Probabilistic Methods for Classification

Jun Zhu
dcszj@mail.tsinghua.edu.cn
http://ml.cs.tsinghua.edu.cn/~jun
State Key Lab of Intelligent Technology & Systems
Tsinghua University

September 24, 2019
Outline

- Probabilistic methods for supervised learning
- Naive Bayes classifier
- Logistic regression
- Exponential family distributions
- Generalized linear models
An Intuitive Example

[diagram showing a scatter plot with points representing Grasshoppers and Katydid species, with labels for Antenna Length and Abdomen Length]
With more data …

Build a histogram, e.g., for “Antenna length”
Empirical distribution

- **Histogram** (or empirical distribution)

- Smooth with kernel density estimation (KDE):

[Courtesy of E. Keogh]
Classification?

Classify another insect we find. Its antennae are 3 units long.
Is it more probable that the insect is a **Grasshopper** or a **Katydid**?

[Antennae length is 3]

[Courtesy of E. Keogh]
Classification Probability

\[ P(\text{Grasshopper} \mid 3) = \frac{10}{10 + 2} = 0.833 \]
\[ P(\text{Katydid} \mid 3) = \frac{2}{10 + 2} = 0.166 \]

Antennae length is 3

[Courtesy of E. Keogh]
Classification Probability

\[ P(\text{Grasshopper} \mid 7) = \frac{3}{3+9} = 0.250 \]
\[ P(\text{Katydid} \mid 7) = \frac{9}{3+9} = 0.750 \]

Antennae length is 7
Classification Probability

\[
P(\text{Grasshopper} \mid 5) = \frac{6}{6 + 6} = 0.500
\]

\[
P(\text{Katydid} \mid 5) = \frac{6}{6 + 6} = 0.500
\]

Antennae length is 5

[Courtesy of E. Keogh]
Naïve Bayes Classifier

The simplest “category-feature” generative model:

- **Category**: “bird”, “Mammal”
- **Features**: “has beak”, “can fly” …
Naïve Bayes Classifier

A mathematic model:

- **Naive Bayes assumption**: features $X_1, \ldots, X_d$ are conditionally independent given the class label $Y$

$$p(x, y) = p(y)p(x|y)$$

- **Prior**
- **Likelihood**
Naïve Bayes Classifier

A mathematic model:

\[
p(y|x) = \frac{p(x, y)}{p(x)} = \frac{p(y)p(x|y)}{p(x)}
\]

Bayes’ decision rule:

\[
y^* = \arg \max_{y \in Y} p(y|x)
\]
**Bayes Error**

*Theorem*: Bayes classifier is optimal!

\[
p(error|x) = \begin{cases} 
    p(y = 1|x) & \text{if we decide } y = 0 \\
    p(y = 0|x) & \text{if we decide } y = 1 
\end{cases}
\]

\[
p(error) = \int_{-\infty}^{\infty} p(error|x)p(x)dx
\]
However, the true distribution is unknown.

Learning!

- We need to estimate it!
Naïve Bayes Classifier

How to learn model parameters?

- Assume $X$ are $d$ binary features, $Y$ has 2 possible labels

$$p(y|\pi) = \begin{cases} \pi & \text{if } y = 1 \text{ (i.e., bird)} \\ 1 - \pi & \text{otherwise} \end{cases}$$

- How many parameters to estimate?

$$p(x_j|y = 0, q) = \begin{cases} q_{0j} & \text{if } x_j = 1 \\ 1 - q_{0j} & \text{otherwise} \end{cases} \quad p(x_j|y = 1, q) = \begin{cases} q_{1j} & \text{if } x_j = 1 \\ 1 - q_{1j} & \text{otherwise} \end{cases}$$

- Has beak? - $X_1$
- Can fly? - $X_2$
- Has fur? - $X_3$
- Has four legs? - $X_4$
Naïve Bayes Classifier

- How to learn model parameters?
- A set of training data:
  - (1, 1, 0, 0; 1)
  - (1, 0, 0, 0; 1)
  - (0, 1, 1, 0; 0)
  - (0, 0, 1, 1; 0)
- Maximum likelihood estimation \((N: \# \text{ of training data})\)

\[
p(\{x_i, y_i|\pi, q\}) = \prod_{i=1}^{N} p(x_i, y_i|\pi, q)
\]
Naïve Bayes Classifier

