Kernel method, clustering and summary

• 2018/04/24

➢ Announcement:

☐ Slides for this lecture are here


☐ Review session for the final (Thursday)

☐ Final on May 3 4:50PM – 6:40PM

   Study guide is on Blackboard

☐ Participation for end-of-semester survey (current response rate 69%):

   https://fultonapps.asu.edu/eval

   X response rate -> 1.0 * X for everyone

Slides are largely based on information from http://ai.berkeley.edu and Russel
Last time

• Perceptron
  • Concept
  • Error-based learning
  • Issues with Perceptron

• Required reading (red means it will be on your exams):
  o R&N: Chapter 18.6.3
Classification: comparison

Naïve Bayes
- Builds a model training data
- Gives prediction probabilities
- Strong independence assumptions about feature
- One pass through data (counting)
- Nonlinear classifier (in general)

Perceptrons / MIRA:
- Makes less assumptions about data
- Error-driven learning
- Multiple passes through data (prediction)
- Linear classifier (will relax today)

NB can be a linear classifier (e.g., when it uses categorical data)
-- binary Naïve Bayes with Boolean variables are linear (try to prove it!)
Outline for today

• Perceptron
  • Concept
  • Error-based learning
  • Issues with Perceptron
  • Nonlinearity
• Data based classification
• Clustering
  • K-means

• Required reading (red means it will be on your exams):
  o R&N: Chapter 18.8
Non-linear separator

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:

\[ \Phi: x \rightarrow \varphi(x) \]
Perceptron weights

• What is the final value of a weight $w_y$ of a perceptron?
  – Can it be any real vector?
  – No! It’s built by adding up inputs.

\[ w_y = 0 + f(x_1) - f(x_5) + \ldots \]

\[ w_y = \sum_i \alpha_{i,y} f(x_i) \]

• Can reconstruct weight vectors (the **primal representation**) from update counts (the **dual representation**)

\[ \alpha_y = \langle \alpha_{1,y} \alpha_{2,y} \ldots \alpha_{n,y} \rangle \]
Dual Perceptron

• How to classify a new example $x$?

\[
\text{score}(y, x) = w_y \cdot f(x) \\
= \left( \sum_i \alpha_{i,y} f(x_i) \right) \cdot f(x) \\
= \sum_i \alpha_{i,y} (f(x_i) \cdot f(x)) \\
= \sum_i \alpha_{i,y} K(x_i, x)
\]

• If someone tells us the value of $K$ for each pair of examples, never need to build the weight vectors (or the feature vectors)!

What is the size of $\alpha$?
• Start with zero counts (alpha)
• Pick up training instances one by one
• Try to classify $x_n$

$$y = \arg \max_y \sum_i \alpha_{i,y} K(x_i, x_n)$$

• If correct, no change!
• If wrong: lower count of wrong class (for this instance), raise count of right class (for this instance)

$$\alpha_{n,y} = \alpha_{n,y} - 1$$
$$\alpha_{n,y^*} = \alpha_{n,y^*} + 1$$

$$w_y = w_y - f(x_n)$$
$$w_{y^*} = w_{y^*} + f(x_n)$$

$$w_y = \sum_i \alpha_{i,y} f(x_i) = w_y - f(x_n)$$
Kernelized perceptron

• If we had a black box (kernel) $K$ that told us the dot product of the features (we do NOT need to know what these features are, or even how many of them there are!) of two examples $x$ and $x'$:
  – Could work entirely with the dual representation
  – No need to ever take dot products ("kernel trick")

$$score(y, x) = w_y \cdot f(x)$$

$$= \sum_i \alpha_{i,y} K(x_i, x)$$

• Downside: if we have too many examples...
Kernels

So far: a very strange way of doing a very simple calculation

“Kernel trick”: we can substitute any* kernel function in place of the dot product

\[
\text{score}(y, x) = w_y \cdot f(x) = \sum_i \alpha_{i,y} K(x_i, x)
\]

* Fine print: if your kernel doesn’t satisfy certain technical requirements, lots of proofs break. E.g. convergence, mistake bounds. In practice, illegal kernels sometimes work (but not always).
Feature transformation

- Data that is linearly separable works out great for linear decision rules:
  \[ f(x) = [x] \]

- But what are we going to do if the dataset is just too hard?
  \[ f(x) = [x] \]

- How about... mapping data to a higher-dimensional space:
  \[ K(x, x') = f'(x)f'(x') \]
  \[ f'(x) = [x, x^2] \]
  \[ = [x, x^2][x', x'^2] \]
  \[ = xx' + x^2x'^2 \]

This and next few slides adapted from Ray Mooney, UT
Non-linear separator

• General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:

\[ \Phi: x \rightarrow \varphi(x) \]
Some kernels

- Kernels **implicitly** map original vectors to higher dimensional spaces, take the dot product there, and hand the result back.

