Logistic Regression

COMP 135 Intro to Machine Learning
Liping Liu

slides modified from Roni Khardon’s with permission

Predictive prob from linear functions

• Recall: directly fit $p(y|x)$ for prediction
• Linear function with a link function $g$
  \[ \eta = w^T x + w_0, \]
  \[ p(y = 1|x) = \mu = g^{-1}(\eta) \]
  - Which function $g^{-1}$ to use?
  - How to learn values of $w$ and $w_0$?

Link function $g$ is from the tradition of statistics.
We always directly specify $g^{-1}(\cdot)$

Link function

• Choices of link functions:
  - Logistic function
    \[ g^{-1}(\eta) = \frac{1}{1 + \exp(-\eta)} = \frac{\exp(\eta)}{1 + \exp(\eta)} \]

Likelihood of training instances

• Likelihood of a single instance
  \[ \mu = p(y = 1|x; w, w_0) = g^{-1}(x; w, w_0) \]
  \[ \log p(y|x; w, w_0) = \begin{cases} 
  \log \mu, & \text{if } y = 1 \\
  \log (1 - \mu), & \text{if } y = 0 
\end{cases} \]
  - Write it together as
    \[ \log p(y|x; w, w_0) = y \log \mu + (1 - y) \log (1 - \mu) \]
    Log likelihood of Bernoulli random variable

Predictive prob from linear functions

• Recall: directly fit $p(y|x)$ for prediction
• Linear function with a link function $g$
  \[ \eta = w^T x + w_0, \]
  \[ p(y = 1|x) = \mu = g^{-1}(\eta) \]
  - Which function $g^{-1}$ to use?
  - How to learn values of $w$ and $w_0$?

Link function $g$ is from the tradition of statistics.
We always directly specify $g^{-1}(\cdot)$

Link function

• Two considerations
  - Requirement of domain and range
    \[ g^{-1}: \mathbb{R} \to [0, 1] \]
  - Smooth: small changes of $x$ do not cause big changes of $y$
**Principle of MLE**

- Maximize the likelihood of the data to estimate the parameters
  - We have data \((y_i, x_i), i = 1 \ldots N\)
  - We have a model \(p(y|x; \theta)\)
  - We maximize the data likelihood to estimate \(\theta\) by
    \[
    \max_{\theta} \sum_{i} p(y_i \mid x_i; \theta)
    \]
  - We predict by \(p(y_{\text{new}} \mid x_{\text{new}}; \theta)\)

*Again assume instances are independent. Why?*

**Learn the logistic regression**

- MLE estimation
  \[
  (w, w_0) = \arg \max_{w, w_0} \sum_{i} \log p(y_i \mid x_i; w, w_0)
  \]
- The learned model is \((w, w_0)\)

Will talk about solving the optimization problem later.

**Prediction by logistic regression**

- Prediction
  - Decision rule
    \[
    \hat{y} = \begin{cases} 
    p(y = 1 \mid x) > 0.5 & \text{if } \hat{y} = w^T x + w_0 > 0 
    \end{cases}
    \]

**Geometric understanding**

- Linear decision boundary

**Linear decision rule**

- The classification rule \(\hat{y} = [w^T x + w_0 > 0]\) is powerful
  - Express \(y = (x_1 \text{ and } x_2 \text{ and } x_3)\),
    \(y = [x_1 + x_2 + x_3 > 3]\)
  - Express \(y = (x_1 \text{ or } x_2 \text{ or } x_3)\):
    \(y = [x_1 + x_2 + x_3 > 1]\)

- The classification rule \(\hat{y} = [w^T x + w_0 > 0]\) has weakness
  - It cannot represent XOR

Image from [RN] AIMA
Linear decision rule

- Relation with the data dimensionality
  - In 2d sample space, can you find 3 different instances of two classes that cannot be linearly separated?
  - In 3d sample space, can you find 4 different instances of two classes that cannot be linearly separated?
  - In general, instances in high dimensional space is easier for linear decision rule

Overfitting

- Common data problem
  - Label is random for the same $x$
  - Label are not “smooth” with large variations within a small area of $x$
  - Not enough data to reveal such variations

