Chapter 8
Logistic Regression

8.1 Representation

Logistic regression can be binomial or multinomial. The **binomial logistic regression** model has the following form

$$p(y|\boldsymbol{x},\boldsymbol{w}) = \text{Ber}(y|\text{sigm}(\boldsymbol{w}^T\boldsymbol{x})) \tag{8.1}$$

where $\boldsymbol{w}$ and $\boldsymbol{x}$ are extended vectors, i.e., $\boldsymbol{w} = (b, w_1, w_2, \cdots, w_D)$, $\boldsymbol{x} = (1, x_1, x_2, \cdots, x_D)$.

8.2 Optimization

8.2.1 MLE

$$\begin{aligned}
\ell(\boldsymbol{w}) &= \log\left\{\prod_{i=1}^{N}[\pi(\boldsymbol{x}_i)]^{y_i}[1-\pi(\boldsymbol{x}_i)]^{1-y_i}\right\} \\
&\quad \text{, where } \pi(\boldsymbol{x}) \triangleq P(y=1|\boldsymbol{x},\boldsymbol{w}) \\
&= \sum_{i=1}^{N}[y_i \log \pi(\boldsymbol{x}_i) + (1-y_i)\log(1-\pi(\boldsymbol{x}_i))] \\
&= \sum_{i=1}^{N}\left[y_i \log\frac{\pi(\boldsymbol{x}_i)}{1-\pi(\boldsymbol{x}_i)} + \log(1-\pi(\boldsymbol{x}_i))\right] \\
&= \sum_{i=1}^{N}[y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i) - \log(1+\exp(\boldsymbol{w}\cdot\boldsymbol{x}_i))] \\
J(\boldsymbol{w}) &\triangleq \text{NLL}(\boldsymbol{w}) = -\ell(\boldsymbol{w}) \\
&= -\sum_{i=1}^{N}[y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i) - \log(1+\exp(\boldsymbol{w}\cdot\boldsymbol{x}_i))]
\end{aligned} \tag{8.2}$$

Equation 8.2 is also called the **cross-entropy** error function (see Equation 2.53).

Unlike linear regression, we can no longer write down the MLE in closed form. Instead, we need to use an optimization algorithm to compute it, see Appendix A. For this, we need to derive the gradient and Hessian.

In the case of logistic regression, one can show that the gradient and Hessian of this are given by the following

$$\boldsymbol{g}(\boldsymbol{w}) = \frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{w}} = \sum_{i=1}^{N}[\pi(\boldsymbol{x}_i) - y_i]\boldsymbol{x}_i = \boldsymbol{X}(\boldsymbol{\pi}-\boldsymbol{y}) \tag{8.3}$$

$$\begin{aligned}
\boldsymbol{H}(\boldsymbol{w}) &= \frac{\mathrm{d}\boldsymbol{g}^T}{\mathrm{d}\boldsymbol{w}} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{w}}(\boldsymbol{\pi}-\boldsymbol{y})^T\boldsymbol{X}^T \\
&= \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{w}}\boldsymbol{\pi}^T\boldsymbol{X}^T \\
&= (\pi(\boldsymbol{x}_i)(1-\pi(\boldsymbol{x}_i))\boldsymbol{x}_i, \cdots,)\boldsymbol{X}^T \\
&= \boldsymbol{X}\boldsymbol{S}\boldsymbol{X}^T, \quad \boldsymbol{S} \triangleq \text{diag}(\pi(\boldsymbol{x}_i)(1-\pi(\boldsymbol{x}_i))\boldsymbol{x}_i)
\end{aligned} \tag{8.4}$$

8.2.1.1 Iteratively reweighted least squares (IRLS)

TODO

8.2.2 MAP

Just as we prefer ridge regression to linear regression, so we should prefer MAP estimation for logistic regression to computing the MLE. ℓ_2 regularization

we can use ℓ_2 regularization, just as we did with ridge regression. We note that the new objective, gradient and Hessian have the following forms:

$$J'(\boldsymbol{w}) \triangleq \text{NLL}(\boldsymbol{w}) + \lambda\boldsymbol{w}^T\boldsymbol{w} \tag{8.5}$$

$$\boldsymbol{g}'(\boldsymbol{w}) = \boldsymbol{g}(\boldsymbol{w}) + \lambda\boldsymbol{w} \tag{8.6}$$

$$\boldsymbol{H}'(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w}) + \lambda\boldsymbol{I} \tag{8.7}$$

It is a simple matter to pass these modified equations into any gradient-based optimizer.

