UVA CS 4501: Machine Learning

Lecture 13 Extra: More about Logistic Regression

Dr. Yanjun Qi

University of Virginia

Department of Computer Science

3/15/18
A Dataset for classification

\[ f : [X] \rightarrow [C] \]

Output as Discrete Class Label \( C_1, C_2, \ldots, C_L \)

\[
\arg \max_{C} P(C / X) \quad C = c_1, \ldots, c_L
\]

- **Data/points/instances/examples/samples/records:** [rows]
- **Features/attributes/dimensions/independent variables/covariates/predictors/regressors:** [columns, except the last]
- **Target/outcome/response/label/dependent variable:** special column to be predicted [last column]
Establishing a probabilistic model for classification

- Discriminative model

\[
\hat{c} = \arg \max_C P(C \mid X), \quad C = c_1, \ldots, c_L
\]

\[
\begin{align*}
P(c_1 \mid x) & \quad P(c_2 \mid x) \quad \cdots \quad P(c_L \mid x) \\
x_1 & \quad x_2 \quad \cdots \quad x_p
\end{align*}
\]

\[
x = (x_1, x_2, \ldots, x_p)
\]
A Dataset for classification

\[ f : [X] \longrightarrow [C] \]

Output as Discrete Class Label \( C_1, C_2, \ldots, C_L \)

- **Discriminative**
  \[
  \text{argmax}_{C} P(C \mid X) \quad C = c_1, \ldots, c_L
  \]

- **Generative**
  \[
  \text{argmax}_{C} P(C \mid X) = \text{argmax}_{C} P(X, C) = \text{argmax}_{C} P(X \mid C)P(C)
  \]

- Data/points/instances/examples/samples/records: [rows]
- Features/attributes/dimensions/independent variables/covariates/predictors/regressors: [columns, except the last]
- Target/outcome/response/label/dependent variable: special column to be predicted [last column]

3/15/18
Today: Extra

✓ Bayes Classifier and MAP Rule?
  ▪ Bayes Classifier
  ▪ Empirical Prediction Error
  ▪ 0-1 Loss function for Bayes Classifier

✓ Logistic regression
  ▪ Parameter Estimation for LR

\[
p(y|x) = \frac{e^{\beta x}}{1 + e^{\beta x}}
\]
Bayes classifiers

• Treat each feature attribute and the class label as random variables.
Bayes classifiers

• Treat each feature attribute and the class label as random variables.

• Given a sample \( \mathbf{x} \) with attributes \( (x_1, x_2, \ldots, x_p) \):
  – Goal is to predict its class \( C \).
  – Specifically, we want to find the value of \( C_i \) that maximizes \( p(C_i | x_1, x_2, \ldots, x_p) \).
Bayes classifiers

• Treat each feature attribute and the class label as random variables.

• Given a sample \( \mathbf{x} \) with attributes \((x_1, x_2, \ldots, x_p)\):
  – Goal is to predict its class \( C \).
  – Specifically, we want to find the value of \( C_i \) that maximizes \( p( C_i \mid x_1, x_2, \ldots, x_p ) \).

• Can we estimate \( p(C_i \mid \mathbf{x}) = p( C_i \mid x_1, x_2, \ldots, x_p ) \) directly from data?
Bayes classifiers

→ MAP classification rule

• Establishing a probabilistic model for classification

→ MAP classification rule
  – MAP: Maximum A Posterior
Bayes classifiers

MAP classification rule

- Establishing a probabilistic model for classification

MAP classification rule

- MAP: Maximum A Posterior
- Assign $x$ to $c^*$ if

$$P(C = c^* \mid X = x) > P(C = c \mid X = x)$$

for $c \neq c^*$, $c = c_1, \ldots, c_L$
Bayes classifiers

\[ \text{MAP classification rule} \]

- Establishing a probabilistic model for classification

- **MAP** classification rule
  - **MAP**: Maximum A Posterior
  - Assign \( x \) to \( c^* \) if

\[
P(C = c^* | X = x) > P(C = c | X = x) \quad \text{for} \quad c \neq c^*, \ c = c_1, \ldots, c_L
\]

\[
\left\{ \frac{P(C = c_1 | X)}{\sum_i P(C = c_i | X)} \right\} \max \Rightarrow C_i
\]

