Lecture 1
Introduction
Outline

• Course overview
• Introduction to Machine Learning
Course Outline

• Prerequisites
  • Calculus, Statistics, Linear Algebra
  • Some Computer Vision/Image Processing

• Grading
  • Class participation: 10%
  • Four assignments (Matlab): 20%
    • Each assignment is 5%
    • Assignment grades are 0, 40%, 60%, 80%, 100%
  • In class paper presentation  20%
  • Final project: 50%
    • Final Project Presentation 20%
    • Written project report + code, 30 %
    • Matlab, C/C++, anything else as long as I can run it
Course Outline: Content

- Lecture (2/3 of the time), paper discussions (1/3 of the time)
- Machine Learning Topics (tentatively)
  - Nearest neighbor
  - Linear and generalized linear classifiers
  - SVM
  - Boosting
  - Neural Networks
- Computer Vision Topics
  - Image features
  - Mostly detection/recognition
    - object, action, etc
Course Outline: Textbook

• No required textbook, but recommended
  • “Pattern Classification” by R.O. Duda, P.E. Hart and D.G. Stork, second edition
  • “Machine Learning” by Tom M. Mitchell
  • “Pattern Recognition and Machine Learning, by C. Bishop
  • “Machine Learning: a Probabilistic Perspective” by Kevin Patrick Murphy

• Journal/Conference papers
Intro: What is Machine Learning?

• Machine learning is useful when it is too difficult to come up with a program to perform a desired task

• Make computer to learn by showing examples (usually with correct answers)
  - “supervised” learning or learning with a teacher

• In practice: computer program (or function) which has a tunable parameters, tune parameters until the desirable behavior on the examples
Different Types of Learning

• **Supervised Learning**: given training examples of inputs and corresponding outputs, produce the “correct” outputs for new inputs

• **Unsupervised Learning**: given only inputs as training, find structure in the world: e.g. discover “natural” clusters

• **Reinforcement Learning**: not covered in this course
Supervised Machine Learning

- Training samples (or examples) $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$
- Each $\mathbf{x}^i$ is usually multi-dimensional
  - $\mathbf{x}^i_1, \mathbf{x}^i_2, \ldots, \mathbf{x}^i_d$ are called **features**
  - $\mathbf{x}^i$ is also called a **feature vector**
- example
  - $\mathbf{x}^1 = (3, 7, 35)$
  - $\mathbf{x}^2 = (5, 9, 47)$
  - ....
- how many and which features to extract?
- Have target output for each example $\mathbf{y}^1, \mathbf{y}^2, \ldots \mathbf{y}^n$
  - “teacher” gives target outputs
  - $\mathbf{y}^i$ are usually one-dimensional
- example
  - $\mathbf{y}^1 = 1$ (“face”)
  - $\mathbf{y}^2 = 0$ (“not a face”)


Two Types of Supervised Machine Learning

• Classification
  • $y^i$ is finite, typically called a label or a class
  • example: $y^i \in \{"sunny", "cloudy", "raining"\}$

• Regression
  • $y^i$ is continuous, typically called an output value
  • Example: $y^i = \text{temperature} \in [-60,60]$
Toy Application: fish sorting

- classifier
- fish species
- fish image
- sorting chamber
- salmon
- sea bass
Classifer design

• Notice salmon tends to be shorter than sea bass
• Use *fish length* as a feature
• Count number of bass and salmon of each length

<table>
<thead>
<tr>
<th>Length</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>bass</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>8</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>salmon</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

![Graph showing count of bass and salmon by length](image-url)
Single Feature (length) Classifier

- Find the best length $L$ threshold

\[ \text{fish length} < L \quad \text{fish length} > L \]

- classify as salmon
- classify as sea bass

- For example, at $L = 5$, misclassified:
  - 1 sea bass
  - 16 salmon

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>bass</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>8</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>salmon</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Classification error (total error) \( \frac{17}{50} = 34\% \)
• **Tune** parameter $L$ to find the one that performs best
• The best $L = 9$, and still 20% of fish is misclassified
Next Step

- Lesson learned:
  - Length is a poor feature alone!

- What to do?
  - Try another feature
  - Salmon tends to be lighter
  - Try average fish lightness
• Now fish are classified best at lightness threshold of 3.5 with classification error of 8%
Can do better by feature combining

- Use both length and lightness features
- Feature vector \([\text{length, lightness}]\)
- Find linear boundary that separates training samples

- Classification error 4%
Even Better Decision Boundary

- Decision boundary (wiggly) with 0% error
Test Classifier on New Data

- The goal is for classifier to perform well on new data
- Test “wiggly” classifier on new data: 25% error
What Went Wrong?

