Probabilistic Linear Classification: Logistic Regression

Piyush Rai
IIT Kanpur

Probabilistic Machine Learning (CS772A)

Jan 18, 2016

Probabilistic Classification

- Given: N labeled training examples $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$, $\boldsymbol{x}_n \in \mathbb{R}^D$, $y_n \in \{0, 1\}$
- $\mathbf{X}$: $N \times D$ feature matrix, $\boldsymbol{y}$: $N \times 1$ 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 $\boldsymbol{x}_*$
- Want a probabilistic model to be able to also predict the *label probabilities*

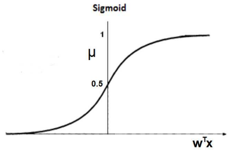
$$
\begin{aligned}
p(y_n = 1 | \boldsymbol{x}_n, \boldsymbol{w}) &= \mu_n \\
p(y_n = 0 | \boldsymbol{x}_n, \boldsymbol{w}) &= 1 - \mu_n
\end{aligned}
$$

- $\mu_n \in (0, 1)$ is the probability of y_n being 1
- Note: Features $\boldsymbol{x}_n$ assumed fixed (given). Only labels y_n being modeled
- $\boldsymbol{w}$ is the model parameter (to be learned)
- How do we define μ_n (want it to be a function of $\boldsymbol{w}$ and input $\boldsymbol{x}_n$)?

Logistic Regression

- Logistic regression defines μ using the sigmoid function

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



- Sigmoid computes a real-valued "score" ($\boldsymbol{w}^\top \boldsymbol{x}$) for input $\boldsymbol{x}$ and "squashes" it between (0,1) to turn this score into a probability (of $\boldsymbol{x}$'s label being 1)
- Thus we have

$$\begin{aligned} p(y=1|\boldsymbol{x},\boldsymbol{w}) &= \mu = \sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{w}^\top \boldsymbol{x})} = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x})}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x})} \\ p(y=0|\boldsymbol{x},\boldsymbol{w}) &= 1-\mu = 1-\sigma(\boldsymbol{w}^\top \boldsymbol{x}) = \frac{1}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x})} \end{aligned}$$

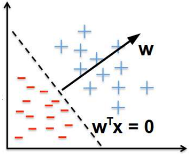
- **Note:** If we assume $y \in \{-1,+1\}$ instead of $y \in \{0,1\}$ then $p(y|\boldsymbol{x},\boldsymbol{w}) = \frac{1}{1+\exp(-y\boldsymbol{w}^\top \boldsymbol{x})}$

Logistic Regression: A Closer Look..

- What's the underlying decision rule in Logistic Regression?
- At the decision boundary, both classes are equiprobable. Thus:

$$
\begin{aligned}
p(y = 1|\boldsymbol{x}, \boldsymbol{w}) &= p(y = 0|\boldsymbol{x}, \boldsymbol{w}) \\
\frac{\exp(\boldsymbol{w}^\top \boldsymbol{x})}{1 + \exp(\boldsymbol{w}^\top \boldsymbol{x})} &= \frac{1}{1 + \exp(\boldsymbol{w}^\top \boldsymbol{x})} \\
\exp(\boldsymbol{w}^\top \boldsymbol{x}) &= 1 \\
\boldsymbol{w}^\top \boldsymbol{x} &= 0
\end{aligned}
$$

- Thus the decision boundary of LR is nothing but a linear hyperplane, just like Perceptron, Support Vector Machine (SVM), etc.
- Therefore $y = 1$ if $\boldsymbol{w}^\top \boldsymbol{x} \geq 0$, otherwise $y = 0$

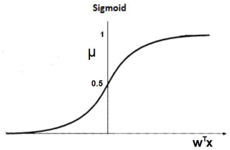


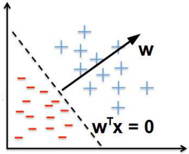
Interpreting the probabilities..

- Recall that

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

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





- High positive score $\boldsymbol{w}^\top \boldsymbol{x}$: High probability of label 1
- High negative score $\boldsymbol{w}^\top \boldsymbol{x}$: Low prob. of label 1 (high prob. of label 0)

Logistic Regression: Parameter Estimation

- Recall, each label y_n is binary with prob. μ_n. Assume Bernoulli likelihood:

$$p(\boldsymbol{y}|\mathbf{X}, \boldsymbol{w}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \prod_{n=1}^{N} \mu_n^{y_n}(1-\mu_n)^{1-y_n}$$

where $\mu_n = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$

- Negative log-likelihood

$$\text{NLL}(\boldsymbol{w}) = -\log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{w}) = -\sum_{n=1}^{N}(y_n \log \mu_n + (1-y_n)\log(1-\mu_n))$$

- Plugging in $\mu_n = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}$ and chugging, we get (verify yourself)

