Instance Based Methods I

CPS 271
Ron Parr

With content adapted from Lise Getoor (& Tom Dietterich, Ray Mooney, Andrew Moore)

Parametric Methods

- Supervised learning
  - Linear classifiers
  - Non-linear classifiers, e.g., neural networks
- These methods are parametric
- Alternative: Remember stuff
- AKA: Case based or memory based

Overview

- Classification
  - Nearest neighbor
  - K-NN
- Regression

Example

- Flood risk
- Data set:
  - GPS coordinates (features)
  - Flood data for previous hundred years
- Task: predict flood risk for new data points

Nearest Neighbor Algorithm

- Learning Algorithm:
  - Store training examples
  - “But that’s not learning…”
- Prediction Algorithm:
  - To classify a new example \( x \) by finding the training example \( (x', t') \) that is nearest to \( x \)
  - Guess the class \( t = t' \)
  - Learning implicit in query mechanism

Decision Boundaries

- The nearest neighbor algorithm does not explicitly compute decision boundaries. However, the decision boundaries form a subset of the Voronoi diagram for the training data.

- Each line segment is equidistant between two points of opposite classes. The more examples that are stored, the more complex the decision boundaries can become.
Issues

- Noise
- Efficiency
  - Use K-D Trees
  - Still bad for high dimensions
- Distance measure (critical!)
  - How to pick
  - Questionably useful in high dimensions
- Irrelevant features

Dealing with Noise

- Consider k nearest neighbors (K-NN)
- Neighbors vote

Picking Distance Measures

- No silver bullet
- Many rules of thumb
- Problem knowledge always helps

Distance: Preprocessing

- What if features don’t have same range?
- Normalize feature values
  - Scale to same range
  - Usually $-1,+1$ scale

Distance Measures

- Two methods for computing similarity:
  1. Explicit similarity measurement for each pair of objects
  2. Similarity obtained indirectly based on vector of object attributes.
- Metric: $d(i,j)$ is a metric iff
  1. $d(i,j) \geq 0$ for all $i, j$ and $d(i,j) = 0$ iff $i = j$
  2. $d(i,j) = d(j,i)$ for all $i$ and $j$
  3. $d(i,j) \leq d(i,k) + d(k,j)$ for all $i, j$ and $k$
- Most common distance metric is Euclidean distance:
  $$d_k(x^{(i)}, x^{(j)}) = \left( \sum_{k=1}^{p} (x_k^{(i)} - x_k^{(j)})^2 \right)^{1/2}$$
- Makes sense in the case where the different measurements are commensurate; each is variable measured in the same units. If the measurements are different, say length and weight, it is not clear.
Standardization

When variables are not commensurate, we can standardize them by dividing by the sample standard deviation. This makes them all equally important.

The estimate for the standard deviation of $x_k$:

$$\hat{\sigma}_i = \left(\frac{1}{n} \sum_{i=1}^{n} (x_{k}^{(i)} - \bar{x}_k)^2\right)^{\frac{1}{2}}$$

where $\bar{x}_k$ is the sample mean:

$$\bar{x}_k = \frac{1}{n} \sum_{i=1}^{n} x_{k}^{(i)}$$

But what about correlation???

Mahalanobis distance

$$d_{Mah}(x^{(i)}, x^{(j)}) = \left(\left(x^{(i)} - x^{(j)}\right)^{T} \Sigma^{-1} \left(x^{(i)} - x^{(j)}\right)\right)^{\frac{1}{2}}$$

Inverse covariance matrix

(compare with Gaussian)

1. It automatically accounts for the scaling of the coordinate axes
2. It corrects for correlation between the different features

Price:

1. The covariance matrices can be hard to determine accurately
2. The memory and time requirements grow quadratically rather than linearly with the number of features.

Other Distance Metrics

- Minkowski or $L_p$ metric:
  $$d_p(x^{(i)}, x^{(j)}) = \left(\sum_{k=1}^{p} (x_{k}^{(i)} - x_{k}^{(j)})^p\right)^{\frac{1}{p}}$$

- Manhattan, city block or $L_1$ metric:
  $$d_1(x^{(i)}, x^{(j)}) = \sum_{k=1}^{p} |x_{k}^{(i)} - x_{k}^{(j)}|$$

- $L_\infty$
  $$d_\infty(x^{(i)}, x^{(j)}) = \max_{k=1}^{p} |x_{k}^{(i)} - x_{k}^{(j)}|$$

Nearest Neighbor Summary

- Advantages
  - Variable-sized hypothesis space
  - Learning is extremely efficient (low $d$)
  - Very flexible decision boundaries

- Disadvantages
  - Distance function must be carefully chosen
  - Irrelevant or correlated features must be eliminated
  - Typically cannot handle more than 30 features
  - Memory costs
  - Expensive queries

Non-Parametric Regression Methods

- Carry over our intuitions from classification
- Look at some set of “neighbors”
- Some issues assumed away (distance metric) [Is this valid?]

