CAP5415
Computer Vision

Yogesh S Rawat
yogesh@ucf.edu
HEC-241
Administrative details
Course project

• Deadline – 27th October
• Send a short description
• Deliverables
  • Code
    • Well commented
    • Modular
  • Report
    • Describe the problem
    • Describe your approach
    • Evaluation, dataset
    • Results
    • Discussion and your analysis
• Demo – if no other evaluation
  • For example, edge detection in videos
Questions?
Classification - I

Lecture 11
Classification

• Categorizing given set of data into classes
Classification – computer vision

• Digit classification
  • MNIST
  • Multi-mnist
  • SVHN

• Object classification
  • CIFAR-10
  • CIFAR-100
  • ImageNET
Object Recognition

• **Problem:** Given an image A, does A contain an image of a person?
Object Recognition

• **Problem:** Given an image A, does A contain an image of a person?
Object Recognition

• **Problem:** Given an image A, does A contain an image of a person?
Image classification - ImageNet

• Images for each category of WordNet
• 1000 classes
• 1.2M images
• 100K test
• Top 5 error
Dataset split

Training Images
- Train classifier

Validation Images
- Measure error
- Tune model hyperparameters

Testing Images
- Secret labels
- Measure error

Random train/validate splits = cross validation
Training

Images

- Image Features
- Training
- Trained classifier

Testing

- Image not in training set
- Image Features
- Apply classifier
- Prediction

Slide credit: D. Hoiem and L. Lazebnik
Features

- Raw pixels
- Histograms
- Templates
- SIFT
- GIST
- HOG
- SURF
- …

L. Lazebnik
Training

Images

Labels

Image Features

Training

Trained classifier

Testing

Image not in training set

Image Features

Apply classifier

Prediction

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:

\[ f(\text{apple}) = \text{“apple”} \]
\[ f(\text{tomato}) = \text{“tomato”} \]
\[ f(\text{cow}) = \text{“cow”} \]
The machine learning framework

\[ f(x) = y \]

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

\[ \{(x_1, y_1), ..., (x_N, y_N)\} \]

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

**Testing:** Apply \( f \) to an unseen *test example* \( x_u \) and output the predicted value \( y_u = f(x_u) \) to *classify* \( x_u \).

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.
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

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

- All we need is a distance function for our inputs
- No training required!
Pop quiz?
Decision boundary for NN 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

1-nearest

3-nearest

5-nearest
Algorithm

1. Get features for training samples
2. Get features for testing sample
3. For each training sample
   1. Computer distance from testing sample
4. Sort distance vector for all samples
5. Pick classes based on top k samples
K-nearest neighbor

• Pros
  • Simple and easy to implement
  • No training required
  • Classification, regression, search, etc.

• Cons
  • Slow with large samples
  • Slow with high dimensional features
Classifiers: Linear

Find a *linear function* to separate the classes
Motivation

How would you classify this data?

• denotes +1
• denotes -1
How would you classify this data?
Motivation

- denotes +1
- denotes -1

How would you classify this data?
How would you classify this data?

- denotes +1
- denotes -1
Motivation

denotes +1
denotes -1

Any of these would be fine..

..but which is best?
Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.
The maximum margin linear classifier is the linear classifier with the maximum margin. This is the simplest kind of SVM (Called an LSVM)
The maximum margin linear classifier is the linear classifier with the maximum margin. This is the simplest kind of SVM (Called an LSVM).
SVM - background

• A supervised approach for classification and regression
SVM - background

• A supervised approach for classification and regression
  • Developed in the computer science society at 1990s, has grown in popularity since then.
SVM - background

• A supervised approach for classification and regression
  • Developed in the computer science society at 1990s
  • Has grown in popularity since then
  • Shown to perform well in variety of settings
  • And often considered as one of the best “out-of-the box” classifiers

Neural networks were a big deal in early 90s
Kind of delayed Deep Learning era 😊
SVM - background

• An approach for classification
  • Developed in the computer science society at 1990s, has grown in popularity since then.
  • Shown to perform well in variety of settings, and often considered as one of the best “out-of-the box” classifiers.