8.3 Multinomial logistic regression

8.3.1 Representation

Multinomial logistic regression model is also called a **maximum entropy classifier**, which has the following form

$$p(y=c|\boldsymbol{x},\boldsymbol{W}) = \frac{\exp(\boldsymbol{w}_c^T\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{w}_c^T\boldsymbol{x})} \tag{8.8}$$

8.3.2 MLE

Let $\boldsymbol{y}_i = (\mathbb{I}(y_i=1),\mathbb{I}(y_i=1),\cdots,\mathbb{I}(y_i=C))$, $\boldsymbol{\mu}_i = (p(y=1|\boldsymbol{x}_i,\boldsymbol{W}), p(y=2|\boldsymbol{x}_i,\boldsymbol{W}),\cdots,p(y=C|\boldsymbol{x}_i,\boldsymbol{W}))$, then the log-likelihood function can be written as

$$\ell(\boldsymbol{W}) = \log\prod_{i=1}^{N}\prod_{c=1}^{C}\mu_{ic}^{y_{ic}} = \sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic}\log\mu_{ic} \tag{8.9}$$

$$= \sum_{i=1}^{N}\left[\left(\sum_{c=1}^{C} y_{ic}\boldsymbol{w}_c^T\boldsymbol{x}_i\right) - \log\left(\sum_{c=1}^{C}\exp(\boldsymbol{w}_c^T\boldsymbol{x}_i)\right)\right] \tag{8.10}$$

Define the objective function as NLL

$$J(\boldsymbol{W}) = \text{NLL}(\boldsymbol{W}) = -\ell(\boldsymbol{W}) \tag{8.11}$$

Define $\boldsymbol{A}\otimes\boldsymbol{B}$ be the **kronecker product** of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.If $\boldsymbol{A}$ is an $m\times n$ matrix and $\boldsymbol{B}$ is a $p\times q$ matrix, then $\boldsymbol{A}\otimes\boldsymbol{B}$ is the $mp\times nq$ block matrix

$$\boldsymbol{A}\otimes\boldsymbol{B}\triangle\begin{pmatrix} a_{11}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ \vdots & \vdots & \vdots \\ a_{m1}\boldsymbol{B} & \cdots & a_{mn}\boldsymbol{B}\end{pmatrix} \tag{8.12}$$

The gradient and Hessian are given by

$$\boldsymbol{g}(\boldsymbol{W}) = \sum_{i=1}^{N}(\boldsymbol{\mu}-\boldsymbol{y}_i)\otimes\boldsymbol{x}_i \tag{8.13}$$

$$\boldsymbol{H}(\boldsymbol{W}) = \sum_{i=1}^{N}(\text{diag}(\boldsymbol{\mu}_i)-\boldsymbol{\mu}_i\boldsymbol{\mu}_i^T)\otimes(\boldsymbol{x}_i\boldsymbol{x}_i^T) \tag{8.14}$$

where $\boldsymbol{y}_i = (\mathbb{I}(y_i=1),\mathbb{I}(y_i=1),\cdots,\mathbb{I}(y_i=C-1))$ and $\boldsymbol{\mu}_i = (p(y=1|\boldsymbol{x}_i,\boldsymbol{W}), p(y=2|\boldsymbol{x}_i,\boldsymbol{W}),\cdots,p(y=C-1|\boldsymbol{x}_i,\boldsymbol{W}))$ are column vectors of length $C-1$.

Pass them to any gradient-based optimizer.