Adapt from Prof. Ke Chen NB slides
Bayes Classifiers – MAP Rule

Task: Classify a new instance \( X \) based on a tuple of attribute values \( X = \langle X_1, X_2, \ldots, X_p \rangle \) into one of the classes

\[
c_{MAP} = \arg\max_{c_j \in C} P(c_j \mid x_1, x_2, \ldots, x_p)
\]

MAP = Maximum Aposteriori Probability

WHY?
0-1 LOSS for Classification

- Procedure for categorical output variable $C$
  \[ L(k, \ell) = 0 \] if $k = \ell$
  \[ L(k, \ell) = 1 \] if $k \neq \ell$

- Frequently, 0-1 loss function used:
  \[ L(k, \ell) \]

- $L(k, \ell)$ is the price paid for misclassifying an element from class $C_k$ as belonging to class $C_\ell$

$\rightarrow L \times L$ matrix

$C_1, C_2, \ldots, C_L$
Expected prediction error (EPE)

- Expected prediction error (EPE), with expectation taken w.r.t. the joint distribution $\Pr(C,X)$

$$
\Pr(C,X) = \Pr(C \mid X) \Pr(X)
$$

$$
\mathbb{E}_{P(f)} = \mathbb{E}_{X,C}(L(C,f(X)))
$$

$$
= \mathbb{E}_{X} \sum_{k=1}^{L} L[C_{k},f(X)] \Pr(C_{k} \mid X)
\tag{e.g. 0-1 loss}
$$

Consider sample population distribution
\[ E_{PE}(f) = E_{\tilde{x}, c} \left( L(c, f(\tilde{x})) \right) \]
\[ = E_{\tilde{x}} \left( E_{c | \tilde{x}} \left[ L(c, f(\tilde{x})) \right] \right) \]
\[ = E_{\tilde{x}} \sum_{k=1}^{L} L [C_k, f(\tilde{x})] \Pr(C_k | \tilde{x}) \]

\[ \arg\min_{f} E_{PE}(f(\tilde{x})) \]
\[ \Rightarrow \text{Point-wise minimization} \quad \text{when} \quad \tilde{x} = x \]
\[ \Rightarrow \hat{f}(\tilde{x} = x) = \arg\min_{f(\tilde{x}) \in C} \sum_{k=1}^{L} L [C_k, f(\tilde{x})] \Pr(C_k | \tilde{x} = x) \]

\[ \Rightarrow \hat{f}(x) = \arg\max_{C \in C} \Pr(C_k | \tilde{x} = x) \]
Expected prediction error (EPE)

\[ E_{\text{P,E}}(f) = E_{X,C}(L(C,f(X))) = E_X \sum_{k=1}^{K} L(C_k,f(X)) \Pr(C_k | X) \]

- Pointwise minimization suffices

\[ \hat{f}(X) = \arg\min_{g \in C} \sum_{k=1}^{K} L(C_k,g) \Pr(C_k | X = x) \]

\[ \hat{f}(X) = C_k \text{ if } \Pr(C_k | X = x) = \max_{g \in C} \Pr(g | X = x) \]

Bayes Classifier

Consider sample population distribution
## SUMMARY: WHEN EPE USES DIFFERENT LOSS

<table>
<thead>
<tr>
<th>Loss Function</th>
<th>Estimator $\hat{f}(x)$</th>
</tr>
</thead>
</table>
| $L_2$         | $\text{EPE} = \mathbb{E}_{x,y} (y - f(x))^2$  
$\hat{f}(x) = E[Y|X = x]$ |
| $L_1$         | $\hat{f}(x) = \text{median}(Y|X = x)$ |
| 0-1           | $\hat{f}(x) = \arg\max_Y P(Y|X = x)$  
(Bayes classifier / MAP) |
Today: Extra

✓ Why Bayes Classification – MAP Rule?
  ▪ Empirical Prediction Error
  ▪ 0-1 Loss function for Bayes Classifier

✓ Logistic regression
  ▪ Parameter Estimation for LR

\[ p(y|x) = \frac{e^{\beta x}}{1 + e^{\beta x}} \]
Newton’s method for optimization

