Instance-Based Learning
(aka non-parametric methods)

CSE 446 Machine Learning Daniel Weld
March 7, 2012

with slides from Carlos Guestrin & Bishop

Machine Learning
Supervised Learning
Unsupervised Learning
Supervised Learning
Reinforcement Learning

Parameteric
Non-parametric

Space of ML Problems

Type of Supervision
(eg, Experience, Feedback)

<table>
<thead>
<tr>
<th>What is Being Learned?</th>
<th>Labeled Examples</th>
<th>Reward</th>
<th>Nothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete Function</td>
<td>Classification</td>
<td>Clustering</td>
<td></td>
</tr>
<tr>
<td>Continuous Function</td>
<td>Regression</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Policy</td>
<td>Apprenticeship Learning</td>
<td>Reinforcement Learning</td>
<td></td>
</tr>
</tbody>
</table>

Nonparametric Methods

- Parametric models restricted to specific forms
  - May not be a good fit
- Nonparametric methods make few assumptions
  - Often work extremely well in practice

Todo

- Bishop approach is too theoretical – don’t do it!

Histograms

Histogram methods partition the data space into distinct bins with widths \( \Delta_i \) and count the number of observations, \( n_i \), in each bin.

\[
P_i = \frac{n_i}{\Delta_i}
\]

- Often, the same width is used for all bins, \( \Delta_i = \Delta \).
- \( \Delta \) acts as a smoothing parameter.

In an D-dimensional space, using M bins in each dimension will require \( M^D \) bins!

Why is this bad?
Nonparametric Methods

- Assume observations drawn from a density \( p(x) \) and consider a small region \( R \) containing \( x \) such that

\[
P = \int_R p(x) \, dx.
\]

- The probability that \( K \) out of \( N \) observations lie inside \( R \) is \( \text{Bin}(K \mid N, P) \) and if \( N \) is large

\[
K \approx NP.
\]

If \( V \) (the volume of \( R \)) is sufficiently small, \( p(x) \) is approximately constant over \( R \) and

\[
\frac{P}{V} \approx p(x).
\]

Thus

\[
\frac{P}{V} \approx \frac{K}{NV}.
\]

Nonparametric Methods (3.5)

- Hold \( K \) fixed, determine \( V \) from data
  - K-nearest-neighbour approach

- Hold \( V \) fixed, determine \( K \) from data
  - Kernel-density estimation

Kernel Density Estimation

- Fix \( V \), estimate \( K \) from the data.
- Let \( R \) be a hypercube centred on \( x \)
  - Define the kernel function (Parzen window)

\[
k(x - x_i)/h) = \begin{cases} 1, & \frac{|x_i - x_k|}{h} < 1/2, \\ 0, & \text{otherwise,} \end{cases} \quad i = 1, \ldots, D.
\]

It follows that

\[
K = \sum_{i=1}^{N} k \left( \frac{x - x_i}{h} \right) \quad \text{and hence} \quad p(x) = \frac{1}{h} \sum_{i=1}^{N} \frac{1}{K} k \left( \frac{x - x_i}{h} \right).
\]

Kernel Density Estimation

To avoid discontinuities in \( p(x) \), use a smooth kernel, e.g. a Gaussian

\[
p(x) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2\pi h^D} \exp \left( \frac{-|x - x_i|^2}{2h^2} \right).
\]

Any kernel such that

\[
f(x) \geq 0,
\]

\[
\int f(x) \, dx = 1
\]

will work.

Pros / Cons

- Nonparametric models (besides histograms)
  - Requires storing and computing with the entire data set.

- Parametric models, once fitted,
  - Much more efficient in terms of storage and computation.

Nearest Neighbour Density Estimation

Fix \( K \), estimate \( V \) from data.
Consider a hypersphere centred on \( x \)
Let it grow to a volume, \( V^* \)
that includes \( K \) of \( N \) data points.

Then

\[
p(x) = \frac{K}{NV^*}.
\]
K-Nearest-Neighbours for Classification

- Given a data set with $N_k$ data points from class $C_k$ and $N$, we have $\sum_k N_k = N$.
  and correspondingly.

- Since $\frac{N_k}{N} = \frac{p(C_k)N}{p(C_k)p(x)} = \frac{K_0}{K}$.

