Machine Learning Crash Course

Computer Vision

Jia-Bin Huang, Virginia Tech

Many slides from D. Hoiem, J. Hays
Administrative stuffs

• HW 4
  • Due 11:59pm on Wed, November 2\textsuperscript{nd}
• What is a category?

• Why would we want to put an image in one?
  To predict, describe, interact. To organize.

• Many different ways to categorize
Examples of Categorization in Vision

• Part or object detection
  • E.g., for each window: face or non-face?

• Scene categorization
  • Indoor vs. outdoor, urban, forest, kitchen, etc.

• Action recognition
  • Picking up vs. sitting down vs. standing ...

• Emotion recognition

• Region classification
  • Label pixels into different object/surface categories

• Boundary classification
  • Boundary vs. non-boundary

• Etc, etc.
Image Categorization

Training

Training Images

Training Labels

Image Features

Classifier Training

Trained Classifier

Testing

Test Image

Image Features

Trained Classifier

Prediction

Outdoor
Feature design is paramount

- Most features can be thought of as templates, histograms (counts), or combinations

- Think about the right features for the problem
  - Coverage
  - Concision
  - Directness
Today’s class: Machine Learning

• Machine learning overview

• Unsupervised Learning
  • Dimensionality reduction
  • Clustering

• Supervised Learning
  • Classification
  • Regression
• “If you were a current computer science student what area would you start studying heavily?”
  • Answer: Machine Learning.
  • “The ultimate is computers that learn”
  • Bill Gates, Reddit AMA

• “Machine learning is the next Internet”
  • Tony Tether, Director, DARPA

• “Machine learning is today’s discontinuity”
  • Jerry Yang, CEO, Yahoo
Google snaps up object recognition startup

Machine Learning Startup Acquired by ai-one

Press Release
For Immediate Release:  August 4, 2011

San Diego artificial intelligence startup acquired by leading
learning SDKs as market for advanced

Microsoft acquires legal-focused machine-learning vendor Equivio

Summary: Microsoft has purchased Equivio, maker of a machine-learning platform for the legal industry, for an undisclosed amount.

Microsoft has purchased Equivio, an eDiscovery/compliance vendor with a specialization in text analysis, for an undisclosed amount.

Microsoft officials announced the acquisition of the Israeli company -- its first acquisition of 2015 using more of its offshore cash -- on January 20.

Update: The Wall Street Journal reported back in October last year that Microsoft planned to buy Equivio for $200 million.

Update No. 2: A Microsoft spokesperson said the $200 million estimate was inflated and incorrect, but declined to provide a different figure.

DeepMind

DeepMind is a cutting edge artificial intelligence company. We combine the best techniques from machine learning and systems neuroscience to build powerful general-purpose learning algorithms.

Founded by Demis Hassabis, Shane Legg and Mustafa Suleyman, the company is based in London and supported by some of the most iconic technology entrepreneurs and investors of the past decade. Our first commercial ...
Machine Learning:
Making predictions or decisions from Data
Resources

• Disclaimer:
  • This overview will not cover statistical underpinnings of learning methods. We’ve looking at ML as a tool.

• ML related courses at Virginia Tech
  • ECE 5424 / 4424 - CS 5824 / 4824 Introduction to Machine Learning
  • CS 4804 Introduction to Artificial Intelligence
  • ECE 6504 Neural Networks and Deep Learning

• External courses
  • Machine Learning by Andrew Ng
    [https://www.coursera.org/learn/machine-learning](https://www.coursera.org/learn/machine-learning)
  • Learning from Data by Yaser S. Abu-Mostafa
Impact of Machine Learning

• Machine Learning is arguably the greatest export from computing to other scientific fields.
Machine Learning Applications

- High Energy Physics
- Market Analysis
- Machine Vision
- Text Categorization
- OCR HWR
- Bioinformatics
- System diagnosis

Training examples vs inputs graph.