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

- To do MLE for $\boldsymbol{w}$, we'll minimize negative log-likelihood $\text{NLL}(\boldsymbol{w})$ w.r.t. $\boldsymbol{w}$
- **Important note:** $\text{NLL}(\boldsymbol{w})$ is convex in $\boldsymbol{w}$, so global minima can be found

MLE Estimation for Logistic Regression

- We have $\text{NLL}(\boldsymbol{w}) = -\sum_{n=1}^{N}(y_n \boldsymbol{w}^\top \boldsymbol{x}_n - \log(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}_n)))$
- Taking the derivative of $\text{NLL}(\boldsymbol{w})$ w.r.t. $\boldsymbol{w}$

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

- Can't get a closed form estimate for $\boldsymbol{w}$ by setting the derivative to zero
- One solution: Iterative minimization via gradient descent. Gradient is:

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

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

MLE Estimation via Gradient Descent

- Gradient descent (GD) or steepest descent

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \mathbf{g}_t$$

  where η_t is the learning rate (or step size), and $\mathbf{g}_t$ 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 η_t by line-search
  - Add a momentum term to the updates

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \mathbf{g}_t + \alpha_t(\boldsymbol{w}_t - \boldsymbol{w}_{t-1})$$

  - Use methods such as conjugate gradient
  - Use second-order methods (e.g., **Newton's method**) to exploit the curvature of the objective function NLL($\boldsymbol{w}$): 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:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \mathbf{H}_t^{-1}\mathbf{g}_t$$

  where $\mathbf{H}_t$ is the Hessian matrix at step t

- Hessian: double derivative of the objective function (NLL($\boldsymbol{w}$) in this case)

$$\mathbf{H} = \frac{\partial^2 \text{NLL}(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}^\top} = \frac{\partial \mathbf{g}^\top}{\partial \boldsymbol{w}}$$

- Recall that the gradient is: $\mathbf{g} = -\sum_{n=1}^N (y_n - \mu_n)\boldsymbol{x}_n = \mathbf{X}^\top(\boldsymbol{\mu} - \boldsymbol{y})$
- Thus $\mathbf{H} = \frac{\partial \mathbf{g}^\top}{\partial \boldsymbol{w}} = -\frac{\partial}{\partial \boldsymbol{w}} \sum_{n=1}^N (y_n - \mu_n)\boldsymbol{x}_n^\top = \sum_{n=1}^N \frac{\partial \mu_n}{\partial \boldsymbol{w}} \boldsymbol{x}_n^\top$
- Using the fact that $\frac{\partial \mu_n}{\partial \boldsymbol{w}} = \frac{\partial}{\partial \boldsymbol{w}} \left(\frac{\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x}_n)} \right) = \mu_n(1-\mu_n)\boldsymbol{x}_n$, we have

$$\boxed{\mathbf{H} = \sum_{n=1}^N \mu_n(1-\mu_n)\boldsymbol{x}_n\boldsymbol{x}_n^\top = \mathbf{X}^\top \mathbf{S} \mathbf{X}}$$

  where $\mathbf{S}$ 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:

$$\begin{aligned}
\boldsymbol{w}_{t+1} &= \boldsymbol{w}_t - \mathbf{H}_t^{-1}\mathbf{g}_t \\
&= \boldsymbol{w}_t - (\mathbf{X}^\top \mathbf{S}_t \mathbf{X})^{-1}\mathbf{X}^\top(\boldsymbol{\mu}_t - \boldsymbol{y}) \\
&= \boldsymbol{w}_t + (\mathbf{X}^\top \mathbf{S}_t \mathbf{X})^{-1}\mathbf{X}^\top(\boldsymbol{y} - \boldsymbol{\mu}_t) \\
&= (\mathbf{X}^\top \mathbf{S}_t \mathbf{X})^{-1}[(\mathbf{X}^\top \mathbf{S}_t \mathbf{X})\boldsymbol{w}_t + \mathbf{X}^\top(\boldsymbol{y} - \boldsymbol{\mu}_t)] \\
&= (\mathbf{X}^\top \mathbf{S}_t \mathbf{X})^{-1}\mathbf{X}^\top[\mathbf{S}_t \mathbf{X}\boldsymbol{w}_t + \boldsymbol{y} - \boldsymbol{\mu}_t] \\
&= (\mathbf{X}^\top \mathbf{S}_t \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{S}_t[\mathbf{X}\boldsymbol{w}_t + \mathbf{S}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_t)] \\
&= (\mathbf{X}^\top \mathbf{S}_t \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{S}_t \hat{\boldsymbol{y}}_t
\end{aligned}$$

- Interpreting the solution found by Newton's method:
  - It basically solves an Iteratively Reweighted Least Squares (IRLS) problem

$$\arg\min_{\boldsymbol{w}} \sum_{n=1}^{N} S_{tn}(\hat{y}_{tn} - \boldsymbol{w}^\top \boldsymbol{x}_n)^2$$

  - Note that the (redefined) response vector $\hat{\boldsymbol{y}}_t$ 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 $\mathbf{S}_t$