**Maximum likelihood estimation** \((N: \# \text{ of training data})\)

\[
(\hat{\pi}, \hat{q}) = \text{arg max}_{\pi, q} p(\{x_i, y_i\} | \pi, q)
\]

\[
(\hat{\pi}, \hat{q}) = \text{arg max}_{\pi, q} \log p(\{x_i, y_i\} | \pi, q)
\]

**Results** *(count frequency! Exercise?)*:

\[
\hat{\pi} = \frac{N_1}{N} \quad \hat{q}_{0j} = \frac{N_0^j}{N_0} \quad \hat{q}_{1j} = \frac{N_1^j}{N_1}
\]

\[
N_k = \sum_{i=1}^{N} I(y_i = k) : \# \text{ of data in category } k
\]

\[
N_k^j = \sum_{i=1}^{N} I(y_i = k, x_{i,j} = 1) : \# \text{ of data in category } k \text{ that has feature } j
\]
Naïve Bayes Classifier

Data scarcity issue (zero-counts problem):

\[ \hat{\pi} = \frac{N_1}{N} \quad \hat{q}_{0j} = \frac{N_{0j}}{N_0} \quad \hat{q}_{1j} = \frac{N_{1j}}{N_1} \]

- How about if some features do not appear?

Laplace smoothing (Additive smoothing):

\[ \hat{q}_{0j} = \frac{N_{0j} + \alpha}{N_0 + 2\alpha} \quad \alpha > 0 \]

\[ \hat{q}_{1j} = \frac{N_{1j} + \alpha}{N_1 + 2\alpha} \]
A Bayesian Treatment

Put a prior on the parameters

\[ p_0(q_{0j} | \alpha_1, \alpha_2) = \text{Beta}(\alpha_1, \alpha_2) = \frac{\Gamma(\alpha_1 + \alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} q_{0j}^{\alpha_1 - 1} (1 - q_{0j})^{\alpha_2 - 1} \]
A Bayesian Treatment

Maximum a Posterior Estimate (MAP):

\[
\hat{q} = \arg \max_q \log p(q|\{x_i, y_i\})
\]

\[
= \arg \max_q \log p_0(q) + \log p(\{x_i, y_i\}|q)
\]

Results (Exercise?):

\[
\hat{q}_{0j} = \frac{N_0^j + \alpha_1 - 1}{N_0 + \alpha_1 + \alpha_2 - 2}
\]

\[
\hat{q}_{1j} = \frac{N_1^j + \alpha_1 - 1}{N_1 + \alpha_1 + \alpha_2 - 2}
\]
A Bayesian Treatment

Maximum a Posterior Estimate (MAP):

$$\hat{q}_{0j} = \frac{N^j_0 + \alpha_1 - 1}{N_0 + \alpha_1 + \alpha_2 - 2}$$

- If $\alpha_1 = \alpha_2 = 1$ (non-informative prior), no effect
  - MLE is a special case of Bayesian estimate
- Increase $\alpha_1, \alpha_2$, lead to heavier influence from prior
Bayesian Regression

Goal: learn a function from noisy observed data

- Linear
  \[ \mathcal{F}_{\text{linear}} = \{ f : f = wx + b, \ w, b \in \mathbb{R} \} \]

- Polynomial
  \[ \mathcal{F}_{\text{polynomial}} = \{ f : f = \sum_{k} w_k x^k, \ w_k \in \mathbb{R} \} \]

- …
Bayesian Regression

- Noisy observations
  \[ y = f(x) + \epsilon, \text{ where } \epsilon \sim \mathcal{N}(0, \sigma_n^2) \]

- Gaussian likelihood function for linear regression \( f(x_i) = w^\top x_i \)
  \[ p(y|X,w) = \prod_{i=1}^{N} p(y_i|x_i,w) = \mathcal{N}(X^\top w, \sigma_n^2 I) \]

- Gaussian prior (Conjugate)
  \[ w \sim \mathcal{N}(0, \Sigma_d) \]

- Inference with Bayes’ rule
  - Posterior
    \[ p(w|X,y) = \mathcal{N}(\frac{1}{\sigma_n^2} A^{-1} X y, A^{-1}), \text{ where } A = \sigma_n^{-2} XX^\top + \Sigma_d^{-1} \]
  - Marginal likelihood
  - Prediction
    \[ p(y|X) = \int p(y|X,w)p(w)dw \]
    \[ p(f_*|x_*,X,y) = \int p(f_*|x_*,w)p(w|X,y)dw = \mathcal{N}\left(\frac{1}{\sigma_n^2} x_*^\top A^{-1} X y, x_*^\top A^{-1} x_*\right) \]
Extensions of NB

We covered the case with binary features and binary class labels

NB is applicable to the cases:
- Discrete features + discrete class labels
- Continuous features + discrete class labels
- ...