- **Linear kernel:** \( K(x, x') = x' \cdot x' = \sum_i x_i x'_i \)

- **Quadratic kernel:** \( K(x, x') = (x \cdot x' + 1)^2 \)

  \[
  = \sum_{i,j} x_i x_j x'_i x'_j + 2 \sum_i x_i x'_i + 1
  \]

- **RBF (radial basis function):** infinite dimensional representation

  \( K(x, x') = \exp(-\|x - x'\|^2) \)
Why kernels?

• Can’t you just add these features on your own (e.g. add all pairs of features instead of using the quadratic kernel)?
  – Yes, in principle, just compute them
  \[ K(x, x') = (x \cdot x' + 1)^2 \]
  – No need to modify any algorithms
  – But, number of features can get large (or infinite)
  – Some kernels not as easily thought of in their expanded representation, e.g. RBF kernels

• Kernels let us compute with these features implicitly
  – Example: implicit dot product in quadratic kernel takes much less space and time per dot product
  – Of course, there’s the cost for using the pure dual algorithms: you need to compute the kernel function for every pair of samples
Case based learning
Case based learning

• Classification from similarity
  – Case-based reasoning
  – Predict an instance’s label using similar instances

• Nearest-neighbor classification
  – 1-NN: copy the label of the most similar data point
  – K-NN: vote the k nearest neighbors (need a weighting scheme)
  ➢ **Key issue: how to define similarity**
  – Trade-offs: Small k gives relevant neighbors, Large k gives smoother functions

http://www.cs.cmu.edu/~zhuxj/courseproject/knndemo/KNN.html
Nearest neighbor classifier

- Nearest neighbor for digits:
  - Take new image
  - Compare to all training images
  - Assign based on closest example

- Encoding: image is vector of intensities:
  $1 = \langle 0.0, 0.0, 0.3, 0.8, 0.7, 0.1, \ldots, 0.0 \rangle$

- What’s the similarity function?
  - Dot product of two images vectors?
    $\text{sim}(x, x') = x \cdot x' = \sum_i x_i x'_i$
  - Usually normalize vectors so $||x|| = 1$
  - $\min = 0$ (when?), $\max = 1$ (when?)
Similarity functions
Basic similarities

Many similarities based on feature dot products:

\[
\text{sim}(x, x') = f(x) \cdot f(x') = \sum_i f_i(x) f_i(x')
\]

If features are just the pixels:

\[
\text{sim}(x, x') = x \cdot x' = \sum_i x_i x'_i
\]

Note: not all similarities are of this form
Invariant metrics

Better similarity functions use knowledge about vision
Example: invariant metrics:

Similarities are invariant under certain transformations
Rotation, scaling, translation, stroke-thickness...
E.g:  

16 x 16 = 256 pixels; a point in 256-dim space
These points have small similarity in $\mathbb{R}^{256}$ (why?)

How can we incorporate such invariances into our similarities?

This and next few slides adapted from Xiao Hu, UIUC
Invariant metrics

- Each example is now a curve in $R^{256}$
- Rotation invariant similarity:
  
  $s' = \max s( r(3), r(3) )$

- E.g. highest similarity between images’ rotation lines
Template deformation

Deformable templates:
- An “ideal” version of each category
- Best-fit to image using min variance
- Cost for high distortion of template
- Cost for image points being far from distorted template

Used in many commercial digit recognizers

Examples from [Hastie 94]
Parametric/non-parametric model

- Parametric models:
  - Fixed set of parameters
  - More data means better settings
- Non-parametric models:
  - Complexity of the classifier increases with data
  - Better in the limit, often worse in the non-limit
- (K)NN is non-parametric
K-NN vs Perceptron

• Nearest neighbor-like approaches
  – Use similarity functions
    • kernel function is a special type of similarity function
  – Data-based

• Perceptron-like approaches
  – Error-based
Review: classification

• Classification systems:
  – Supervised learning
  – Make a prediction given evidence
  – We’ve seen several methods for this
  – Useful when you have labeled data
Clustering

• Clustering systems:
  – Unsupervised learning
  – Detect patterns in unlabeled data
    • E.g. group emails or search results
    • E.g. find categories of customers
    • E.g. detect anomalous program executions
  – Useful when don’t know what you’re looking for
  – Requires data, but no labels
  – Often get gibberish
Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns

- What could “similar” mean?
  - One option: small (squared) Euclidean distance

$$\text{dist}(x, y) = (x - y)^T (x - y) = \sum_i (x_i - y_i)^2$$
K-means
K-means

• An iterative clustering algorithm
  – Pick K random points as cluster centers (means)
  – Alternate:
    • Assign data instances to closest mean
    • Assign each mean to the average of its assigned points
  – Stop when no points’ assignments change
K-means in action
K-means as optimization

Consider the total distance to the means:

$$\phi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$

Each iteration reduces \( \phi \)

Two stages each iteration:
- Update assignments: fix means \( c \), change assignments \( a \)
- Update means: fix assignments \( a \), change means \( c \)
Phase 1: update assignment

- For each point, re-assign to closest mean:

$$a_i = \arg\min_k \text{dist}(x_i, c_k)$$

- Can only decrease total distance phi!

$$\phi(\{x_i\}, \{a_i\}, \{c_k\}) = \sum_i \text{dist}(x_i, c_{a_i})$$
Phase 2: update means

- Move each mean to the average of its assigned points:

$$c_k = \frac{1}{|\{i : a_i = k\}|} \sum_{i : a_i = k} x_i$$

- Also can only decrease total distance... (Why?)

- Fun fact: the point $y$ with minimum squared Euclidean distance to a set of points $\{x\}$ is their mean
Initialization

- K-means is non-deterministic
  - Requires initial means
  - It does matter what you pick!
  - What can go wrong?
**K means get stuck**

A local optimum:

Why doesn’t this work out like the earlier example, with the purple taking over half the blue?

Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics
K-means questions

- Will K-means converge?
  - To a global optimum?

- Will it always find the true patterns in the data?

- How many clusters to pick?
Example: image clustering

<table>
<thead>
<tr>
<th>Category</th>
<th>Exemplary Photos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friends (users posing with others friends; At least two human faces are in the photo)</td>
<td><img src="image1" alt="Exemplary Photos" /></td>
</tr>
<tr>
<td>Food (food, recipes, cakes, drinks, etc.)</td>
<td><img src="image2" alt="Exemplary Photos" /></td>
</tr>
<tr>
<td>Gadget (electronic goods, tools, motorbikes, cars, etc.)</td>
<td><img src="image3" alt="Exemplary Photos" /></td>
</tr>
<tr>
<td>Captioned Photo (pictures with embed text, memes, and so on)</td>
<td><img src="image4" alt="Exemplary Photos" /></td>
</tr>
<tr>
<td>Pet (animals like cats and dogs which are the main objects in the picture)</td>
<td><img src="image5" alt="Exemplary Photos" /></td>
</tr>
<tr>
<td>Activity (both outdoor &amp; indoor activities, places where activities happen, e.g., concert, landmarks)</td>
<td><img src="image6" alt="Exemplary Photos" /></td>
</tr>
<tr>
<td>Selfie (self-portraits; only one human face is present in the photo)</td>
<td><img src="image7" alt="Exemplary Photos" /></td>
</tr>
<tr>
<td>Fashion (shoes, costumes, makeup, personal belongings, etc.)</td>
<td><img src="image8" alt="Exemplary Photos" /></td>
</tr>
</tbody>
</table>
That is it!
Summary of topics covered

• Part 1: Making Decisions
  – Search
  – Games and adversarial search

• Part 2: Decision under uncertainty
  – MDP
  – Reinforcement learning

• Part 3: Probabilistic reasoning
  – Markov Model & HMM
  – Bayesian network

• Part 4: Machine learning
  – Classification
  – Clustering
Object detection

- Key challenges:
  - Many other objects may be there
Autonomous helicopter flight

- Key challenges:
  - Track helicopter position and orientation during flight
  - Decide on control inputs to send to helicopter
Machine translation

- Translate text from one language to another
- Recombines fragments of example translations
- Challenges:
  - What fragments? [learning to translate]
Starcraft
Autonomous cars

- 5 Lasers
- Camera
- Radar
- E-stop
- GPS compass
- 6 Computers
- IMU
- Control Screen
- Steering motor
Apprenticeship via classification
Outline for today

• Perceptron
  • Concept
  • Error-based learning
  • Issues with Perceptron
  • Nonlinearity
• Data based classification
• Clustering
  • K-means

• Required reading (red means it will be on your exams):
  o R&N: Chapter 18.8