Genomics
Proteomics

Slide: Isabelle Guyon
Dimensionality Reduction

- **PCA, ICA, LLE, Isomap**

- PCA is the most important technique to know. It takes advantage of correlations in data dimensions to produce the best possible lower dimensional representation based on linear projections (minimizes reconstruction error).

- PCA should be used for dimensionality reduction, not for discovering patterns or making predictions. Don't try to assign semantic meaning to the bases.
Eigenfaces example

Mean: $\mu$

Top eigenvectors: $u_1, \ldots, u_k$
Machine Learning Problems

Supervised Learning

- classification or categorization

Unsupervised Learning

- clustering
- dimensionality reduction

Discrete

Continuous

regression
Clustering

• Clustering:
  • group together similar points and represent them with a single token

• Key Challenges:
  • What makes two points/images/patches similar?
  • How do we compute an overall grouping from pairwise similarities?
Why do we cluster?

• **Summarizing data**
  – Look at large amounts of data
  – Patch-based compression or denoising
  – Represent a large continuous vector with the cluster number

• **Counting**
  – Histograms of texture, color, SIFT vectors

• **Segmentation**
  – Separate the image into different regions

• **Prediction**
  – Images in the same cluster may have the same labels
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 pdf

• **Spectral clustering**
  – Split the nodes in a graph based on assigned links with similarity weights
# Machine Learning Problems

<table>
<thead>
<tr>
<th>Supervised Learning</th>
<th>Unsupervised Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete</td>
<td>clustering</td>
</tr>
<tr>
<td>classification or categorization</td>
<td>dimensionality reduction</td>
</tr>
<tr>
<td>Continuous</td>
<td></td>
</tr>
<tr>
<td>regression</td>
<td></td>
</tr>
</tbody>
</table>
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

\[ y = f(x) \]

- **Training**: given a *training set* of labeled examples \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \), estimate the prediction function \( f \) by minimizing the prediction error on the training set.

- **Testing**: apply \( f \) to a never before seen *test example* \( x \) and output the predicted value \( y = f(x) \).
A classifier maps from the feature space to a label.
Different types of classification

• **Exemplar-based**: transfer category labels from examples with most similar features
  • What similarity function? What parameters?

• **Linear classifier**: confidence in positive label is a weighted sum of features
  • What are the weights?

• **Non-linear classifier**: predictions based on more complex function of features
  • What form does the classifier take? Parameters?

• **Generative classifier**: assign to the label that best explains the features (makes features most likely)
  • What is the probability function and its parameters?

Note: You can always fully design the classifier by hand, but usually this is too difficult. Typical solution: learn from training examples.
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

• Goal of training is to learn feature weights or distance measures so that visual similarity predicts label similarity

• *We want the simplest function that is confidently correct*
Exemplar-based Models

• Transfer the label(s) of the most similar training examples
K-nearest neighbor classifier
1-nearest neighbor
3-nearest neighbor
5-nearest neighbor
Using K-NN

• Simple, a good one to try first

• Higher K gives smoother functions

• No training time (unless you want to learn a distance function)

• With infinite examples, 1-NN provably has error that is at most twice Bayes optimal error
Discriminative classifiers

Learn a simple function of the input features that confidently predicts the true labels on the training set

\[ y = f(x) \]

Training Goals
1. Accurate classification of training data
2. Correct classifications are confident
3. Classification function is simple
Classifiers: Logistic Regression

- Objective
- Parameterization
- Regularization
- Training
- Inference

The objective function of most discriminative classifiers includes a loss term and a regularization term.

\[
\log \frac{P(x_1, x_2 \mid y = 1)}{P(x_1, x_2 \mid y = -1)} = w^T x
\]

\[
P(y = 1 \mid x_1, x_2) = \frac{1}{1 + \exp(-w^T x)}
\]
Using Logistic Regression

• Quick, simple classifier (good one to try first)

• Use L2 or L1 regularization
  • L1 does feature selection and is robust to irrelevant features but slower to train
Classifiers: Linear SVM
Classifiers: Kernelized SVM
Using SVMs