More dependency between features can be considered
- Tree augmented NB
- ...

Gaussian Naive Bayes (GNB)

- E.g.: character recognition: feature $X_i$ is intensity at pixel $i$:

- The generative process is
  \[ Y \sim \text{Bernoulli}(\pi) \]
  \[ P(X_i|Y = y) = \mathcal{N}(\mu_{iy}, \sigma_{iy}^2) \]
  - Different mean and variance for each class $k$ and each feature $i$

- Sometimes assume variance is:
  - independent of $Y$ (i.e., $\sigma_i$ )
  - or independent of $X$ (i.e., $\sigma_y$)
  - or both (i.e., $\sigma$)
Estimating Parameters & Prediction

MLE estimates

$$\hat{\mu}_{ik} = \frac{1}{\sum_n \mathbb{I}(y_n = k)} \sum_n x_{ni} \mathbb{I}(y_n = k)$$

Prediction:

$$h(x) = \arg\max_y P(y) \prod_i P(x_i | y)$$
What you need to know about NB classifier

- What’s the assumption
- Why we use it
- How do we learn it
- Why is Bayesian estimation (MAP) important
Linear regression and linear classification

- Linear fit
- Linear decision boundary

Mathematical expressions:

\[ w^T x + b = 0 \]

\[ w^T x + b > 0 \]

\[ w^T x + b < 0 \]
What’s the decision boundary of NB?

Is it linear or non-linear?

There are several distributions that lead to a linear decision boundary, e.g., GNB with equal variance

\[
P(X_i | Y = y) = \mathcal{N}(\mu_{iy}, \sigma_i^2)
\]

Decision boundary (??):

\[
\log \frac{\prod_{i=1}^{d} P(X_i | Y = 0) P(Y = 0)}{\prod_{i=1}^{d} P(X_i | Y = 1) P(Y = 1)} = 0
\]

\[
\Rightarrow \log \frac{1 - \pi}{\pi} + \sum_i \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} + \sum_i \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} x_i = 0
\]

\[
\Rightarrow w_0 + \sum_i w_i x_i = 0
\]
Gaussian Naive Bayes (GNB)

- Decision boundary (the general multivariate Gaussian case):

\[
P_1 = P(Y = 0), \quad P_2 = P(Y = 1)
\]

\[
p_1(X) = p(X|Y = 0) = \mathcal{N}(M_1, \Sigma_1)
\]

\[
p_2(X) = p(X|Y = 1) = \mathcal{N}(M_2, \Sigma_2)
\]
The predictive distribution of GNB

- Understanding the predictive distribution

\[ p(y = 1|\mathbf{x}, \mu, \Sigma, \pi) = \frac{p(y = 1, \mathbf{x}|\mu, \Sigma, \pi)}{p(\mathbf{x}|\mu, \Sigma, \pi)} \]

- Under naive Bayes assumption:

\[
p(y = 1|\mathbf{x}, \mu, \Sigma, \pi) = \frac{1}{1 + \frac{p(y=0, \mathbf{x}|\mu, \Sigma, \pi)}{p(y=1, \mathbf{x}|\mu, \Sigma, \pi)}}
= \frac{1}{1 + \frac{(1-\pi) \prod_i \mathcal{N}(x_i|\mu_{i0},\sigma_i^2)}{\pi \prod_i \mathcal{N}(x_i|\mu_{i1},\sigma_i^2)}}
= \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x} - w_0)}
\]

- **Note:** For multi-class, the predictive distribution is softmax!
Generative vs. Discriminative Classifiers

- **Generative classifiers** (e.g., Naive Bayes)
  - Assume some functional form for \( P(X,Y) \) (or \( P(Y) \) and \( P(X|Y) \))
  - Estimate parameters of \( P(X,Y) \) directly from training data
  - Make prediction
    \[
    \hat{y} = \arg\max_y P(x, Y = y)
    \]
  - But, we note that
    \[
    \hat{y} = \arg\max_y P(Y = y|x)
    \]
- **Discriminative classifiers** (e.g., Logistic regression)
  - Assume some functional form for \( P(Y|X) \)
  - Estimate parameters of \( P(Y|X) \) directly from training data

Why not learn \( P(Y|X) \) directly? Or, why not learn the decision boundary directly?
Logistic Regression

Recall the predictive distribution of GNB!

