Data Mining

CS57300
Purdue University

Bruno Ribeiro

January 22, 2018
Differences Between Classification & Prediction

- Classification
  - Observing feature x predict discrete-valued label y

- Prediction (point estimate)
  - Observing feature x predict of real-value label y
  - Forecasting: predictions + confidence intervals

- Ranking prediction (rank estimate)
  - Predict item ranking
    - E.g.: Google results, Netflix recommendations
Predictive modeling

• Data representation:
  • Training set: Paired attribute vectors and labels $<y(i), x(i)>$
  or
  $n \times p$ tabular data with label ($y$) and $p-1$ attributes ($x$)

• Task: estimate a predictive function $f(x; \theta) = y$
  • Assume that there is a function $y = f(x)$ that maps data instances ($x$) to labels ($y$)
  • Construct a model that approximates the mapping
    • Classification: if $y$ is categorical
    • Regression: if $y$ is real-valued
Modeling approaches
Learning predictive models

- Choose a **data representation**

- Select a **knowledge representation** (a “model”)
  - Defines a **space** of possible models $M=\{M_1, M_2, ..., M_k\}$

- Use **search** to identify “best” model(s)
  - Search the space of models (i.e., with alternative structures and/or parameters)
  - Evaluate possible models with **scoring function** to determine the model which best fits the data
Scoring functions

• Given a model M and dataset D, we would like to “score” model M with respect to D
  • Goal is to rank the models in terms of their utility (for capturing D) and choose the “best” model
  • Score function can be used to search over parameters and/or model structure
  • Score functions can be different for:
    • Models vs. patterns
    • Predictive vs. descriptive functions
    • Models with varying complexity (i.e., number parameters)
Predictive scoring functions

- Assess the quality of predictions for a set of instances
  - Measures **difference** between the prediction $M$ makes for an instance $i$ and the true class label value of $i$

\[
S(M) = \sum_{i=1}^{N_{test}} d[f(x(i); M), y(i)]
\]

- Sum over examples
- Distance between predicted and true
- True class label for item $i$
- Predicted class label for item $i$
Predictive scoring functions

- Common score functions:
  - Zero-one loss
    
    \[
    S_{0/1}(M) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} I[f(x(i); M), y(i)]
    \]

    where \( I(a, b) = \begin{cases} 
    1 & a \neq b \\
    0 & \text{otherwise}
    \end{cases} \)

  - Squared loss
    
    \[
    S_{sq}(M) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} [f(x(i); M) - y(i)]^2
    \]

    Careful with definition of class labels! Why?

- More later…

- Do we minimize or maximize these functions?
Scoring functions

• Guide search *inside* of algorithms
  • Select path in heuristic search
  • Decide when to stop
  • Identify whether to include model or pattern in output

• Evaluate results *outside* of algorithms
  • Measure the absolute quality of model or pattern
  • Compare the relative quality of different algorithms or outputs
Where’s the search?
Find Parameters that Minimize Model Score (Error)

Usually we maximize (- score)

Learned model \( \approx (\theta_0 = 0.8, \theta_1 = 0.4) \)
Searching over models/patterns

• Consider a **space** of possible models $M = \{M_1, M_2, \ldots, M_k\}$ with parameters $\theta$

• Search could be over model structures or parameters, e.g.:
  
  • **Parameters**: In a linear regression model, find the regression coefficients ($\beta$) that minimize squared loss on the training data

  • **Model structure**: In a decision trees, find the tree structure that maximizes accuracy on the training data
Optimization over score functions

• **Smooth** functions:
  - If a function is *smooth*, it is differentiable and the derivatives are continuous, then we can use gradient-based optimization
    - If function is *convex*, we can solve the minimization problem in closed form: $\nabla S(\theta)$ using **convex optimization**
    - If function is smooth but non-linear, we can use iterative search over the surface of $S$ to find a local minimum (e.g., hill-climbing)

• **Non-smooth** functions:
  - If the function is *discrete*, then traditional optimization methods that rely on smoothness are not applicable. Instead we need to use **combinatorial optimization**
Linear Methods for Classification
Motivation 1/2

• Given $x$ features of a car (length, width, mpg, maximum speed, …)
• Classify cars into categories based on $x$
Motivation 2/2

- A person arrives at the emergency room with a set of symptoms that could possibly be attributed to one of three medical conditions. Which of the three conditions does the individual have?

- An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, on the basis of the user’s IP address, past transaction history, and so forth.