8.3.3 MAP

The new objective

$$J'(\boldsymbol{W}) = \text{NLL}(\boldsymbol{w}) - \log p(\boldsymbol{W}) \tag{8.15}$$

$$\text{, where } p(\boldsymbol{W}) \triangleq \prod_{c=1}^{C}\mathcal{N}(\boldsymbol{w}_c|\boldsymbol{0},\boldsymbol{V}_0)$$

$$= J(\boldsymbol{W}) + \frac{1}{2}\sum_{c=1}^{C}\boldsymbol{w}_c\boldsymbol{V}_0^{-1}\boldsymbol{w}_c \tag{8.16}$$

$$\tag{8.17}$$

Its gradient and Hessian are given by

$$\boldsymbol{g}'(\boldsymbol{w}) = \boldsymbol{g}(\boldsymbol{W}) + \boldsymbol{V}_0^{-1}\left(\sum_{c=1}^{C}\boldsymbol{w}_c\right) \tag{8.18}$$

$$\boldsymbol{H}'(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w}) + \boldsymbol{I}_{\boldsymbol{C}}\otimes\boldsymbol{V}_0^{-1} \tag{8.19}$$

This can be passed to any gradient-based optimizer to find the MAP estimate. Note, however, that the Hessian has size $((CD)(CD))$, which is C times more row and columns than in the binary case, so limited memory BFGS is more appropriate than Newtons method.

8.4 Bayesian logistic regression

It is natural to want to compute the full posterior over the parameters, $p(\boldsymbol{w}|\mathcal{D})$, for logistic regression models. This can be useful for any situation where we want to associate confidence intervals with our predictions (e.g., this is necessary when solving contextual bandit problems, discussed in Section TODO).

Unfortunately, unlike the linear regression case, this cannot be done exactly, since there is no convenient conjugate prior for logistic regression. We discuss one simple approximation below; some other approaches include MCMC (Section TODO), variational inference (Section TODO), expectation propagation (Kuss and Rasmussen 2005), etc. For notational simplicity, we stick to binary logistic regression.

8.4.1 Laplace approximation

8.4.2 Derivation of the BIC

8.4.3 Gaussian approximation for logistic regression

8.4.4 Approximating the posterior predictive

8.4.5 Residual analysis (outlier detection) *

8.5 Online learning and stochastic optimization

Traditionally machine learning is performed **offline**, however, if we have **streaming data**, we need to perform **online learning**, so we can update our estimates as each new data point arrives rather than waiting until the end (which may never occur). And even if we have a batch of data, we might want to treat it like a stream if it is too large to hold in main memory. Below we discuss learning methods for this kind of scenario.

TODO

8.5.1 The perceptron algorithm

8.5.1.1 Representation

$$\mathcal{H}: y = f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^T \boldsymbol{x} + b) \tag{8.20}$$

where $\text{sign}(x) = \begin{cases} +1, & x \geq 0 \\ -1, & x < 0 \end{cases}$, see Fig. 8.1[19].

8.5.1.2 Evaluation

$$L(\boldsymbol{w}, b) = -y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \tag{8.21}$$

$$R_{emp}(f) = -\sum_i y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \tag{8.22}$$

$$\tag{8.23}$$

8.5.1.3 Optimization

Primal form

Convergency

[19] https://en.wikipedia.org/wiki/Perceptron

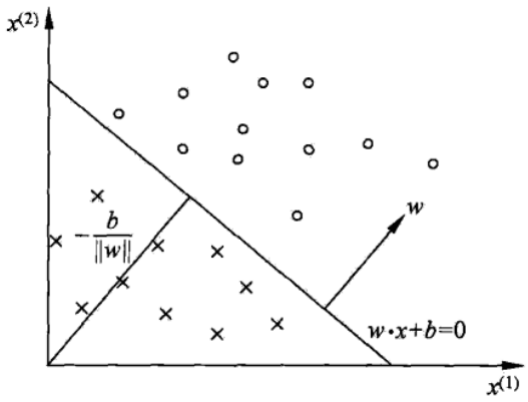

Fig. 8.1: Perceptron

$\boldsymbol{w} \leftarrow 0$; $b \leftarrow 0$; $k \leftarrow 0$;
while *no mistakes made within the for loop* **do**
  for $i \leftarrow 1$ **to** N **do**
    if $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \leq 0$ **then**
      $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y_i \boldsymbol{x}_i$;
      $b \leftarrow b + \eta y_i$;
      $k \leftarrow k + 1$;
    end
  end
end

