Probabilistic Linear Classification: Logistic Regression

Piyush Rai
IIT Kanpur

Probabilistic Machine Learning (CS772A)

Jan 18, 2016
Probabilistic Classification

• Given: \( N \) labeled training examples \( \{x_n, y_n\}_{n=1}^{N}, \ x_n \in \mathbb{R}^D, \ y_n \in \{0, 1\} \)
• \( X \): \( N \times D \) feature matrix, \( y \): \( N \times 1 \) label vector
• \( y_n = 1 \): positive example, \( y_n = 0 \): negative example
• Goal: Learn a classifier that predicts the binary label \( y_* \) for a new input \( x_* \)
• Want a probabilistic model to be able to also predict the label probabilities
  \[
p(y_n = 1 | x_n, w) = \mu_n \\
p(y_n = 0 | x_n, w) = 1 - \mu_n
\]
• \( \mu_n \in (0, 1) \) is the probability of \( y_n \) being 1
• Note: Features \( x_n \) assumed fixed (given). Only labels \( y_n \) being modeled
• \( w \) is the model parameter (to be learned)
• How do we define \( \mu_n \) (want it to be a function of \( w \) and input \( x_n \))?
Logistic Regression

- Logistic regression defines \( \mu \) using the sigmoid function

\[
\mu = \sigma (w^\top x) = \frac{1}{1 + \exp(-w^\top x)} = \frac{\exp(w^\top x)}{1 + \exp(w^\top x)}
\]

- Sigmoid computes a real-valued “score” \((w^\top x)\) for input \(x\) and “squashes” it between \((0,1)\) to turn this score into a probability (of \(x\)’s label being 1)

- Thus we have

\[
p(y = 1|x, w) = \mu = \sigma (w^\top x) = \frac{1}{1 + \exp(-w^\top x)} = \frac{\exp(w^\top x)}{1 + \exp(w^\top x)}
\]

\[
p(y = 0|x, w) = 1 - \mu = 1 - \sigma (w^\top x) = \frac{1}{1 + \exp(w^\top x)}
\]

- **Note:** If we assume \(y \in \{-1, +1\}\) instead of \(y \in \{0, 1\}\) then \(p(y|x, w) = \frac{1}{1+\exp(-yw^\top x)}\)
What’s the underlying decision rule in Logistic Regression?

At the decision boundary, both classes are equiprobable. Thus:

\[ p(y = 1 | x, w) = p(y = 0 | x, w) \]
\[ \frac{\exp(w^\top x)}{1 + \exp(w^\top x)} = \frac{1}{1 + \exp(w^\top x)} \]
\[ \exp(w^\top x) = 1 \]
\[ w^\top x = 0 \]

Thus the decision boundary of LR is nothing but a linear hyperplane, just like Perceptron, Support Vector Machine (SVM), etc.

Therefore \( y = 1 \) if \( w^\top x \geq 0 \), otherwise \( y = 0 \)
Interpreting the probabilities..

- Recall that
  \[ p(y = 1|x, w) = \mu = \frac{1}{1 + \exp(-w^\top x)} \]

- Note that the “score” \( w^\top x \) is also a measure of distance of \( x \) from the hyperplane (score is positive for pos. examples, negative for neg. examples)

- High positive score \( w^\top x \): High probability of label 1
- High negative score \( w^\top x \): Low prob. of label 1 (high prob. of label 0)
Logistic Regression: Parameter Estimation

- Recall, each label \( y_n \) is binary with prob. \( \mu_n \). Assume Bernoulli likelihood:
  \[
p(y \mid X, w) = \prod_{n=1}^{N} p(y_n \mid x_n, w) = \prod_{n=1}^{N} \mu_n^{y_n}(1 - \mu_n)^{1-y_n}
\]

  where \( \mu_n = \frac{\exp(w^\top x_n)}{1 + \exp(w^\top x_n)} \)

- Negative log-likelihood
  \[
  \text{NLL}(w) = - \log p(Y \mid X, w) = - \sum_{n=1}^{N} (y_n \log \mu_n + (1 - y_n) \log(1 - \mu_n))
  \]

- Plugging in \( \mu_n = \frac{\exp(w^\top x_n)}{1 + \exp(w^\top x_n)} \) and chugging, we get (verify yourself)
  \[
  \text{NLL}(w) = - \sum_{n=1}^{N} (y_n w^\top x_n - \log(1 + \exp(w^\top x_n)))
  \]

- To do MLE for \( w \), we’ll minimize negative log-likelihood \( \text{NLL}(w) \) w.r.t. \( w \)

- **Important note:** \( \text{NLL}(w) \) is convex in \( w \), so global minima can be found
MLE Estimation for Logistic Regression

