Classification Algorithms – Regression & Knn

Outline

- **Linear Models (Regression)**
- Instance-based (Nearest-neighbor)

Linear models

- Work most naturally with numeric attributes
- Standard technique for numeric prediction: linear regression
  - Outcome is linear combination of attributes

$$x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$$

- Weights are calculated from the training data
- Predicted value for first training instance $\mathbf{a}^{(1)}$

$$w_0 a_0^{(1)} + w_1 a_1^{(1)} + w_2 a_2^{(1)} + \ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)}$$

Minimizing the squared error

- Choose $k+1$ coefficients to minimize the squared error on the training data
- Squared error: $$\sum_{i=1}^{n}\left(x^{(i)}-\sum_{j=0}^{k} w_j a_j^{(i)}\right)^2$$
- Derive coefficients using standard matrix operations
- Can be done if there are more instances than attributes (roughly speaking)
- Minimizing the *absolute error* is more difficult

Regression for Classification

- *Any* regression technique can be used for classification
  - Training: perform a regression for each class, setting the output to 1 for training instances that belong to class, and 0 for those that don't
  - Prediction: predict class corresponding to model with largest output value (*membership value*)
- For linear regression this is known as *multi-response linear regression*

*Theoretical justification

Observed target value (either 0 or 1)

Model ***Instance*** ***The scheme minimizes this***

$$E_y\{(f(X)-Y)^2 \mid X=x\}$$

True class probability

$$= E_y\{(f(X)-P(Y=1 \mid X=x)+P(Y=1 \mid X=x)-Y)^2 \mid X=x\}$$

$$= (f(x)-P(Y=1 \mid X=x))^2 + 2\times(f(x)-P(Y=1 \mid X=x))\times$$
$$E_y\{P(Y=1 \mid X=x)-Y \mid X=x\} + E_y\{(P(Y=1 \mid X=x)-Y)^2 \mid X=x\}$$

$$= (f(x)-P(Y=1 \mid X=x))^2 + 2\times(f(x)-P(Y=1 \mid X=x))\times$$
$$(P(Y=1 \mid X=x)-E_y\{Y \mid X=x\}) + E_y\{(P(Y=1 \mid X=x)-Y)^2 \mid X=x\}$$

$$= \underbrace{(f(x)-P(Y=1 \mid X=x))^2}_{\textbf{\textit{We want to minimize this}}} + \underbrace{E_y\{(P(Y=1 \mid X=x)-Y)^2 \mid X=x\}}_{\textbf{\textit{Constant}}}$$

*Pairwise regression

- Another way of using regression for classification:
  - A regression function for every *pair* of classes, using only instances from these two classes
  - Assign output of +1 to one member of the pair, –1 to the other
- Prediction is done by voting
  - Class that receives most votes is predicted
  - Alternative: "don't know" if there is no agreement
- More likely to be accurate but more expensive

Logistic regression

- Problem: some assumptions violated when linear regression is applied to classification problems
- *Logistic* regression: alternative to linear regression
  - Designed for classification problems
  - Tries to estimate class probabilities directly
    - Does this using the *maximum likelihood* method
  - Uses this linear model:

$$\log\left(\frac{P}{1-P}\right) = w_0 a_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$$

P= Class probability

Discussion of linear models

- Not appropriate if data exhibits non-linear dependencies
- But: can serve as building blocks for more complex schemes (i.e. model trees)
- Example: multi-response linear regression defines a *hyperplane* for any two given classes:

$$(w_0^{(1)} - w_0^{(2)})a_0 + (w_1^{(1)} - w_1^{(2)})a_1 + (w_2^{(1)} - w_2^{(2)})a_2 + \ldots + (w_k^{(1)} - w_k^{(2)})a_k > 0$$

Comments on basic methods

- Minsky and Papert (1969) showed that linear classifiers have limitations, e.g. can’t learn XOR
  - But: combinations of them can ($\rightarrow$ Neural Nets)

Outline

- Linear Models (Regression)
- **Instance-based (Nearest-neighbor)**

Instance-based representation

- Simplest form of learning: *vote learning*
  - Training instances are searched for instance that most closely resembles new instance
  - The instances themselves represent the knowledge
  - Also called ***instance-based*** learning
- Similarity function defines what’s “learned”
- Instance-based learning is ***lazy*** learning
- Methods:
  - *nearest-neighbor*
  - *k-nearest-neighbor*
  - …

The distance function

- Simplest case: one numeric attribute
  - Distance is the difference between the two attribute values involved (or a function thereof)
- Several numeric attributes: normally, Euclidean distance is used and attributes are normalized
- Nominal attributes: distance is set to 1 if values are different, 0 if they are equal
- Are all attributes equally important?
  - Weighting the attributes might be necessary

Instance-based learning

- Distance function defines what's learned
- Most instance-based schemes use *Euclidean distance*:

  $$\sqrt{(a_1^{(1)} - a_1^{(2)})^2 + (a_2^{(1)} - a_2^{(2)})^2 + \ldots + (a_k^{(1)} - a_k^{(2)})^2}$$

  a$^{(1)}$ and **a**$^{(2)}$: two instances with *k* attributes
- Taking the square root is not required when comparing distances
- Other popular metric: *city-block (Manhattan) metric*
  - Adds differences without squaring them

Normalization and other issues

- Different attributes are measured on different scales $\Rightarrow$ need to be *normalized*:

$$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i} \qquad \text{or} \qquad a_i = \frac{v_i - Avg(v_i)}{StDev(v_i)}$$

v_i: the actual value of attribute i

- Nominal attributes: distance either 0 or 1
- Common policy for missing values: assumed to be maximally distant (given normalized attributes)

Discussion of 1-NN

- Often very accurate
- … but slow:
  - simple version scans entire training data to derive a prediction
- Assumes all attributes are equally important
  - Remedy: attribute selection or weights
- Possible remedies against noisy instances:
  - Take a majority vote over the *k* nearest neighbors
  - Removing noisy instances from dataset (difficult!)
- Statisticians have used *k*-NN since early 1950s
  - If $n \to \infty$ and $k/n \to 0$, error approaches minimum

Summary

- Simple methods frequently work well
  - robust against noise, errors
- Advanced methods, if properly used, can improve on simple methods
- No method is universally best