• Good general purpose classifier
  • Generalization depends on margin, so works well with many weak features
  • No feature selection
  • Usually requires some parameter tuning

• Choosing kernel
  • Linear: fast training/testing – start here
  • RBF: related to neural networks, nearest neighbor
  • Chi-squared, histogram intersection: good for histograms (but slower, esp. chi-squared)
  • Can learn a kernel function
Classifiers: Decision Trees
Discrete AdaBoost (Freund & Schapire 1996b)

1. Start with weights $w_i = 1/N$, $i = 1, \ldots, N$.

2. Repeat for $m = 1, 2, \ldots, M$:
   
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights $w_i$ on the training data.
   
   (b) Compute $err_m = E_w[1_{y \neq f_m(x)}]$, $c_m = \log((1 - err_m)/err_m)$.
   
   (c) Set $w_i \leftarrow w_i \exp[c_m \cdot 1_{y_i \neq f_m(x_i)}]$, $i = 1, 2, \ldots N$, and renormalize so that $\sum_i w_i = 1$.

3. Output the classifier $\text{sign}[\sum_{m=1}^{M} c_m f_m(x)]$
Boosted Decision Trees

\[ P(label \mid \text{good segment, data}) \]

[Collins et al. 2002]
Using Boosted Decision Trees

• Flexible: can deal with both continuous and categorical variables
• How to control bias/variance trade-off
  • Size of trees
  • Number of trees
• Boosting trees often works best with a small number of well-designed features
• Boosting “stubs” can give a fast classifier
Generative classifiers

• Model the joint probability of the features and the labels
  • Allows direct control of independence assumptions
  • Can incorporate priors
  • Often simple to train (depending on the model)

• Examples
  • Naïve Bayes
  • Mixture of Gaussians for each class
Naïve Bayes

- Objective
- Parameterization
- Regularization
- Training
- Inference
Using Naïve Bayes

• Simple thing to try for categorical data

• Very fast to train/test
Many classifiers to choose from

• SVM
• Neural networks
• Naïve Bayes
• Bayesian network
• Logistic regression
• Randomized Forests
• Boosted Decision Trees
• K-nearest neighbor
• RBMs
• Deep networks
• Etc.

Which is the best one?
No Free Lunch Theorem
Generalization Theory

• It’s not enough to do well on the training set: we want to also make good predictions for new examples
Bias-Variance Trade-off

\[ E(\text{MSE}) = \text{noise}^2 + \text{bias}^2 + \text{variance} \]

- Unavoidable error
- Error due to incorrect assumptions
- Error due to variance parameter estimates from training samples

See the following for explanation of bias-variance (also Bishop’s “Neural Networks” book):
http://www.inf.ed.ac.uk/teaching/courses/mlsc/Notes/Lecture4/BiasVariance.pdf
Bias and Variance

Error = noise^2 + bias^2 + variance
Choosing the trade-off

- Need validation set
- Validation set is separate from the test set
Effect of Training Size

Fixed classifier

Generalization Error

Number of Training Examples

Error
How to reduce variance?

• Choose a simpler classifier
• Regularize the parameters
• Use fewer features
• Get more training data

Which of these could actually lead to greater error?
Reducing Risk of Error

- Margins
The perfect classification algorithm

• Objective function: encodes the right loss for the problem

• Parameterization: makes assumptions that fit the problem

• Regularization: right level of regularization for amount of training data

• Training algorithm: can find parameters that maximize objective on training set