K acts as a smoother
- As $N \to \infty$, the error rate of the 1-nearest-neighbour classifier is never more than twice the optimal error (obtained from the true conditional class distributions).

Linear Regression: What can go wrong?

What do we do if the bias is too strong?
- Might want the data to drive the complexity of the model!
- Try instance-based Learning (a.k.a. non-parametric methods)?

Using data to predict new data

Nearest neighbor with Lots of Data
Univariate 1-Nearest Neighbor

Given datapoints \((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\), where we assume \(y_i = f(x_i)\) for some unknown function \(f\). Given query point \(x_q\), your job is to predict \(\hat{y} \approx f(x_q)\).

1. Find the closest \(x_i\) in our set of datapoints
   \[ \hat{y} = y_i \]

2. Predict \(\hat{y} = y_i(x_q)\)

Here’s a dataset with one input, one output and four datapoints.

1-Nearest Neighbor

To define an instance-based learner, specify four things:

1. A distance metric
   - Euclidean (and many more)
2. How many nearby neighbors to look at?
   - One
3. A weighting function (optional)
   - Unused
4. How to fit with the local points?
   - Just predict the same output as the nearest neighbor.

Notable distance metrics (and their level sets)

- Scaled Euclidean (L2)
- Mahalanobis (here, \(\Sigma\) on the previous slide is not necessarily diagonal, but is symmetric)
- \(L_1\) norm (absolute)
- \(L_1\) (max) norm

1-Nearest Neighbor is an example of...

Instance-based learning

A function approximator that has been around since about 1910.

To make a prediction, search database for similar datapoints, and fit with the local points.

To define an instance-based learner, specify four things:

- A distance metric
- How many nearby neighbors to look at?
- A weighting function (optional)
- How to fit with the local points?

Multivariate 1-NN examples

Classification

Regression

Consistency of 1-NN

- Consider an estimator \(f_n\), trained on \(n\) examples (e.g., 1-NN, neural nets, regression, ...).
- Estimator is consistent if true error goes to zero as amount of data increases (e.g., for noise-free data, consistent if for any data distribution \(p(x)\): \(\lim_{n \to \infty} MSE(f_n) = 0\)

\[ MSE(f_n) = \int p(x) (f_n(x) - y)^2 \, dx \]

- Linear regression is not consistent!
  - Representation bias
- 1-NN is consistent
  - What about noisy data?
  - What about variance?
1-NN overfits?

**k-Nearest Neighbor**

Instance-based learning, four things to specify:

1. **A distance metric**
   - Euclidian (and many more)

2. **How many nearby neighbors to look at?**
   - \( k \)

1. **A weighting function (optional)**
   - Unused

2. **How to fit with the local points?**
   - Return the average output
   - \( \text{predict} = \frac{1}{k} \sum y_i \) (summing over \( k \) nearest neighbors)

---

Which is better? What can we do about the discontinuities?

**Weighted distance metrics**

Suppose the input vectors \( x_1, x_2, \ldots, x_n \) are two dimensional:

\[
(x_1, y_1) = (x_1^1, x_1^2), (x_2, y_2) = (x_2^1, x_2^2), \ldots, (x_n, y_n) = (x_n^1, x_n^2)
\]

Nearest-neighbor regions in input space:

\[
\text{Dist}(x, y) = \sqrt{(x^1 - y^1)^2 + (x^2 - y^2)^2}
\]

The relative scaling of the distance metric affect region shapes

---

**Weighted Euclidean distance metric**

\[
D(x_i, x_j) = \sqrt{\sum \left( \frac{x_{ij} - x_{ij}'}{\sigma_{jj}} \right)^2}
\]

Or equivalently,

\[
D(x_i, x_j) = \sqrt{\sum \left( x_{ij} - x_{ij}' \right)^2}
\]

where

\[
\Sigma = \begin{bmatrix}
\sigma_1^2 & 0 & \cdots & 0 \\
0 & \sigma_2^2 & \cdots & 0 \\
\vdots & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & \sigma_N^2
\end{bmatrix}
\]

Other Metrics...
- Mahalanobis, Rank-based, Correlation-based, …

---

**Kernel Regression**

Instance-based learning:

1. **A distance metric**
   - Euclidian (and many more)

2. **How many nearby neighbors to look at?**
   - All of them

3. **A weighting function**
   - \( w_i = \exp(-D(x_i, \text{query})^2 / K_w^2) \)
     - Nearby points to the query are weighted strongly.
     - Far points weighted weakly.
     - The \( K_w \) parameter is the Kernel Width. Very important.