MAP Estimation for Logisic Regression

- MLE estimate of $\boldsymbol{w}$ can lead to overfitting. Solution: use a prior on $\boldsymbol{w}$
- Just like the linear regression case, let's put a Gausian prior on $\boldsymbol{w}$

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

- MAP objective: MLE objective + $\log p(\boldsymbol{w})$
- Leads to the objective (negative of log posterior, ignoring constants):

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

- Estimation of $\boldsymbol{w}$ proceeds the same way as MLE excepet that now we have

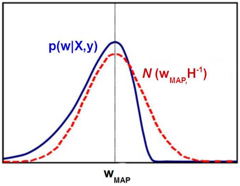
$$\begin{aligned}
\text{Gradient:} \quad \mathbf{g} &= \mathbf{X}^\top(\boldsymbol{\mu} - \boldsymbol{y}) + \lambda \boldsymbol{w} \\
\text{Hessian:} \quad \mathbf{H} &= \mathbf{X}^\top \mathbf{S} \mathbf{X} + \lambda \mathbf{I}_D
\end{aligned}$$

- Can now apply iterative optimization (gradient des., Newton's method, etc.)
- **Note:** MAP estimation for log. reg. is equivalent to regularized log. reg.

Fully Bayesian Estimation for Logistic Regression

- What about the **full posterior** on $\boldsymbol{w}$?
- 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

$$p(\boldsymbol{w}|\mathbf{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{w}_{MAP}, \mathbf{H}^{-1})$$



- Will see other ways of approximating the posterior later during the semester

Derivation of the Laplace Approximation

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

$$p(\boldsymbol{w}|\mathbf{X},\boldsymbol{y}) = \frac{\exp(-E(\boldsymbol{w}))}{Z}$$

  where $E(\boldsymbol{w}) = -\log p(\boldsymbol{y}|\mathbf{X},\boldsymbol{w})p(\boldsymbol{w})$ and Z is the normalizer

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

$$\begin{aligned} E(\boldsymbol{w}) &\approx E(\boldsymbol{w}_*) + (\boldsymbol{w}-\boldsymbol{w}_*)^\top \mathbf{g} + \frac{1}{2}(\boldsymbol{w}-\boldsymbol{w}_*)^\top \mathbf{H}(\boldsymbol{w}-\boldsymbol{w}_*) \\ &= E(\boldsymbol{w}_*) + \frac{1}{2}(\boldsymbol{w}-\boldsymbol{w}_*)^\top \mathbf{H}(\boldsymbol{w}-\boldsymbol{w}_*) \qquad \text{(because } \mathbf{g}=0 \text{ at } \boldsymbol{w}_*\text{)} \end{aligned}$$

- Thus the posterior

$$p(\boldsymbol{w}|\mathbf{X},\boldsymbol{y}) \approx \frac{\exp(-E(\boldsymbol{w}_*))\exp(-\frac{1}{2}(\boldsymbol{w}-\boldsymbol{w}_*)^\top \mathbf{H}(\boldsymbol{w}-\boldsymbol{w}_*)))}{Z}$$

- Using $\int_{\boldsymbol{w}} p(\boldsymbol{w}|\mathbf{X},\boldsymbol{y})d\boldsymbol{w} = 1$, we get $Z = \exp(-E(\boldsymbol{w}_*))(2\pi)^{D/2}|\mathbf{H}|^{-1/2}$. Thus

$$p(\boldsymbol{w}|\mathbf{X},\boldsymbol{y}) = \mathcal{N}(\boldsymbol{w}_*, \mathbf{H}^{-1})$$

Multinomial Logistic Regression

- 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

$$p(y_n = k|\boldsymbol{x}_n, \mathbf{W}) = \frac{\exp(\boldsymbol{w}_k^\top \boldsymbol{x}_n)}{\sum_{\ell=1}^K \exp(\boldsymbol{w}_\ell^\top \boldsymbol{x}_n)} = \mu_{nk}$$

- μ_{nk}: probability that example n belongs to class k. Also, $\sum_{\ell=1}^K \mu_{n\ell} = 1$
- $\mathbf{W} = [\boldsymbol{w}_1 \ \boldsymbol{w}_2 \ \ldots \ \boldsymbol{w}_K]$ is $D \times K$ weight matrix (column k for class k)
- Likelihood for the multinomial (or multinoulli) logistic regression model

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

  where $y_{n\ell} = 1$ if true class of example n is ℓ and $y_{n\ell'} = 0$ for all other $\ell' \neq \ell$
- Can do MLE/MAP/fully Bayesian estimation for $\mathbf{W}$ similar to the binary case
- **Decision rule:** $y_* = \arg\max_{\ell=1,\ldots,K} \boldsymbol{w}_\ell^\top \boldsymbol{x}_*$, i.e., predict the class whose weight vector gives the largest score (or, equivalently, the largest probability)

Next class: Generalized Linear Models