(Max-margin Classifier) ➔ Support Vector Classifier ➔ Support Vector Machines

(historical appearances in the literature)
Maximal-Margin Classifier

• In a $p$-dimensional space, a hyperplane is a flat affine subspace of dimension $p-1$. 
Maximal-Margin Classifier

- In a $p$-dimensional space, a **hyperplane** is a flat affine subspace of dimension $p-1$.
  - Ex. In 2D, a hyperplane is 1D line,
  - Ex. In 3D, a hyperplane is 2D plane.
Maximal-Margin Classifier

- In a $p$-dimensional space, a hyperplane is a flat affine subspace of dimension $p-1$.
  - Ex. In 2D, a hyperplane is 1D line,
  - Ex. In 3D, a hyperplane is 2D plane.

General hyperplane definition

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$
Maximal-Margin Classifier

• In a $p$-dimensional space, a hyperplane is a flat affine subspace of dimension $p-1$.
  • Ex. In 2D, a hyperplane is 1D line,
  • Ex. In 3D, a hyperplane is 2D plane.

General hyperplane definition

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots \beta_p X_p = 0$$

Hyperplane for 2D data.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$
Maximal-Margin Classifier

• In a $p$-dimensional space, a hyperplane is a flat affine subspace of dimension $p-1$.
  • Ex. In 2D, a hyperplane is 1D line,
  • Ex. In 3D, a hyperplane is 2D plane.

General hyperplane definition

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p = 0$$

Hyperplane for 2D data.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

$$1 + 2X_1 + 3X_2 = 0$$
Max-margin classifier

Left: There are two classes of observations: blue and in purple (each of which has measurements on two variables). Three separating hyperplanes, out of many possible, are shown in black. Right: A separating hyperplane is shown in black: a test observation that falls in the blue portion of the grid will be assigned to the blue class, and a test observation that falls into the purple portion of the grid will be assigned to the purple class.
Max-margin classifier

There are two classes of observations, shown in blue and in purple. The maximal margin hyperplane is shown as a solid line.

The margin is the distance from the solid line to either of the dashed lines. The two blue points and the purple point that lie on the dashed lines are the support vectors, and the distance from those points to the margin is indicated by arrows.

The purple and blue grid indicates the decision rule made by a classifier based on this separating hyperplane.
Construction of Max-Margin Classifier

- Consider n training observations \[ x_1, \ldots, x_n \in \mathbb{R}^p \]
Construction of Max-Margin Classifier

- Consider $n$ training observations $x_1, \ldots, x_n \in \mathbb{R}^p$
- Associated class labels $y_1, \ldots, y_n \in \{-1, +1\}$
Construction of Max-Margin Classifier

- Consider $n$ training observations $x_1, \ldots, x_n \in \mathbb{R}^p$
- Associated class labels $y_1, \ldots, y_n \in \{-1, +1\}$
- Max-margin hyperplane is the solution to:

\[
\begin{align*}
\text{maximize} & \quad M \\
\text{subject to} & \quad \sum_{j=1}^{p} \beta_j^2 = 1, \\
& \quad y_i (\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \ldots, n.
\end{align*}
\]
Construction of Max-Margin Classifier

• Consider n training observations \( x_1, \ldots, x_n \in \mathbb{R}^p \)
• Associated class labels \( y_1, \ldots, y_n \in \{-1, +1\} \)
• Max-margin hyperplane is the solution to:

\[
\begin{aligned}
\text{maximize } & M \\
\text{subject to } & \sum_{j=1}^{p} \beta_j^2 = 1, \\
& y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \quad \forall i = 1, \ldots, n.
\end{aligned}
\]

• It ensures that each observation is on the correct side
  • at least a distance M from the hyperplane
In practice we are sometimes faced with non-linear class boundaries. Support vector classifier or any linear classifier will perform poorly here.

Left: The observations fall into two classes, with a non-linear boundary between them.
Right: The support vector classifier seeks a linear boundary, and consequently performs very poorly.
Nonlinear SVMs

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

\[ \Phi: \mathbf{x} \rightarrow \varphi(\mathbf{x}) \]
Disadvantages of Linear Decision Surfaces

![Graph showing disadvantages of linear decision surfaces with two variables, Var1 and Var2.](image-url)
Advantages of non-linear Decision Surfaces

Var₁

Var₂
**Left:** An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from previous slides, resulting in a far more appropriate decision rule.

**Right:** An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.
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: 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
Questions?

Sources for this lecture include materials from works by Abhijit Mahalanobis, James Tompkin, and Ulas Bagci