Overfitting

- Model learns some patterns that cannot generalize
  - Model works hard in learning to fix every training error
  - Performs well on training set but badly on test set
  - Analogy in real world: memorize all answers of exercise problems, but cannot solve problems in tests
- Solution: use simpler/more rigid model

Regularization

- Linear model is more likely to overfit high dimensional data
- Penalize model complexity
  - Minimize a regularizers $R(w, w_0)$ together with the maximization of the likelihood
  - Examples of regularizers:
    \begin{align*}
    R(w, w_0) &= w^T w = ||w||^2_2 \\
    R(w, w_0) &= \sum_i |w_i| = ||w||_1
    \end{align*}

Objective with regularization

- Add regularizer to the optimization problem
  - The parameter $\lambda$ tradeoff the two objectives
  - The number $\frac{1}{2}$ is to cancel the number 2 in derivative of $||w||^2_2$
  \begin{align*}
  \max_{w, w_0} & \sum_i \log p(y_i | x_i; w, w_0) - \frac{\lambda}{2} ||w||^2_2 \\
  \min_{w, w_0} & -\sum_i \log p(y_i | x_i; w, w_0) + \frac{\lambda}{2} ||w||^2_2
  \end{align*}

More explanation of regularization

- More explanations
  - A large $\lambda$ drives $w$ to zero
  - With large $\lambda$ the model is rigid
    - negative log-likelihood plays less important role in training
  - $\lambda = 0$ recovers the basic model
    - Model is more susceptible to variances in training sets

Will focus on 2-norm in this course.
See more norms in [ICML]
Learning the model

• Solve the optimization problem

\[
\min_{w,w_0} L(w, w_0) = -\sum_i \log p(y_i | x_i; w, w_0) + \frac{\lambda}{2} \|w\|^2
\]

• Recall: need to find a \(w, w_0\) such that

\[
\frac{\partial L(w, w_0)}{\partial w} = 0, \quad \frac{\partial L(w, w_0)}{\partial w_0} = 0
\]

• No closed-form solution, need iterative solution

Mini-tutorial of optimization

• You have a function \(f(x)\), need to find \(\min_x f(x)\)

• Gradient descent

- Start at an initial point \(x_0\)
- Iteratively do: \(x_{t+1} = x_t - \alpha_t f(x_t)\)
- Until \(x_t\) or \(f(x_t)\) does not change much

Gradient descent - illustration

• One dimension

Optimization

• How do I set step size?
  - A small number in general (many studies on this problem)
  - E.g., \(\alpha_t = \frac{\alpha_0}{t}\), with \(\alpha_0\) being a constant

• Will I reach the optimal point?
  - Gradient descent methods are guaranteed to get global minimum for convex function
Back to our problem

- Useful fact about sigmoid
  \[ L(w, w_0) = -\sum_i \log p(y_i | x_i; w, w_0) + \frac{\lambda}{2} \|w\|_2^2 \]
  - Our problem is convex!
  - Need to calculate the gradient (work on white board)

Final algorithm

Start with \( w^0, w_0^0 = 0 \)
For \( t = 0, \ldots, (T-1) \)
\[
\begin{align*}
  w^{t+1} &= w^t - \frac{\alpha}{t} \nabla L(w^t, w_0^t) - \lambda w^t \\
  w_0^{t+1} &= w_0^t - \frac{\alpha}{t} \nabla L(w^t, w_0^t) - \lambda w_0^t
\end{align*}
\]
if \( L(w^{t+1}, w_0^{t+1}) - L(w^t, w_0^t) < \delta \)
break
Return \( w^T, w_0^T \)

Discriminative and generative

- Generative model
  - Model data likelihood \( p(y, x; \theta) \)
  - Example: Naive Bayes
- Discriminative model
  - Model \( p(y|x; \theta') \)
  - Example: logistic regression
  - Most classifiers are discriminative

Discriminative and generative

- Discriminative model generally performs better
  - Intuitive understanding: generative models implicitly fit \( p(x) \), which is unnecessary, since \( x \) will be known at testing stage
  \[
  \log p(x, y; \theta) = \log p(x; \theta) + \log p(y|x; \theta)
  \]

Recap: Logistic regression model

- Expressive for high dimensional data
- Has limitations, rigid decision boundary
- Easy geometric understanding
- Discriminative model
- Need iterative optimization for model fitting, but still relatively easy problem