Algorithm 1: Perceptron learning algorithm, primal form, using SGD

Theorem 8.1. ***(Novikoff)*** *If traning data set $\mathcal{D}$ is linearly separable, then*

1. *There exists a hyperplane denoted as $\widehat{\boldsymbol{w}}_{opt} \cdot \boldsymbol{x} + b_{opt} = 0$ which can correctly seperate all samples, and*

$$\exists \gamma > 0,\ \forall i,\ y_i(\boldsymbol{w}_{opt} \cdot \boldsymbol{x}_i + b_{opt}) \geq \gamma \tag{8.24}$$

2.

$$k \leq \left(\frac{R}{\gamma}\right)^2, \text{ where } R = \max_{1 \leq i \leq N} ||\widehat{\boldsymbol{x}}_i|| \tag{8.25}$$

Proof. (1) let $\gamma = \min_i y_i(\boldsymbol{w}_{opt} \cdot \boldsymbol{x}_i + b_{opt})$, then we get $y_i(\boldsymbol{w}_{opt} \cdot \boldsymbol{x}_i + b_{opt}) \geq \gamma$.

(2) The algorithm start from $\widehat{\boldsymbol{x}}_0 = 0$, if a instance is misclassified, then update the weight. Let $\widehat{\boldsymbol{w}}_{k-1}$ denotes the extended weight before the k-th misclassified instance, then we can get

$$y_i(\widehat{\boldsymbol{w}}_{k-1} \cdot \widehat{\boldsymbol{x}}_i) = y_i(\boldsymbol{w}_{k-1} \cdot \boldsymbol{x}_i + b_{k-1}) \leq 0 \tag{8.26}$$

$$\widehat{\boldsymbol{w}}_k = \widehat{\boldsymbol{w}}_{k-1} + \eta y_i \widehat{\boldsymbol{x}}_i \tag{8.27}$$

We could infer the following two equations, the proof procedure are omitted.

1. $\widehat{\boldsymbol{w}}_k \cdot \widehat{\boldsymbol{w}}_{opt} \geq k\eta\gamma$
2. $||\widehat{\boldsymbol{w}}_k||^2 \leq k\eta^2 R^2$

From above two equations we get

$$k\eta\gamma \leq \widehat{\boldsymbol{w}}_k \cdot \widehat{\boldsymbol{w}}_{opt} \leq ||\widehat{\boldsymbol{w}}_k|| \, ||\widehat{\boldsymbol{w}}_{opt}|| \leq \sqrt{k}\eta R$$
$$k^2\gamma^2 \leq kR^2$$
$$\text{i.e. } k \leq \left(\frac{R}{\gamma}\right)^2$$

Dual form

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i \tag{8.28}$$

$$b = \sum_{i=1}^{N} \alpha_i y_i \tag{8.29}$$

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{j=1}^{N} \alpha_j y_j \boldsymbol{x}_j \cdot \boldsymbol{x} + b\right) \tag{8.30}$$

$\boldsymbol{\alpha} \leftarrow 0$; $b \leftarrow 0$; $k \leftarrow 0$;
while *no mistakes made within the for loop* **do**
  for $i \leftarrow 1$ **to** N **do**
    if $y_i\left(\sum_{j=1}^{N} \alpha_j y_j \boldsymbol{x}_j \cdot \boldsymbol{x}_i + b\right) \leq 0$ **then**
      $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \eta$;
      $b \leftarrow b + \eta y_i$;
      $k \leftarrow k + 1$;
    end
  end
end