Assume the following functional form for $P(Y | X)$

$$P(y = 1 | x) = \frac{1}{1 + \exp(-(w_0 + w^T x))}$$

- Logistic function (or Sigmoid) applied to a linear function of the data (for $\alpha = 1$):

$$\psi_\alpha(v) = \frac{1}{1 + \exp(-\alpha v)}$$

$a \to \infty$: step function

use a large $\alpha$ can be good for some neural networks
Logistic Regression

What’s the decision boundary of logistic regression? (linear or nonlinear?)

\[
P(y = 1 | x) = \frac{1}{1 + \exp(-(w_0 + w^T x))}
\]

\[
\log \frac{P(Y = 1 | x)}{P(y = 0 | x)} = 0
\]

\[
w^T x + w_0 = 0
\]

Logistic regression is a linear classifier!
Representation

- Logistic regression

\[ P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-(w_0 + \mathbf{w}^\top \mathbf{x}))} \]

- For notation simplicity, we use the augmented vector:

  \[
  \begin{pmatrix}
  1 \\
  \mathbf{x}
  \end{pmatrix}
  \text{ input features : } \begin{pmatrix}
  w_0 \\
  \mathbf{w}
  \end{pmatrix}
  \text{ model weights : }
  \]

- Then, we have

\[ P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})} \]
Multiclass Logistic Regression

For more than 2 classes, where \( y \in \{1, \ldots, K\} \), logistic regression classifier is defined as

\[
\forall k < K : \quad P(Y = k | x) = \frac{\exp(w_k^\top x)}{1 + \sum_{j=1}^{K-1} \exp(w_j^\top x)}
\]

\[
P(Y = K | x) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(w_j^\top x)}
\]

- Well normalized distribution! No weights for class K!

- Is the decision boundary still linear?
Training Logistic Regression

—we consider the binary classification

\[ P(y = 1|x) = \frac{1}{1 + \exp(-w^\top x)} \]

—training data \( D = \{ (x_i, y_i) \}_{i=1}^N \)

—how to learn the parameters?

—can we do MLE?

\[ \hat{w}_{MLE} = \arg\max_w \prod_{i=1}^N P(x_i, y_i | w) \]

—no! don’t have a model for \( P(X) \) or \( P(X | Y) \)

—can we do large-margin learning?
Maximum Conditional Likelihood Estimate

We learn the parameters by solving

\[
\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \prod_{i=1}^{N} P(y_i | x_i, \mathbf{w})
\]

**Discriminative philosophy** – don’t waste effort on learning \( P(X) \), focus on \( P(Y | X) \) – that’s all that matters for classification!
Maximum Conditional Likelihood Estimate

\[ \hat{w} = \arg \max_w \prod_{i=1}^{N} P(y_i \mid x_i, w) \]

\[ P(y = 1 \mid x) = \frac{1}{1 + \exp(-w^\top x)} \]

We have:

\[ \mathcal{L}(w) = \log \prod_{i=1}^{N} P(y_i \mid x_i, w) \]

\[ = \sum_i \left[ y_i w^\top x_i - \log(1 + \exp(w^\top x_i)) \right] \]
Maximum Conditional Likelihood Estimate

\[ \hat{w} = \arg\max_w \mathcal{L}(w) \]

\[ \mathcal{L}(w) = \sum_i [y_i w^\top x_i - \log(1 + \exp(w^\top x_i))] \]

- **Bad news:** no closed-form solution!
- **Good news:** \( \mathcal{L}(w) \) is a concave function of \( w \)!
  - Is the original logistic function concave?

Read [S. Boyd, Convex Optimization, Chap. 1] for an introduction to convex optimization.
Optimizing concave/convex function

- Conditional likelihood for logistic regression is concave
- Maximum of a concave function = minimum of a convex function
  - Gradient ascent (concave) / Gradient descent (convex)

Gradient:
\[ \nabla_w \mathcal{L}(w) = \begin{pmatrix} \frac{\partial \mathcal{L}(w)}{\partial w_0} \\ \vdots \\ \frac{\partial \mathcal{L}(w)}{\partial w_d} \end{pmatrix} \]

Update rule:
\[ w_{t+1} = w_t + \eta \nabla_w \mathcal{L}(w)|_{w_t} \]
Gradient Ascent for Logistic Regression

Property of sigmoid function

\[ \nabla_v \psi = \psi(1 - \psi) \]

Gradient ascent algorithm iteratively does:

\[ \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \sum_{i=1}^{N} \mathbf{x}_i \left( y_i - \mu^t_i \right) \]

- where \( \mu^t_i = P(y = 1 | \mathbf{x}_i, \mathbf{w}_t) \) is the prediction made by the current model