4. **How to fit with the local points?**
   - Predict the weighted average of the outputs:
   - \( \text{predict} = \Sigma w_i y_i / \Sigma w_i \)
Many possible weighting functions

\[ w_i = \exp(-D(x_i, \text{query})^2 / K_w^2) \]

Typically:
- Choose \( D \) manually
- Optimize \( K_w \) using gradient descent

(Our examples use Gaussian)

Kernel regression predictions

Increasing the kernel width \( K_w \) means further away points get an opportunity to influence you. As \( K_w \to \infty \), the prediction tends to the global average.

Kernel regression on our test cases

Choosing a good \( K_w \) is important! Remind you of anything we have seen?

Kernel regression: problem solved?

Where are we having problems?
- Sometimes in the middle…
- Generally, on the ends (extrapolation is hard!)

Time to try something more powerful…!!!

Locally weighted regression

Kernel regression:
- Take a very very conservative function approximator called AVERAGING.
- Locally weight it.

Locally weighted regression:
- Take a conservative function approximator called LINEAR REGRESSION.
- Locally weight it.

Locally weighted regression

Instance-based learning, four things to specify:
- A distance metric
  Any
- How many nearby neighbors to look at?
  All of them
- A weighting function (optional)
  Kernels: \( w_i = \exp(-D(x_i, \text{query})^2 / K_w^2) \)
- How to fit with the local points?

General weighted regression:

\[ \hat{\beta} = \arg\min_{\beta} \sum_{i \in D} w_i [y_i - \beta x_i]^2 \]
How LWR works

**Linear regression**
- Same parameters for all queries
  \[ \hat{\beta} = (X'X)^{-1}X'Y \]

**Locally weighted regression**
- Solve weighted linear regression for each query
  \[ \phi = \left( XW X' \right)^{-1}XW Y \]

LWR on our test cases

LWR on our test cases

Locally weighted polynomial regression

Kernel Regression: Kernel width KW at optimal level.

- KW = 1/100 x-axis
- KW = 1/40 x-axis
- KW = 1/15 x-axis

Local quadratic regression is easy: just add quadratic terms to the WXTWX matrix. As the regression degree increases, the kernel width can increase without introducing bias.

Challenges for based learning

- Must store and retrieve all data!
- Most real work done during testing
- For every test sample, must search through all dataset – very slow!
- But, there are fast methods for dealing with large datasets
- Instance-based learning often poor with noisy or irrelevant features
  - In high dimensional spaces, all points will be very far from each other
  - Can need a number of examples that is exponential in the dimension of X
  - But, sometimes you are ok if you are clever about features

Curse of the irrelevant feature

This is a contrived example, but similar problems are common in practice.

Need some form of feature selection!!

Curse of Dimensionality

This is a contrived example, but similar problems are common in practice.

Need some form of feature selection!!
Curse of Dimensionality

Fraction of volume of sphere lying in the range $r = [1, 1-\varepsilon]$

Gaussian Densities in higher dimensions

Side-Stepping the Curse

- Dimensionality reduction
  - Eg, PCA
- Then use NN

In Summary: Instance-Based Learning

- k-NN
  - Simplest learning algorithm
  - With sufficient data, very hard to beat "strawman" approach
  - Picking $k$?
- Kernel regression
  - Set $k$ to $n$ (number of data points) and optimize weights by gradient descent
  - Smoother than k-NN
- Locally weighted regression
  - Generalizes kernel regression, not just local average
- Curse of dimensionality
  - Must remember (very large) dataset for prediction
  - Irrelevant features often killers for instance-based approaches

Acknowledgment

- This lecture contains some material from Andrew Moore’s excellent collection of ML tutorials:
  - http://www.cs.cmu.edu/~awm/tutorials

©Carlos Guestrin 2005-