Algorithm 2: Perceptron learning algorithm, dual form

8.6 Generative vs discriminative classifiers

8.6.1 Pros and cons of each approach

- **Easy to fit?** As we have seen, it is usually very easy to fit generative classifiers. For example, in Sections 3.5.1 and 4.2.4, we show that we can fit a naive Bayes model and an LDA model by simple counting and averaging. By contrast, logistic regression requires solving a convex optimization problem (see Section 8.2 for the details), which is much slower.
- **Fit classes separately?** In a generative classifier, we estimate the parameters of each class conditional density independently, so we do not have to retrain the model when we add more classes. In contrast, in discriminative models, all the parameters interact, so the whole model must be retrained if we add a new class. (This is also the case if we train a generative model to maximize a discriminative objective Salojarvi et al. (2005).)
- **Handle missing features easily?** Sometimes some of the inputs (components ofx) are not observed. In a generative classifier, there is a simple method for dealing with this, as we discuss in Section 8.6.2. However, in a discriminative classifier, there is no principled solution to this problem, since the model assumes that $\boldsymbol{x}$is always available to be conditioned on (although see (Marlin 2008) for some heuristic approaches).
- **Can handle unlabeled training data?** There is much interest in **semi-supervised learning**, which uses unlabeled data to help solve a supervised task. This is fairly easy to do using generative models (see e.g., (Lasserre et al. 2006; Liang et al. 2007)), but is much harder to do with discriminative models.
- **Symmetric in inputs and outputs?** We can run a generative model backwards, and infer probable inputs given the output by computing $p(\boldsymbol{x}|y)$. This is not possible with a discriminative model. The reason is that a generative model defines a joint distribution on $\boldsymbol{x}$ and y, and hence treats both inputs and outputs symmetrically.
- **Can handle feature preprocessing?** A big advantage of discriminative methods is that they allow us to preprocess the input in arbitrary ways, e.g., we can replace $\boldsymbol{x}$ with $\boldsymbol{\phi}(\boldsymbol{x})$, which could be some basis function expansion, etc. It is often hard to define a generative model on such pre-processed data, since the new features are correlated in complex ways.
- **Well-calibrated probabilities?** Some generative models, such as naive Bayes, make strong independence assumptions which are often not valid. This can result in very extreme posterior class probabilities (very near 0 or 1). Discriminative models, such as logistic regression, are usually better calibrated in terms of their probability estimates.

See Table 8.1 for a summary of the classification and regression techniques we cover in this book.

8.6.2 Dealing with missing data

Sometimes some of the inputs (components of $\boldsymbol{x}$) are not observed; this could be due to a sensor failure, or a failure to complete an entry in a survey, etc. This is called the **missing data problem** (Little. and Rubin 1987). The ability to handle missing data in a principled way is one of the biggest advantages of generative models.

To formalize our assumptions, we can associate a binary response variable $r_i \in \{0,1\}$ that specifies whether each value $\boldsymbol{x}_i$ is observed or not. The joint model has the

Model	Classif/regr	Gen/Discr	Param/Non	Section
Discriminant analysis	Classif	Gen	Param	Sec. 4.2.2, 4.2.4
Naive Bayes classifier	Classif	Gen	Param	Sec. 3.5, 3.5.1.2
Tree-augmented Naive Bayes classifier	Classif	Gen	Param	Sec. 10.2.1
Linear regression	Regr	Discrim	Param	Sec. 1.4.5, 7.3, 7.6
Logistic regression	Classif	Discrim	Param	Sec. 1.4.6, 8.2.1.1, 8.4.3, 21.8.1.1
Sparse linear/ logistic regression	Both	Discrim	Param	Ch. 13
Mixture of experts	Both	Discrim	Param	Sec. 11.2.4
Multilayer perceptron (MLP)/ Neural network	Both	Discrim	Param	Ch. 16
Conditional random field (CRF)	Classif	Discrim	Param	Sec. 19.6
K nearest neighbor classifier	Classif	Gen	Non	Sec. TODO, TODO
(Infinite) Mixture Discriminant analysis	Classif	Gen	Non	Sec. 14.7.3
Classification and regression trees (CART)	Both	Discrim	Non	Sec. 16.2
Boosted model	Both	Discrim	Non	Sec. 16.4
Sparse kernelized lin/logreg (SKLR)	Both	Discrim	Non	Sec. 14.3.2
Relevance vector machine (RVM)	Both	Discrim	Non	Sec. 14.3.2
Support vector machine (SVM)	Both	Discrim	Non	Sec. 14.5
Gaussian processes (GP)	Both	Discrim	Non	Ch. 15
Smoothing splines	Regr	Discrim	Non	Section 15.4.6