Until the change (of objective or gradient) falls below some threshold
Issues

- Gradient descent is the simplest optimization methods, faster convergence can be obtained by using
  - E.g., Newton method, conjugate gradient ascent, IRLS (iterative reweighted least squares)

- The vanilla logistic regression often over-fits; using a regularization can help a lot!
Effects of step-size

- Large $\eta$ $\Rightarrow$ fast convergence but larger residual error; Also possible oscillations
- Small $\eta$ $\Rightarrow$ slow convergence but small residual error
The Newton’s Method

AKA: Newton-Raphson method

A method that finds the root of: \( f(x) = 0 \)

\[
x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}
\]
The Newton’s Method

- To maximize the conditional likelihood

\[ \mathcal{L}(w) = \sum_i [y_i w^\top x_i - \log(1 + \exp(w^\top x_i))] \]

- We need to find \( w^* \) such that

\[ \nabla \mathcal{L}(w^*) = 0 \]

- So we can perform the following iteration:

\[ w_{t+1} \leftarrow w_t - H^{-1} \nabla_w \mathcal{L}(w)|_{w_t} \]

- where \( H \) is known as the Hessian matrix:

\[ H = \nabla^2_w \mathcal{L}(w)|_{w_t} \]
Newton’s Method for LR

The update equation

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - H^{-1} \nabla_w \mathcal{L}(\mathbf{w})|_{\mathbf{w}_t}$$

where the gradient is:

$$\nabla_w \mathcal{L}(\mathbf{w})|_{\mathbf{w}_t} = \sum_i (y_i - \mu_i) \mathbf{x}_i = X(\mathbf{y} - \mathbf{\mu})$$

$$\mu_i = \psi(\mathbf{w}_t^\top \mathbf{x}_i)$$

The Hessian matrix is:

$$H = \nabla^2_w \mathcal{L}(\mathbf{w})|_{\mathbf{w}_t} = -\sum_i \mu_i (1 - \mu_i) \mathbf{x}_i \mathbf{x}_i^\top = -X R X^\top$$

where $$R_{ii} = \mu_i (1 - \mu_i)$$
Iterative reweighted least squares (IRLS)

In least square estimate of linear regression, we have

\[ w = (XX^\top)^{-1}Xy \]

Now, for logistic regression

\[ w_{t+1} = w_t - H^{-1}\nabla_w \mathcal{L}(w_t) \]
\[ = w_t - (XRX^\top)^{-1}X(\mu - y) \]
\[ = (XRX^\top)^{-1}\{XRX^\top w_t - X(\mu - y)\} \]
\[ = (XRX^\top)^{-1}XRz \]

where \( z = X^\top w_t - R^{-1}(\mu - y) \)
Convergence curves

Legend: X-axis: Iteration #; Y-axis: classification error

- In each figure, red for IRLS and blue for gradient descent
LR: Practical Issues

- IRLS takes $O(N + d^3)$ per iteration, where $N$ is # training points and $d$ is feature dimension, but converges in fewer iterations.

- Quasi-Newton methods, that approximate the Hessian, work faster.

- Conjugate gradient takes $O(Nd)$ per iteration, and usually works best in practice.

- Stochastic gradient descent can also be used if $N$ is large c.f. perceptron rule.
Gaussian NB vs. Logistic Regression

<table>
<thead>
<tr>
<th>GNB</th>
<th>LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian parameters</td>
<td>Regression parameters</td>
</tr>
</tbody>
</table>

Representation equivalence
- But only in some special case! (GNB with class independent variances)

What’s the differences?
- LR makes no assumption about $P(X \mid Y)$ in learning
- They optimize different functions, obtain different solutions
Generative vs. Discriminative

Given infinite data (asymptotically)

- (1) If conditional independence assumption holds, discriminative and generative NB perform similar

\[ \epsilon_{\text{Dis},\infty} \sim \epsilon_{\text{Gen},\infty} \]

- (2) If conditional independence assumption does NOT hold, discriminative outperform generative NB

\[ \epsilon_{\text{Dis},\infty} < \epsilon_{\text{Gen},\infty} \]

[Ng & Jordan, NIPS 2001]
Generative vs. Discriminative

Given finite data \((N \text{ data points, } d \text{ features})\)

\[
\epsilon_{\text{Dis},n} \leq \epsilon_{\text{Dis},\infty} + O \left( \sqrt{\frac{d}{N}} \right)
\]

\[
\epsilon_{\text{Gen},n} \leq \epsilon_{\text{Gen},\infty} + O \left( \sqrt{\frac{\log d}{N}} \right)
\]

- Naive Bayes (generative) requires \(N = O(\log d)\) to converge to its asymptotic error, whereas logistic regression (discriminative) requires \(N = O(d)\).