- On the basis of DNA sequence data for a number of patients with and without a given disease, a biologist would like to figure out which DNA mutations are deleterious (disease-causing) and which are not.
Linear Discriminant Function (Two Classes)

- Two classes
- \( x \) is a D-dimensional real-valued vector (set of features)
- \( y \) is the car class

\[
y_c = \begin{cases} 
1 & \text{, if car } c \text{ is “small”} \\
-1 & \text{, if car } c \text{ is “luxury”} 
\end{cases}
\]

- Find linear discriminant weights \( w \)

\[
y(x) = w^T x + w_0
\]

- Score function is the squared error

\[
\sum_{c \in \text{TestDataCars}} (y_c - y(x_c))^2
\]

- Search algorithm: least squares algorithm

Figure: C. Bishop
How to Deal with Multiple Classes?
Naïve Approach: one vs. many Classification

• How to classify objects into multiple types?

\[ y_c^{(1)} = \begin{cases} 
1 & \text{, if car } c \text{ is “small”} \\
-1 & \text{, if car } c \text{ is “luxury”}
\end{cases} \]

\[ y_c^{(2)} = \begin{cases} 
1 & \text{, if car } c \text{ is “small”} \\
-1 & \text{, if car } c \text{ is “medium”}
\end{cases} \]

\[ y_c^{(3)} = \begin{cases} 
1 & \text{, if car } c \text{ is “medium”} \\
-1 & \text{, if car } c \text{ is “luxury”}
\end{cases} \]

Might work OK in some scenarios… but not clear in this case
Issue with using binary classifiers for K classes

Figure 4.2

Attempting to construct a K-class discriminant from a set of two class discriminants leads to ambiguous regions, shown in green. On the left is an example involving the use of two discriminants designed to distinguish points in class $C_k$ from points not in class $C_k$. On the right is an example involving three discriminant functions each of which is used to separate a pair of classes $C_k$ and $C_j$.

An alternative is to introduce $K \left( \frac{K-1}{2} \right)$ binary discriminant functions, one for every possible pair of classes. This is known as a one-versus-one classifier. Each point is then classified according to a majority vote amongst the discriminant functions. However, this too runs into the problem of ambiguous regions, as illustrated in the right-hand diagram of Figure 4.2.

We can avoid these difficulties by considering a single K-class discriminant comprising $K$ linear functions of the form

$$y_k(x) = w_k^T x + w_k^0$$

(4.9)

and then assigning a point $x$ to class $C_k$ if $y_k(x) > y_j(x)$ for all $j \neq k$. The decision boundary between class $C_k$ and class $C_j$ is therefore given by

$$y_k(x) = y_j(x)$$

and hence corresponds to a $(D-1)$-dimensional hyperplane defined by

$$(w_k - w_j)^T x + (w_k^0 - w_j^0) = 0$$

(4.10)

This has the same form as the decision boundary for the two-class case discussed in Section 4.1.1, and so analogous geometrical properties apply.

The decision regions of such a discriminant are always singly connected and convex. To see this, consider two points $x_A$ and $x_B$ both of which lie inside decision region $R_k$, as illustrated in Figure 4.3. Any point $\hat{x}$ that lies on the line connecting $x_A$ and $x_B$ can be expressed in the form

$$\hat{x} = \lambda x_A + (1 - \lambda) x_B$$

(4.11)

small

medium
luxury

$R_1$

Uncertain classification region

$R_2$

$R_3$

Figure: C. Bishop
Using Least Squares for Multiple Classes
(Solution)
We start by encoding the classes as a one-hot binary coding scheme.

Class 3 is encoded as $i_3$, the 3rd column of identity matrix $I_K$.

- We start by encoding the classes as a **one-hot** binary coding scheme.
- Class 3 is encoded as $i_3$, the 3rd column of identity matrix $I_K$. 

Linear Function

- Assign item with features $\mathbf{x}$ to class $k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x})$, $\forall j \neq k$

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

- The decision boundary is a $(D-1)$ dimensional hyperplane

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$$

for $D=2$
Least Squares Solution in Matrix Form

\[ y(x) = \tilde{W}^T \tilde{x} \]

where

\[ \tilde{W} = \begin{bmatrix} w_{10} & \cdots & w_{k0} & \cdots & w_{K0} \\ w_{1}^T & \cdots & w_{k}^T & \cdots & w_{K}^T \end{bmatrix} \]

and \( \tilde{x} = (1, x^T)^T \)

The least squares solution is

\[ \tilde{W} = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T T = \tilde{X}^\dagger T \]

where

\[ \tilde{X} = \begin{bmatrix} 1 & \cdots & 1 \\ x_1^T & \cdots & x_n^T \end{bmatrix} \]

and

\[ T = \begin{bmatrix} i_{y_1}, \cdots, i_{y_n} \end{bmatrix} \]

One hot encoding of class \( y_1 \)
Working with non-linearly separable data...
Linear Methods → Linear Boundaries?

Data from three classes, with linear decision boundaries

Linear boundaries in the five-dimensional space
\[ \phi(x) = (X_1, X_2, X_1X_2, X_1^2, X_2^2) \]

Linear inequalities in the transformed space are quadratic inequalities in the original space.
Tips

• Helps if $\phi_n = \phi(x_n)$ is binary, quantized or normalized
  • Example: if 1st feature $x_n(1)$ is age of user $n$, then
    • $\phi_n(1)$ could be indicator if $x_n(1)$ belongs to 1st quantile
    • $\phi_n(2)$ indicator whether $x_n(1)$ belongs to 2nd quantile
    • ...