Table 8.1: List of various models for classification and regression which we discuss in this book. Columns are as follows: Model name; is the model suitable for classification, regression, or both; is the model generative or discriminative; is the model parametric or non-parametric; list of sections in book which discuss the model. See also http://pmtk3.googlecode.com/svn/trunk/docs/tutorial/html/tutSupervised.html for the PMTK equivalents of these models. Any generative probabilistic model (e.g., HMMs, Boltzmann machines, Bayesian networks, etc.) can be turned into a classifier by using it as a class conditional density

form $p(\boldsymbol{x}_i, r_i|\boldsymbol{\theta}, \boldsymbol{\phi}) = p(r_i|\boldsymbol{x}_i, \boldsymbol{\phi})p(\boldsymbol{x}_i|\boldsymbol{\theta})$, where $\boldsymbol{\phi}$ are the parameters controlling whether the item is observed or not.

- If we assume $p(r_i|\boldsymbol{x}_i, \boldsymbol{\phi}) = p(r_i|\boldsymbol{\phi})$, we say the data is **missing completely at random** or **MCAR**.
- If we assume $p(r_i|\boldsymbol{x}_i, \boldsymbol{\phi}) = p(r_i|\boldsymbol{x}_i^o, \boldsymbol{\phi})$, where $\boldsymbol{x}_i^o$ is the observed part of $\boldsymbol{x}_i$, we say the data is **missing at random** or **MAR**.
- If neither of these assumptions hold, we say the data is **not missing at random** or **NMAR**. In this case, we have to model the missing data mechanism, since the pattern of missingness is informative about the values of the missing data and the corresponding parameters. This is the case in most collaborative filtering problems, for example.

See e.g., (Marlin 2008) for further discussion. We will henceforth assume the data is MAR.

When dealing with missing data, it is helpful to distinguish the cases when there is missingness only at test time (so the training data is **complete data**), from the harder case when there is missingness also at training time. We will discuss these two cases below. Note that the class label is always missing at test time, by definition; if the class label is also sometimes missing at training time, the problem is called semi-supervised learning.

8.6.2.1 Missing data at test time

In a generative classifier, we can handle features that are MAR by marginalizing them out. For example, if we are missing the value ofx1, we can compute

$$p(y=c|\boldsymbol{x}_{2:D}, \boldsymbol{\theta}) \propto p(y=c|\boldsymbol{\theta})p(\boldsymbol{x}_{2:D}|y=c, \boldsymbol{\theta}) \tag{8.31}$$

$$=\propto p(y=c|\boldsymbol{\theta}) \sum_{x_1} p(x_1, \boldsymbol{x}_{2:D}|y=c, \boldsymbol{\theta}) \tag{8.32}$$

Similarly, in discriminant analysis, no matter what regularization method was used to estimate the parameters, we can always analytically marginalize out the missing variables (see Section 4.3):

$$p(\boldsymbol{x}_{2:D}|y=c, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x}_{2:D}|\boldsymbol{\mu}_{c,2:D}, \boldsymbol{\Sigma}_{c,2:D}) \tag{8.33}$$

8.6.2.2 Missing data at training time

Missing data at training time is harder to deal with. In particular, computing the MLE or MAP estimate is no longer a simple optimization problem, for reasons discussed in Section TODO. However, soon we will study are a variety of more sophisticated algorithms (such as EM algorithm, in Section 11.4) for finding approximate ML or MAP estimates in such cases.

8.6.3 Fishers linear discriminant analysis (FLDA) *

TODO