Why not just use Linear Regression?

Here, linear regression manages to capture a significant trend in the data, but there is visual evidence of bias.

Here, linear regression appears to have a much better fit, but the bias is very clear.

Here, linear regression may indeed be the right thing.

Bias: the underlying choice of model (in this case, a line) cannot, with any choice of parameters (constant term and slope) and with any amount of data (the dots) capture the full relationship.

Copyright © 2001, Andrew W. Moore
Why not just Join the Dots?

Here, joining the dots is clearly fitting noise.
Here, joining the dots looks very sensible.
Again, a clear case of noise fitting.

Why is fitting the noise so bad?

One-Nearest Neighbor

Similar to Join The Dots with Pro and Con.

• PRO: It is easy to implement with multivariate inputs.
• CON: It no longer interpolates locally

k-Nearest Neighbor (here k=9)

K-nearest neighbor for function fitting smooths away noise, but there are clear deficiencies.

What can we do about all the discontinuities that k-NN gives us?

Kernel Regression

(Not Dual/GP Regression)

Four things make a memory based learner:

1. A distance metric
   Scaled Euclidean
2. How many nearby neighbors to look at?
   All of them
3. A weighting function (optional)
   \( w_i = \exp(-D(x_i, \text{query})^2 / K^2) \)
   Nearby points to the query are weighted strongly, far points weakly. The \( K \) parameter is the Kernel Width. Very important.
4. How to fit with the local points?
   Predict the weighted average of the outputs:
   \[ \text{predict} = \frac{\sum w_i y_i}{\sum w_i} \]

Kernel Regression Predictions

Increasing the kernel width \( K \) means further away points get an opportunity to influence you.
As \( K \to \infty \), the prediction tends to the global average.
**Kernel Regression on our test cases**

Choosing a good $K_w$ is important. Not just for Kernel Regression, but for all the locally weighted learners.

Copyright © 2001, Andrew W. Moore

---

**Kernel Regression can look bad**

Copyright © 2001, Andrew W. Moore

---

**Locally Weighted Regression**

- Like kernel regression LWR uses “neighbors”
- Kernel regression simply averages neighbors
- LWR finds a **locally linear model**

---

**Locally Weighted Regression**

Four things make a memory-based learner:
1. A distance metric
   - Scaled Euclidian
2. How many nearby neighbors to look at?
   - All of them
3. A weighting function (optional)
   - $w(x) = \exp(-D(x^i, x^j)^2 / K_w^2)$
   - Nearby points to the query are weighted strongly, far points weakly. The $K_w$ parameter is the kernel width.
4. How to fit with the local points?
   - First form a local linear model. Find the $\theta$ that minimizes the locally weighted sum of squared residuals:
   - $\theta = \arg\min_\theta \sum_{i} w(x^i) (y^i - \theta^T x^i)^2$
   - Then predict $\hat{y}_{\text{predict}} = \theta^T x^\text{query}$

Note: Here $w$ are neighbor weights $\theta$ are regression weights.

Copyright © 2001, Andrew W. Moore (modified by RP)

---

**Locally weighted Polynomial regression**

Kernel Regression
- Kernel width $K_w$ at optimal level.
- $K_w = 1/100$ x-axis

LW Linear Regression
- Kernel width $K_w$ at optimal level.
- $K_w = 1/40$ x-axis

LW Quadratic Regression
- Kernel width $K_w$ at optimal level.
- $K_w = 1/15$ x-axis

RP: We defer discussion of cost/benefit of extra terms

Copyright © 2001, Andrew W. Moore
When’s Quadratic better than Linear?

- It can let you use a wider kernel without introducing bias,
- but in higher dimensions is expensive, needs more data.
- Two “Part-way-between-linear-and-quadratic” polynomials:
  - "Ellipses": Add $x_i^2$ terms to the model, but not cross-terms (no $x_ix_j$ where $i=j$)
  - "Circles": Add only one extra term to the model:

\[ x_{D+1} = \sum_{i=1}^{D} x_i^2 \]

Locally Weighted Learning: Variants

- Range Searching: Average all neighbors w/in given range
- Range-based linear regression
- Linear Regression on K-nearest-neighbors
- Weighting functions that decay to 0 at the kth nn
- Locally weighted Iteratively Reweighted Least Squares
- Locally weighted Logistic Regression
- Locally weighted classifiers

- Multilinear Interpolation
- Kuhn-Triangulation-based Interpolation
- Spline Smoothers

Non-Parametric Methods: Conclusions

- Very expressive method for
  - Classification
  - Regression
- Perhaps too powerful
- Can be memory/compute intensive for queries
- Heavy dependence upon distance/kernel
- Good method to use when:
  - Data fills feature space well
  - Good intuitions about distance