
- **Training:** find network weights \( w \) to minimize the error between true training labels \( y_i \) and estimated labels \( f_w(x_i) \)

\[ E(w) = \sum_{i} (y_i - f_w(x_i))^2 \]

- Minimization can be done by gradient descent provided \( f \) is differentiable
- This training method is called **back-propagation**

**Two-layer perceptron network**
Learning $w$

- Training examples
  \[(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\]

- Objective: a misclassification loss
  \[
  \min_w \sum_{i=1}^{m} \left( y^{(i)} - h_w(f(x^{(i)})) \right)^2
  \]

- Procedure:
  - Gradient descent / hill climbing
Hill climbing

- Simple, general idea:
  - Start wherever
  - Repeat: move to the best neighboring state
  - If no neighbors better than current, quit
  - Neighbors = small perturbations of $w$

- What’s bad?
  - Complete?
  - Optimal?

Two-layer perceptron network

Two-layer perceptron network
Two-layer neural network

Neural network properties

• Theorem (Universal function approximators): A two-layer network with a sufficient number of neurons can approximate any continuous function to any desired accuracy

• Practical considerations:
  • Can be seen as learning the features
  • Large number of neurons
    • Danger for overfitting
  • Hill-climbing procedure can get stuck in bad local optima

Recap

• Pyramid match kernels:
  – Example of structured input data for kernel-based classifiers (SVM)
• Neural networks / multi-layer perceptrons
  – View of neural networks as learning hierarchy of features
Coming up

• Convolutional neural networks for image classification