Why?

- “Independent class conditional densities” – parameter estimates are not coupled, each parameter is learnt independently, not jointly, from training data.
Experimental Comparison

- UCI Machine Learning Repository 15 datasets, 8 continuous features, 7 discrete features
What you need to know

LR is a linear classifier
- Decision boundary is a hyperplane
LR is learnt by maximizing conditional likelihood
- No closed-form solution
- Concave! Global optimum by gradient ascent methods
GNB with class-independent variances representationally equivalent to LR
- Solutions differ because of objective (loss) functions
In general, NB and LR make different assumptions
- NB: features independent given class, assumption on P(X | Y)
- LR: functional form of P(Y | X), no assumption on P(X | Y)
Convergence rates:
- GNB (usually) needs less data
- LR (usually) gets to better solutions in the limit
Exponential family

For a numeric random variable \( X \)

\[
p(x|\eta) = h(x) \exp \left( \eta^\top T(x) - A(\eta) \right)
\]

\[
= \frac{1}{Z(\eta)} h(x) \exp \left( \eta^\top T(x) \right)
\]

is an **exponential family distribution** with natural (canonical) parameter \( \eta \)

Function \( T(x) \) is a **sufficient statistic**.

Function \( A(\eta) = \log Z(\eta) \) is the log normalizer.

Examples: Bernoulli, multinomial, Gaussian, Poisson, gamma,…
Recall Linear Regression

Let us assume that the target variable and the inputs are related by the equation:

$$y_i = \theta^\top x_i + \epsilon_i$$

where $\epsilon$ is an error term of unmodeled effects or random noise.

Now assume that $\epsilon$ follows a Gaussian $N(0,\sigma)$, then we have:

$$p(y_i|x_i, \theta) = \frac{1}{\sqrt{2\pi\sigma}} \exp \left( -\frac{(y_i - \theta^\top x_i)^2}{2\sigma^2} \right)$$
Recall: Logistic Regression (sigmoid classifier)

The condition distribution: a Bernoulli

\[ p(y|x) = \mu(x)^y (1 - \mu(x))^{1-y} \]

where \( \mu \) is a logistic function

\[ \mu(x) = \frac{1}{1 + e^{-\theta^T x}} \]

We can use the brute-force gradient method as in LR

But we can also apply generic laws by observing the \( p(y|x) \) is an exponential family function, more specifically, a generalized linear model!
Example: Multivariate Gaussian Distribution

For a continuous vector random variable \( \mathbf{x} \in \mathbb{R}^d \):

\[
p(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp \left( -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)
\]

\[
= \frac{1}{(2\pi)^{d/2}} \exp \left( -\frac{1}{2} \text{tr}(\Sigma^{-1} \mathbf{x} \mathbf{x}^\top) + \boldsymbol{\mu}^\top \Sigma^{-1} \mathbf{x} - \frac{1}{2} \boldsymbol{\mu}^\top \Sigma^{-1} \boldsymbol{\mu} - \log |\Sigma| \right)
\]

Exponential family representation

\[
\eta = \left[ \Sigma^{-1} \boldsymbol{\mu}; -\frac{1}{2} \text{vec}(\Sigma^{-1}) \right] = [\eta_1; \text{vec}(\eta_2)], \quad \eta_1 = \Sigma^{-1} \boldsymbol{\mu} \text{ and } \eta_2 = -\frac{1}{2} \Sigma^{-1}
\]

\[
T(\mathbf{x}) = [\mathbf{x}; \text{vec}(\mathbf{x} \mathbf{x}^\top)]
\]

\[
A(\eta) = \frac{1}{2} \boldsymbol{\mu}^\top \Sigma^{-1} \boldsymbol{\mu} + \log |\Sigma| = -\frac{1}{2} \text{tr}(\eta_2 \eta_1 \eta_1^\top) - \frac{1}{2} \log(-2|\eta_2|)
\]

\[
h(\mathbf{x}) = (2\pi)^{-d/2}
\]

Note: a \( d \)-dimensional Gaussian is a \((d + d^2)\)-parameter distribution with a \((d + d^2)\)-element vector of sufficient statistics (but because of symmetry and positivity, parameters are constrained and have lower degree of freedom)
Example: Multinomial distribution