- Have only a limited amount of data for training
- Should ensure decision boundary does not adapt too closely to the particulars of training data, but grasps the “big picture”
- Smoother (simpler) decision boundaries tend to generalize better to new data
• Complicated boundaries *overfit* the data, they are too tuned to the particular training data at hand
• Therefore complicated boundaries tend to not *generalize* well to the new data
• Usually refer to the new data as *test* data
Overfitting: Extreme Example

• Say we have 2 classes: face and non-face images
• Memorize (i.e. store) all the “face” images
• For a new image, see if it is one of the stored faces
  • if yes, output “face” as the classification result
  • If no, output “non-face”
  • also called “rote learning”

• **problem**: new “face” images are different from stored “face” examples
  • zero error on stored data, 50% error on test (new) data
  • decision boundary is very unsmooth
• Rote learning is memorization without generalization
The ability to produce correct outputs on previously unseen examples is called **generalization**.

Big question of learning theory: how to get good generalization with a limited number of examples.

Intuitive idea: favor simpler classifiers.

- William of Occam (1284-1347): “entities are not to be multiplied without necessity”

Simpler decision boundary may not fit ideally to the training data but tends to generalize better to new data.
Underfitting

• Can also underfit data, i.e. too simple decision boundary
  • chosen model is not expressive enough
• No linear decision boundary can well separate the samples
• Training error is too high
  • test error is, of course, also high
Underfitting $\rightarrow$ Overfitting

**underfitting**
- high training error
- high test error

**“just right”**
- low training error
- low test error

**overfitting**
- low training error
- high test error
Sketch of Supervised Machine Learning

- Chose a learning machine $f(x,w)$
  - $w$ are tunable weights
  - $x$ is the input sample
  - $f(x,w)$ should output the correct class of sample $x$
  - use labeled samples to tune weights $w$ so that $f(x,w)$ give the correct label for sample $x$

- Which function $f(x,w)$ do we choose?
  - has to be expressive enough to model our problem well, i.e. to avoid **underfitting**
  - yet not to complicated to avoid **overfitting**
Training and Testing

• There are 2 phases, training and testing
  • Divide all labeled samples \( x^1, x^2, \ldots, x^n \) into 2 sets, \textit{training} set and \textit{test} set
  • Training phase is for “teaching” machine
    • tuning weights \( w \)
  • Testing phase is for evaluating how well machine works on unseen examples
More on Training Phase

- Find the weights $\mathbf{w}$ s.t. $f(x^i, \mathbf{w}) = y^i$ “as much as possible” for training samples $(x^i, y^i)$
  - “as much as possible” needs to be defined

- How do we tune parameters $\mathbf{w}$ to ensure $f(x^i, \mathbf{w}) = y^i$ for most training samples $(x^i, y^i)$?
  - This step is usually done by optimization, can be quite time consuming
More on Testing Phase

• The goal is to design machine which performs well on unseen examples
• Evaluate performance of the trained machine $f(x,w)$ on the test samples (unseen labeled samples)
• Testing the machine on unseen labeled examples lets us approximate how well it will perform in practice
• If testing results are poor, go back to training phase
  • add more features (if underfitting)
  • remove features (if overfitting)
  • or redesign $f(x,w)$
  • or collect more training data
Classification System Design Overview

• Collect and label data by hand
  - salmon  sea bass  salmon  salmon  sea bass  sea bass

• Preprocess by segmenting fish from background

• Extract possibly discriminating features
  - length, lightness, width, number of fins, etc.

• Classifier design
  - Choose model for classifier
  - Train classifier on training data

• Test classifier on test data

we mostly look at these two steps in this course
Application: Face Detection

- Objects – image patches
- Classes – “face” and “not face”
Optical character recognition (OCR)

- Objects – images or image patches
- Classes – digits 0, 1, ..., 9

Digit recognition, AT&T labs
http://www.research.att.com/~yann/

License plate readers
http://en.wikipedia.org/wiki/Automatic_number_plate_recognition

Slide Credit: D. Hoiem
Smile detection

The Smile Shutter flow

Imagine a camera smart enough to catch every smile! In Smile Shutter Mode, your Cyber-shot® camera can automatically trip the shutter at just the right instant to catch the perfect expression.
Object recognition in mobile phones

Point & Find, Nokia
Google Goggles
Interactive Games: Kinect

- Object Recognition: [http://www.youtube.com/watch?feature=iv&v=fQ59dXOo63o](http://www.youtube.com/watch?feature=iv&v=fQ59dXOo63o)
- Mario: [http://www.youtube.com/watch?v=8CTJL5IujHg](http://www.youtube.com/watch?v=8CTJL5IujHg)
- 3D: [http://www.youtube.com/watch?v=7QrnwoO1-8A](http://www.youtube.com/watch?v=7QrnwoO1-8A)
- Robot: [http://www.youtube.com/watch?v=w8BmgfMKFbY](http://www.youtube.com/watch?v=w8BmgfMKFbY)

Slide Credit: D. Hoiem
Application: Scene Classification

- Objects – images
- Classes – “mountain”, “lake”, “field”...
Application: Medical Image Processing

- Objects – pixels
- Classes – different tissue types, stroma, lumen, etc.