• Inference algorithm: can solve for objective function in evaluation
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Learning Objective</th>
<th>Training</th>
<th>Inference</th>
</tr>
</thead>
</table>
| Naïve Bayes          | $\text{maximize } \sum_i \left[ \sum_j \log P(x_{ij} \mid y_i; \theta_j) + \log P(y_i; \theta_0) \right]$                                                                 | $\theta_{kj} = \frac{\sum_i \delta(x_{ij} = 1 \land y_i = k) + r}{\sum_i \delta(y_i = k) + Kr}$                                | $\theta^T x + \theta_0^T (1-x) > 0$  
where $\theta_{ij} = \log \frac{P(x_{ij} = 1 \mid y_i = 1)}{P(x_{ij} = 1 \mid y_i = 0)}$,  
$\theta_{0j} = \log \frac{P(x_{ij} = 0 \mid y_i = 1)}{P(x_{ij} = 0 \mid y_i = 0)}$  
$\theta^T x > t$ |
| Logistic Regression  | $\text{maximize } \sum_i \log \left( P(y_i \mid x, \theta) \right) + \lambda \| \theta \|$                                                                 | Gradient ascent                                                          | $\theta^T x > t$                                                                                                                     |
|                      | where $P(y_i \mid x, \theta) = \frac{1}{1 + \exp(-y_i \theta^T x)}$                                                                                      |                                                                          |                                                                                                                                 |
| Linear SVM           | $\text{minimize } \lambda \sum_i \xi_i + \frac{1}{2} \| \theta \|$                                                                                     | Quadratic programming or subgradient opt.                                 | $\theta^T x > t$                                                                                                                     |
|                      | such that $y_i \theta^T x \geq 1 - \xi_i \land i, \xi_i \geq 0$                                                                                       |                                                                          |                                                                                                                                 |
| Kernelized SVM       | complicated to write                                                                                                                                 | Quadratic programming                                                    | $\sum_i y_i \alpha_i K(\hat{x}_i, x) > 0$                                                                                                                                                   |
| Nearest Neighbor     | most similar features $\rightarrow$ same label                                                                                                         | Record data                                                              | $y_i$  
where $i = \arg \min \limits_i K(\hat{x}_i, x)$                                                                                                                                         |
Characteristics of vision learning problems

• Lots of continuous features
  • E.g., HOG template may have 1000 features
  • Spatial pyramid may have ~15,000 features

• Imbalanced classes
  • Often limited positive examples, practically infinite negative examples

• Difficult prediction tasks
When a massive training set is available

• Relatively new phenomenon
  • MNIST (handwritten letters) in 1990s, LabelMe in 2000s, ImageNet (object images) in 2009, ...

• Want classifiers with low bias (high variance ok) and reasonably efficient training

• Very complex classifiers with simple features are often effective
  • Random forests
  • Deep convolutional networks
New training setup with moderate sized datasets

- **Training Images**
- **Training Labels**
- **Tune CNN features and Neural Network classifier**
- **Trained Classifier**

Dataset similar to task with millions of labeled examples

- **Initialize CNN Features**
Practical tips

• Preparing features for linear classifiers
  • Often helps to make zero-mean, unit-dev
  • For non-ordinal features, convert to a set of binary features

• Selecting classifier meta-parameters (e.g., regularization weight)
  • Cross-validation: split data into subsets; train on all but one subset, test on remaining; repeat holding out each subset
    • Leave-one-out, 5-fold, etc.

• Most popular classifiers in vision
  • SVM: linear for when fast training/classification is needed; performs well with lots of weak features
  • Logistic Regression: outputs a probability; easy to train and apply
  • Nearest neighbor: hard to beat if there is tons of data (e.g., character recognition)
  • Boosted stumps or decision trees: applies to flexible features, incorporates feature selection, powerful classifiers
  • Random forests: outputs probability; good for simple features, tons of data
  • Deep networks / CNNs: flexible output; learns features; adapt existing network (which is trained with tons of data) or train new with tons of data

• Always try at least two types of classifiers
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.
Things to remember

• No free lunch: machine learning algorithms are tools

• Try simple classifiers first

• Better to have smart features and simple classifiers than simple features and smart classifiers
  • Though with enough data, smart features can be learned

• Use increasingly powerful classifiers with more training data (bias-variance tradeoff)
Some Machine Learning References

• General
  • Christopher Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995

• Adaboost

• SVMs

• Random forests