- We have $\text{NLL}(\mathbf{w}) = - \sum_{n=1}^{N} (y_n \mathbf{w}^\top \mathbf{x}_n - \log(1 + \exp(\mathbf{w}^\top \mathbf{x}_n)))$

- Taking the derivative of $\text{NLL}(\mathbf{w})$ w.r.t. $\mathbf{w}$

$$
\frac{\partial \text{NLL}(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} [- \sum_{n=1}^{N} (y_n \mathbf{w}^\top \mathbf{x}_n - \log(1 + \exp(\mathbf{w}^\top \mathbf{x}_n)))] \\
= - \sum_{n=1}^{N} \left( y_n \mathbf{x}_n - \frac{\exp(\mathbf{w}^\top \mathbf{x}_n)}{(1 + \exp(\mathbf{w}^\top \mathbf{x}_n))} \mathbf{x}_n \right)
$$

- Can’t get a closed form estimate for $\mathbf{w}$ by setting the derivative to zero

- One solution: Iterative minimization via gradient descent. Gradient is:

$$
\mathbf{g} = \frac{\partial \text{NLL}(\mathbf{w})}{\partial \mathbf{w}} = - \sum_{n=1}^{N} (y_n - \mu_n) \mathbf{x}_n = \mathbf{X}^\top (\mathbf{\mu} - \mathbf{y})
$$

- Intuitively, a large error on $\mathbf{x}_n \Rightarrow (y_n - \mu_n)$ will be large $\Rightarrow$ large contribution (positive/negative) of $\mathbf{x}_n$ to the gradient
MLE Estimation via Gradient Descent

- Gradient descent (GD) or steepest descent

\[ w_{t+1} = w_t - \eta_t g_t \]

where \( \eta_t \) is the learning rate (or step size), and \( g_t \) is gradient at step \( t \)

- GD can converge slowly and is also sensitive to the step size

- Several ways to remedy this\(^1\). E.g.,
  - Choose the optimal step size \( \eta_t \) by line-search
  - Add a momentum term to the updates

\[ w_{t+1} = w_t - \eta_t g_t + \alpha_t (w_t - w_{t-1}) \]

  - Use methods such as conjugate gradient
  - Use second-order methods (e.g., Newton’s method) to exploit the curvature of the objective function \( \text{NLL}(w) \): Require the Hessian matrix

\(^1\)Also see: “A comparison of numerical optimizers for logistic regression” by Tom Minka
MLE Estimation via Newton’s Method

- Update via Newton’s method:
  \[ w_{t+1} = w_t - H_t^{-1} g_t \]
  
  where \( H_t \) is the Hessian matrix at step \( t \)

- Hessian: double derivative of the objective function (NLL(\( w \)) in this case)
  \[ H = \frac{\partial^2 \text{NLL}(w)}{\partial w \partial w^\top} = \frac{\partial g^\top}{\partial w} \]

- Recall that the gradient is: \( g = -\sum_{n=1}^{N} (y_n - \mu_n)x_n = X^\top(\mu - y) \)

- Thus \( H = \frac{\partial g^\top}{\partial w} = -\frac{\partial}{\partial w} \sum_{n=1}^{N} (y_n - \mu_n)x_n^\top = \sum_{n=1}^{N} \frac{\partial \mu_n}{\partial w} x_n^\top \)

- Using the fact that \( \frac{\partial \mu_n}{\partial w} = \frac{\partial}{\partial w} \left( \frac{\exp(w^\top x_n)}{1+\exp(w^\top x_n)} \right) = \mu_n(1 - \mu_n)x_n \), we have
  \[ H = \sum_{n=1}^{N} \mu_n(1 - \mu_n)x_n x_n^\top = X^\top S X \]

  where \( S \) is a diagonal matrix with its \( n^{th} \) diagonal element = \( \mu_n(1 - \mu_n) \)
MLE Estimation via Newton’s Method

- Update via Newton’s method:

\[
\begin{align*}
    w_{t+1} &= w_t - H_t^{-1} g_t \\
    &= w_t - (X^T S_t X)^{-1} X^T (\mu_t - y) \\
    &= w_t + (X^T S_t X)^{-1} X^T (y - \mu_t) \\
    &= (X^T S_t X)^{-1} [(X^T S_t X)w_t + X^T (y - \mu_t)] \\
    &= (X^T S_t X)^{-1} X^T [S_t Xw_t + y - \mu_t] \\
    &= (X^T S_t X)^{-1} X^T S_t [Xw_t + S^{-1} (y - \mu_t)] \\
    &= (X^T S_t X)^{-1} X^T S_t \hat{y}_t
    