- The most basic second-order optimization algorithm
- Updating parameter with

\[ \theta_{k+1} = \theta_k - H_k^{-1}g_k \]
Review: Hessian Matrix / n==2 case

Singlevariate → multivariate

• 1\textsuperscript{st} derivative to gradient,
  \[ g = \nabla f = \left( \begin{array}{c} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{array} \right) \]

• 2\textsuperscript{nd} derivative to Hessian
  \[ H = \left( \begin{array}{cc} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{array} \right) \]
Review: Hessian Matrix

Suppose that $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a function that takes a vector in $\mathbb{R}^n$ and returns a real number. Then the **Hessian** matrix with respect to $x$, written $\nabla_x^2 f(x)$ or simply as $H$ is the $n \times n$ matrix of partial derivatives,

$$
\nabla_x^2 f(x) \in \mathbb{R}^{n \times n} = 
\begin{bmatrix}
\frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\
\frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2}
\end{bmatrix}.
$$
Newton’s method for optimization

- Making a quadratic/second-order Taylor series approximation

\[
f_{quad}(\theta) = f(\theta_k) + g_k^T(\theta - \theta_k) + \frac{1}{2} (\theta - \theta_k)^T H_k (\theta - \theta_k)
\]

Finding the minimum solution of the above right quadratic approximation (quadratic function minimization is easy!)
\[ \hat{f}(\theta) = f(\theta_k) + g_k^T (\theta - \theta_k) + \frac{1}{2} (\theta - \theta_k)^T H_k (\theta - \theta_k) \]
\[ \frac{\partial f(\theta)}{\partial \theta} = 0 + g_k + \frac{1}{2} H_k \theta - \frac{1}{2} H_k \theta_k := 0 \]
\[ g_k + H_k (\theta - \theta_k) = 0 \]
\[ \Rightarrow \theta = \theta_k - H_k^{-1} g_k \]

See p.24 handout

where \( H_k \in \mathbb{R}^{p \times p} \) and \( g_k \in \mathbb{R}^p \)
Newton’s Method / second-order Taylor series approximation
Newton’s Method / second-order Taylor series approximation
Newton’s Method / second-order Taylor series approximation
Newton’s Method / second-order Taylor series approximation
Newton’s Method

• At each step:

\[ \theta_{k+1} = \theta_k - \frac{f'(\theta_k)}{f''(\theta_k)} \]

\[ \theta_{k+1} = \theta_k - H^{-1}(\theta_k) \nabla f(\theta_k) \]

• Requires 1\textsuperscript{st} and 2\textsuperscript{nd} derivatives
• Quadratic convergence
• \( \Rightarrow \) However, finding the inverse of the Hessian matrix is often expensive
Newton vs. GD for optimization

- **Newton**: a quadratic/second-order Taylor series approximation
  \[
  f_{\text{quad}}(\theta) = f(\theta_k) + g_k^T(\theta - \theta_k) + \frac{1}{2}(\theta - \theta_k)^T H_k(\theta - \theta_k)
  \]

- **GD**: a approximation
  \[
  \theta_{k+1} = \theta_k - \frac{1}{\alpha} g(\theta_k)
  \]

Finding the minimum solution of the above right quadratic approximation (quadratic function minimization is easy!)
Comparison

• Newton’s method vs. Gradient descent

A comparison of gradient descent (green) and Newton's method (red) for minimizing a function (with small step sizes).

Newton’s method uses curvature information to get a more direct route ...
Parameter Estimation for LR
→ MLE from the data

• **RECAP:** Linear regression → Least squares

• Logistic regression: → Maximum likelihood estimation
MLE for Logistic Regression Training

Let’s fit the logistic regression model for $K=2$, i.e., number of classes is 2

Training set: $(x_i, y_i), i=1,\ldots,N$

(conditional )