• Useful to add interaction terms $\phi_h = \phi(x_n) \circ \phi(x_m)$, where
  “$\circ$” is the Hadamard product (or element-wise product)
  • XOR operator can be better than Hadamard product for binary variables.
Issues with Least Squares Classification

With square loss (score), optimization cares too much about reducing distance to boundary of well separable items...

...solution is to change the score function
Logistic Regression (for Classification)

- Back to two classes, \( C_1 \) and \( C_2 \)
- Logistic regression is often used for two classes
  \[ p(C_1 | \phi) = y(\phi) = \sigma \left( w^T \phi \right) \]
  where
  \[ \sigma(a) = \frac{\exp(a)}{1 + \exp(a)} \]
  \( \phi \equiv \phi(x) \)

- But can be easily generalized to \( K \) classes
Finding Logistic Regression Parameters (binary classification)

- For a dataset \( \{\phi_c, t_c\} \) where \( \phi_c \) is transformed feature vector of car \( c \), \( t_c \in \{0,1\} \) is the class of car \( c \in \text{TrainingDataCars} \)

- The likelihood function over the training data is

\[
p(t|w) = \prod_{c \in \text{TrainingDataCars}} y(\phi_c)^{t_c} (1 - y(\phi_c))^{1-t_c}
\]

where \( t = (t_1, \ldots, t_N)^T \), \( y(\phi_n) = \sigma(\mathbf{w}^T \phi_n) \)

- The log of the likelihood function (log-likelihood) is

\[
\log p(t|w) = \sum_{c \in \text{TrainingDataCars}} t_c \log y(\phi_c) + (1 - t_c) \log(1 - y(\phi_c))
\]

- To solve the above equation, we maximize the log-likelihood over parameters \( w \)

- The above equation is also described as cross-entropy loss, logistic loss, log loss, on Tensorflow, Sklearns, and pyTorch.
Solving Logistic Regression via Maximum Likelihood

• The above equations give the following gradient

\[ \nabla_w \log p(t|w) = \sum_{c \in \text{TrainingDataCars}} (t_c - y(\phi_c))\phi_c = \Phi^T(t - y) \]

• The logistic function has derivative

\[ \frac{d}{da} \sigma(a) = \sigma(a)(1 - \sigma(a)) \]

• The second derivative (Hessian) is then (verify!)

\[ H = \Phi^T R \Phi \]

where \( R_c = y(\phi_c)(1 - y(\phi_c)) \) and \( H \) is a positive definite matrix. Because \( H \) is positive definite the optimization is concave on \( w \) and has a unique maximum.

Exercise: what are the dimensions of \( \Phi \)?
Iterative Update

- The iterative parameter update is (Newton-Raphson)

\[
\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T (\mathbf{y} - \mathbf{t}) \\
= (\Phi^T \mathbf{R} \Phi)^{-1} \left\{ \Phi^T \mathbf{R} \Phi \mathbf{w}^{(\text{old})} - \Phi^T (\mathbf{y} - \mathbf{t}) \right\} \\
= (\Phi^T \mathbf{R} \Phi)^{-1} \Phi^T \mathbf{R} \mathbf{z}
\]

where \( \mathbf{z} \) is an \( N \)-dimensional vector with elements

\[
\mathbf{z} = \Phi \mathbf{w}^{(\text{old})} - \mathbf{R}^{-1} (\mathbf{y} - \mathbf{t})
\]
Training Logistic Regression with Data Selection Bias

• Let \( \pi_i \) be the probability of sampling example \( i \) in the training data
• We say a data sample is biased when \( \pi_i \) is not uniform
• Correcting for bias in the likelihood function:

\[
\log p(t|w) = \sum_{c \in \text{TrainingDataCars}} \frac{1}{\pi_c} \left( t_c \log y(\phi_c) + (1 - t_c) \log(1 - y(\phi_c)) \right)
\]

where \( C_c \) is the class of car \( c \).

• Generally, it is a good idea to test training data for different weights
  • The weights can be used to protect against sampling designs which could cause selection bias.
  • The weights can be used to protect against misspecification of the model.

• Unbalanced datasets:
  • Suppose there are more males than females in dataset. What will happen to decision boundary? How to fix it?
Multiclass Logistic Regression (MLR)

- Consider $K$ classes and $N$ observations
- Let $C_i$ be the class of the $i$-th example with feature vector $\phi_i$

$$P(C_c = t_c|\phi_c) = \frac{\exp(w_k^T \phi_c)}{\sum_{h=1}^{K} \exp(w_h^T \phi_c)}$$

a.k.a. softmax

- If we assume one-hot encoding of target variable $t_c$, the log-likelihood function is

$$\sum_{c \in \text{TrainingDataCars}} \sum_{k=1}^{K} t_{c,k} \log \frac{\exp(w_k^T \phi_c)}{\sum_{h=1}^{K} \exp(w_h^T \phi_c)}$$

$$= \sum_{c \in \text{TrainingDataCars}} \sum_{k=1}^{K} t_{c,k} w_k^T \phi_c - \sum_{c \in \text{TrainingDataCars}} \log \sum_{h=1}^{K} \exp(w_h^T \phi_c)$$