For a binary vector random variable \( \mathbf{x} \sim \text{multi}(\mathbf{x} | \pi) \):

\[
p(\mathbf{x} | \pi) = \prod_{i=1}^{d} \pi_i^{x_i} = \exp \left( \sum_{i} x_i \ln \pi_i \right)
\]

\[
= \exp \left( \sum_{i=1}^{d-1} x_i \ln \pi_i + \left( 1 - \sum_{i=1}^{d-1} x_i \right) \ln \left( 1 - \sum_{i=1}^{d-1} \pi_i \right) \right)
\]

\[
= \exp \left( \sum_{i=1}^{d-1} x_i \ln \frac{\pi_i}{1 - \sum_{i=1}^{d-1} \pi_i} + \ln \left( 1 - \sum_{i=1}^{d-1} \pi_i \right) \right)
\]

Exponential family representation

\[
\eta = [\ln(\pi_i/\pi_d); 0]
\]

\[
T(\mathbf{x}) = \mathbf{x}
\]

\[
A(\eta) = -\ln \left( 1 - \sum_{i=1}^{d-1} \pi_i \right) = \ln \left( \sum_{i=1}^{d} e^{\eta_i} \right)
\]

\[
h(\mathbf{x}) = 1
\]
Why exponential family?

Moment generating property (proof?)

\[ \nabla_\eta A(\eta) = \nabla_\eta \log Z(\eta) = \cdots = \mathbb{E}_{p(x|\eta)}[T(x)] \]

\[ \nabla^2_\eta A(\eta) = \cdots = \text{Var}[T(x)] \]
Moment estimation

- We can easily compute moments of any exponential family distribution by taking the derivatives of the log normalizer $A(\eta)$.
- The $q^{\text{th}}$ derivative gives the $q^{\text{th}}$ centered moment.

\[
\nabla_\eta A(\eta) = \text{mean}
\]
\[
\nabla_\eta^2 A(\eta) = \text{variance}
\]
\[
\vdots
\]
Moment vs canonical parameters

- The moment parameter $\mu$ can be derived from the natural (canonical) parameter

$$\nabla_\eta A(\eta) = \mathbb{E}_{p(x|\eta)}[T(x)] \triangleq \mu$$

- $A(\eta)$ is convex since

$$\nabla^2_\eta A(\eta) = \text{Var}[T(x)] > 0$$

- Hence we can invert the relationship and infer the canonical parameter from the moment parameter (1-to-1):

$$\eta \triangleq \psi(\mu)$$

- A distribution in the exponential family can be parameterized not only by $\eta$ — the canonical parameterization, but also by $\mu$ — the moment parameterization.
IID Sampling for Exponential Family

- For exponential family distribution, we can obtain the sufficient statistics by inspection once represented in the standard form:

\[ p(x | \eta) = h(x) \exp(\eta^\top T(x) - A(\eta)) \]

- Sufficient statistics:
\[ T(x) \]

- For IID sampling, the joint distribution is also an exponential family:

\[ p(D | \eta) = \prod h(x_i) \exp(\eta^\top T(x_i) - A(\eta)) \]

\[ = \left( \prod h(x_i) \right) \exp \left( \eta^\top \sum_i T(x_i) - NA(\eta) \right) \]

- Sufficient statistics:
\[ \sum_i T(x_i) \]
MLE for Exponential Family

For iid data, the log-likelihood is

$$\mathcal{L}(\eta; D) = \sum_n \log h(x_n) + \left( \eta^\top \sum_n T(x_n) \right) - N A(\eta)$$

Take derivatives and set to zero:

$$\nabla_\eta \mathcal{L}(\eta; D) = \sum_n T(x_n) - N \nabla_\eta A(\eta) = 0$$

$$\nabla_\eta A(\eta) = \frac{1}{N} \sum_n T(x_n)$$

$$\hat{\mu}_{MLE} = \frac{1}{N} \sum_n T(x_n) \quad \text{Only involve sufficient statistics!}$$

This amounts to moment matching.