Log-likelihood:

$$l(\beta) = \sum_{i=1}^{N} \{ \log \Pr(Y = y_i | X = x_i) \}$$

$$= \sum_{i=1}^{N} y_i \log(\Pr(Y = 1 | X = x_i)) + (1 - y_i) \log(\Pr(Y = 0 | X = x_i))$$

$$= \sum_{i=1}^{N} (y_i \log \frac{\exp(\beta^T x_i)}{1 + \exp(\beta^T x_i)}) + (1 - y_i) \log \frac{1}{1 + \exp(\beta^T x_i)}$$

$$= \sum_{i=1}^{N} (y_i \beta^T x_i - \log(1 + \exp(\beta^T x_i)))$$

$x_i$ are $(p+1)$-dimensional input vector with leading entry 1

$\beta$ is a $(p+1)$-dimensional vector

For Bernoulli distribution

$p(y | x)^y (1 - p)^{1-y}$

We want to maximize the log-likelihood in order to estimate $\beta$
\[ l(\beta) = \sum_{i=1}^{N} \{ \log \text{Pr}(Y = y_i | X = x_i) \} \]

\[
\log \text{Pr}(Y = y_i | X = x_i) = \log \left\{ P(y_i=1 | x) ^ {y_i} (1-P(y_i=1 | x)) ^ {1-y_i} \right\}
\]

\[= y_i \log P(y_i=1 | x) + (1-y_i) \log (1-P(y_i=1 | x)) \]
Newton-Raphson for LR (optional)

\[ \frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} (y_i - \frac{\exp(\beta^T x_i)}{1+\exp(\beta^T x_i)}) x_i = 0 \]

(p+1) Non-linear equations to solve for (p+1) unknowns

Solve by Newton-Raphson method:

\[ \beta^{new} \leftarrow \beta^{old} - \left[ \left( \frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} \right) \right]^{-1} \frac{\partial l(\beta)}{\partial \beta} , \]

where, \[ \left( \frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} \right) = -\sum_{i=1}^{N} x_i x_i^T \left( \frac{\exp(\beta^T x_i)}{1+\exp(\beta^T x_i)} \right) \left( \frac{1}{1+\exp(\beta^T x_i)} \right) \]

minimizes a quadratic approximation to the function we are really interested in.

\[ \theta_{k+1} = \theta_k - H_K^{-1} g_k \]

\( \rho(x_i ; \beta) \)

\( 1 - \rho(x_i ; \beta) \)
Newton-Raphson for LR...

\[
\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} (y_i - \frac{\exp(\beta^T x_i)}{1 + \exp(\beta^T x_i)}) x_i = X^T (y - p)
\]

\[
\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} = -X^T W X
\]

So, NR rule becomes:

\[
\beta^{new} \leftarrow \beta^{old} + (X^T W X)^{-1} X^T (y - p),
\]

\[
x = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_N^T \end{bmatrix}_{N \times (p+1)}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}_{N \times 1}, \quad p = \begin{bmatrix} \exp(\beta^T x_1) / (1 + \exp(\beta^T x_1)) \\ \exp(\beta^T x_2) / (1 + \exp(\beta^T x_2)) \\ \vdots \\ \exp(\beta^T x_N) / (1 + \exp(\beta^T x_N)) \end{bmatrix}_{N \times 1},
\]

\[
X : N \times (p+1) \text{ matrix of } x_i \quad y : N \times 1 \text{ matrix of } y_i \\
p : N \times 1 \text{ matrix of } p(x_i; \beta^{old}) \\
W : N \times N \text{ diagonal matrix of } p(x_i; \beta^{old})(1 - p(x_i; \beta^{old}))
\]

\[
\left( \frac{\exp(\beta^T x_i)}{1 + \exp(\beta^T x_i)} \right) (1 - \frac{1}{1 + \exp(\beta^T x_i)})
\]
Newton-Raphson for LR...

- Newton-Raphson
  \[
  \beta^{new} = \beta^{old} + (X^T WX)^{-1} X^T (y - p)
  = (X^T WX)^{-1} X^T W (X\beta^{old} + W^{-1} (y - p))
  = (X^T WX)^{-1} X^T Wz
  \]

- Adjusted response
  \[
  z = X\beta^{old} + W^{-1} (y - p)
  \]

- Iteratively reweighted least squares (IRLS)
  \[
  \beta^{new} \leftarrow \arg\min_{\beta} (z - X\beta^T)^T W (z - X\beta^T)
  \]
  \[
  \leftarrow \arg\min_{\beta} (y - p)^T W^{-1} (y - p)
  \]

Re expressing Newton step as weighted least square step
References

- Prof. Tan, Steinbach, Kumar’s “Introduction to Data Mining” slide
- Prof. Andrew Moore’s slides
- Prof. Eric Xing’s slides
- Prof. Ke Chen NB slides