\end{align*}
\]

- Interpreting the solution found by Newton’s method:
  - It basically solves an **Iteratively Reweighted Least Squares (IRLS)** problem
    \[
    \arg \min_w \sum_{n=1}^N S_{tn}(\hat{y}_{tn} - w^T x_n)^2
    \]
  - Note that the (redefined) response vector \( \hat{y}_t \) changes in each iteration
  - Each term in the objective has weight \( S_{tn} \) (changes in each iteration)
  - The weight \( S_{tn} \) is the \( n^{th} \) diagonal element of \( S_t \)
MLE estimate of $w$ can lead to overfitting. Solution: use a prior on $w$

Just like the linear regression case, let’s put a Gaussian prior on $w$

$$p(w) = \mathcal{N}(0, \lambda^{-1}I_D) \propto \exp(-\frac{\lambda}{2}w^\top w)$$

MAP objective: MLE objective + log $p(w)$

Leads to the objective (negative of log posterior, ignoring constants):

$$\text{NLL}(w) + \frac{\lambda}{2}w^\top w$$

Estimation of $w$ proceeds the same way as MLE except that now we have

Gradient: $g = X^\top (\mu - y) + \lambda w$

Hessian: $H = X^\top SX + \lambda I_D$

Can now apply iterative optimization (gradient des., Newton’s method, etc.)

**Note:** MAP estimation for log. reg. is equivalent to regularized log. reg.
What about the **full posterior** on $w$?

Not as easy to estimate as in the linear regression case!

Reason: likelihood (logistic-Bernoulli) and prior (Gaussian) not conjugate

Need to *approximate* the posterior in this case

A crude approximation: **Laplace approximation**: Approximate a posterior by a **Gaussian** with mean = MAP estimate and covariance = inverse hessian

\[ p(w|X, y) = \mathcal{N}(w_{\text{MAP}}, H^{-1}) \]

Will see other ways of approximating the posterior later during the semester
Derivation of the Laplace Approximation

- The posterior \( p(w | X, y) = \frac{p(y | X, w)p(w)}{p(y | X)} \). Let’s approximate it as

\[
p(w | X, y) = \frac{\exp(-E(w))}{Z}
\]

where \( E(w) = -\log p(y | X, w)p(w) \) and \( Z \) is the normalizer.

- Expand \( E(w) \) around its minima \( (w_* = w_{MAP}) \) using 2\(^{nd}\) order Taylor exp.

\[
E(w) \approx E(w_*) + (w - w_*)^\top g + \frac{1}{2} (w - w_*)^\top H(w - w_*)
\]

\[
= E(w_*) + \frac{1}{2} (w - w_*)^\top H(w - w_*) \quad \text{(because } g = 0 \text{ at } w_*) \]

- Thus the posterior

\[
p(w | X, y) \approx \frac{\exp(-E(w_*)) \exp(-\frac{1}{2} (w - w_*)^\top H(w - w_*)))}{Z}
\]

- Using \( \int_w p(w | X, y)dw = 1 \), we get \( Z = \exp(-E(w_*))(2\pi)^{D/2}|H|^{-1/2} \). Thus

\[
p(w | X, y) = \mathcal{N}(w_*, H^{-1})
\]
Logistic reg. can be extended to handle \( K > 2 \) classes.

In this case, \( y_n \in \{0, 1, 2, \ldots, K - 1\} \) and label probabilities are defined as

\[
p(y_n = k|x_n, W) = \frac{\exp(w_k^\top x_n)}{\sum_{\ell=1}^{K} \exp(w_\ell^\top x_n)} = \mu_{nk}
\]

\( \mu_{nk} \): probability that example \( n \) belongs to class \( k \). Also, \( \sum_{\ell=1}^{K} \mu_{n\ell} = 1 \)

\( W = [w_1 \ w_2 \ \ldots \ w_K] \) is \( D \times K \) weight matrix (column \( k \) for class \( k \))

Likelihood for the multinomial (or multinoulli) logistic regression model

\[
p(y|X, W) = \prod_{n=1}^{N} \prod_{\ell=1}^{K} \mu_{n\ell}^{y_{n\ell}}
\]

where \( y_{n\ell} = 1 \) if true class of example \( n \) is \( \ell \) and \( y_{n\ell'} = 0 \) for all other \( \ell' \neq \ell \)

Can do MLE/MAP/fully Bayesian estimation for \( W \) similar to the binary case

**Decision rule:** \( y_\star = \arg \max_{\ell=1,\ldots, K} w_\ell^\top x_\star \), i.e., predict the class whose weight vector gives the largest score (or, equivalently, the largest probability)
Next class:
Generalized Linear Models