We can infer the canonical parameters using

$$\hat{\eta}_{MLE} = \psi(\hat{\mu}_{MLE})$$
Examples

**Gaussian:** \[ \eta = \begin{bmatrix} \Sigma^{-1} \mu; \ - \frac{1}{2} \text{vec}(\Sigma^{-1}) \end{bmatrix} \]

\[ T(x) = [x; \text{vec}(xx^\top)] \]

\[ A(\eta) = \frac{1}{2} \mu^\top \Sigma^{-1} \mu + \log |\Sigma| \]

\[ h(x) = (2\pi)^{-d/2} \]

\[ \hat{\mu}_{MLE} = \frac{1}{N} \sum_n T_1(x_n) = \frac{1}{N} \sum_n x_n \]

**Multinomial:**

\[ \eta = [\ln(\pi_i/\pi_d); 0] \]

\[ T(x) = x \]

\[ A(\eta) = -\ln \left(1 - \sum_{i=1}^{d-1} \pi_i \right) \]

\[ h(x) = 1 \]

\[ \hat{\mu}_{MLE} = \frac{1}{N} \sum_n x_n \]

**Poisson:** \[ \eta = \log \lambda \]

\[ T(x) = x \]

\[ A(\eta) = \lambda = e^\eta \]

\[ h(x) = \frac{1}{x!} \]

\[ \hat{\mu}_{MLE} = \frac{1}{N} \sum_n x_n \]
Generalized Linear Models (GLIMs)

The graphical model
- Linear regression
- Discriminative linear classification
- Commonality:
  - What is $p()$? the cond. dist. of $Y$.
  - What is $f()$? the response function.

GLIM
- The observed input $X$ is assumed to enter into the model via a linear combination of its elements $\xi = \theta^T X$
- The conditional mean $\mu$ is represented as a function $f(\xi)$ of $\xi$, where $f$ is known as the response function
- The observed output $Y$ is assumed to be characterized by an exponential family distribution with conditional mean $\mu$. 
GLIM, cont.

The choice of exp family is constrained by the nature of the data $Y$

- Example: $y$ is a continuous vector $\rightarrow$ multivariate Gaussian
  $y$ is a class label $\rightarrow$ Bernoulli or multinomial

The choice of the response function

- Following some mild constrains, e.g., $[0,1]$. Positivity …
- Canonical response function:
  - In this case $\theta^T x$ directly corresponds to canonical parameter $\eta$.
  - $f = \psi^{-1}(\cdot)$
MLE for GLIMs

- Log-likelihood

\[ \mathcal{L}(\theta; D) = \sum_n \log h(y_n) + \sum_n (\eta_n y_n - A(\eta_n)) \]

where \( \eta_n = \psi(\mu_n) \), \( \mu_n = f(\xi_n) \) and \( \xi_n = \theta^\top x_n \)

- Derivative of Log-likelihood

\[ \nabla_\theta \mathcal{L} = \sum_n \left( y_n \nabla_\theta \eta_n - \frac{dA(\eta_n)}{d\eta_n} \nabla_\eta \eta_n \right) \]

\[ = \sum_n (y_n - \mu_n) \nabla_\theta \eta_n \]

This is a fixed point function because \( \mu \) is a function of \( \theta \)
MLE for GLIMs with canonical response

- Log-likelihood
  \[
  \mathcal{L}(\theta; D) = \sum_n \log h(y_n) + \sum_n (\theta^\top x_n y_n - A(\eta_n))
  \]

- Derivative of Log-likelihood
  \[
  \nabla_\theta \mathcal{L} = \sum_n \left( x_n y_n - \frac{dA(\eta_n)}{d\eta_n} \nabla_\theta \eta_n \right)
  = \sum_n (y_n - \mu_n)x_n
  = X(y - \mu)
  \]

  This is a fixed point function because \( \mu \) is a function of \( \theta \)

- Online learning for canonical GLIMs
  - Stochastic gradient ascent = least mean squares (LMS) algorithm:
    \[
    \theta_{t+1} = \theta_t + \rho(y_n - \mu^t_n)x_n
    \]
    where \( \mu^t_n = f(\theta_t^\top x_n) \) and \( \rho \) is a step size
MLE for GLIMs with canonical response

- Log-likelihood

\[ \mathcal{L}(\theta; D) = \sum_n \log h(y_n) + \sum_n \left( \theta^\top x_n y_n - A(\eta_n) \right) \]

- Derivative of Log-likelihood

\[ \nabla_\theta \mathcal{L} = \sum_n \left( x_n y_n - \frac{dA(\eta_n)}{d\eta_n} \nabla_\eta \eta_n \right) \]

\[ = \sum_n (y_n - \mu_n) x_n \]

\[ = X(y - \mu) \]

This is a fixed point function because \( \mu \) is a function of \( \theta \)

- Batch learning applies

  - E.g., the Newton’s method leads to an Iteratively Reweighted Least Square (IRLS) algorithm
What you need to know

Exponential family distribution
Moment estimation
Generalized linear models
Parameter estimation of GLIMs