Logistic regression

Mauricio A. Álvarez, PhD

Scalable Machine Learning,
University of Sheffield

Outline

Probabilistic classifier

- A logistic regression model is an example of a probabilistic classifier.
- Let $\mathbf{x} \in \mathbb{R}^p$ represents a feature vector and y the target value.
- For a binary classification problem we can use $y \in \{0, 1\}$ or $y \in \{-1, +1\}$.
- We model the relationship between y, and $\mathbf{x}$ using a Bernoulli distribution.

Bernoulli distribution (I)

- A Bernoulli random variable Y is a random variable that can only take two possible values.

- For example, the random variable Y associated to the experiment of tossing a coin.

- Output “heads” is assigned 1 ($Y = 1$), and output “tails” is assigned 0 ($Y = 0$).

Bernoulli distribution (II)

- A Bernoulli distribution is a probability distribution for Y, expressed as

$$p(Y = y) = \mathrm{Ber}(y|\mu) = \begin{cases} \mu & y = 1, \\ 1 - \mu & y = 0, \end{cases}$$

where $\mu = P(Y = 1)$.

- The expression above can be summarized in one line using

$$p(Y = y) = \mathrm{Ber}(y|\mu) = \mu^y (1 - \mu)^{1-y},$$

How are y and **x** related in logistic regression?

- The target feature y follows a Bernoulli distribution

$$p(y|\mathbf{x}) = \mathrm{Ber}(y|\mu(\mathbf{x})).$$

- Notice how the probability $\mu = P(y = 1)$ explicity depends on **x**.

- In logistic regression, the probability $\mu(\mathbf{x})$ is given as

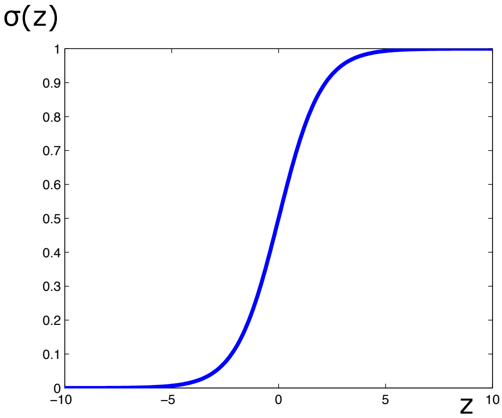
$$\mu(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^\top \mathbf{x})} = \sigma(\mathbf{w}^\top \mathbf{x}),$$

  where $\sigma(z)$ is known as the *logistic sigmoid* function.

- We then have

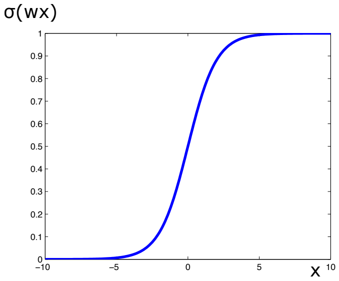
$$p(y|\mathbf{w}, \mathbf{x}) = \mathrm{Ber}(y|\sigma(\mathbf{w}^\top \mathbf{x})).$$

The logistic sigmoid function $\sigma(z)$



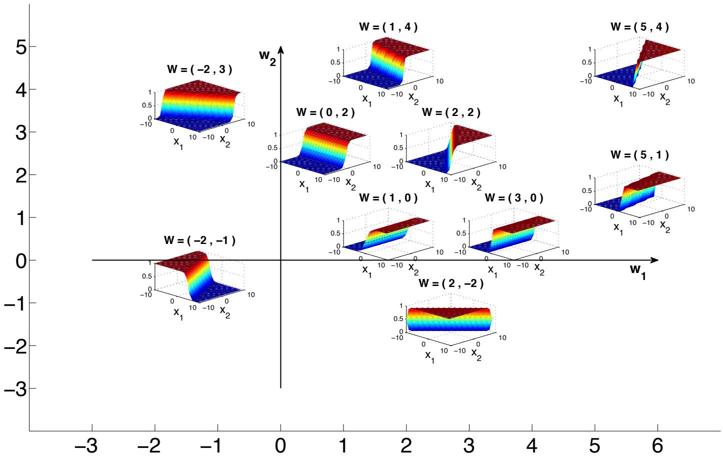
- Recall $\sigma(z) = \dfrac{1}{1 + \exp(-z)}$.
- If $z \to \infty$, $\sigma(z) = 1$. If $z \to -\infty$, $\sigma(z) = 0$. If $z = 0$, $\sigma(z) = 0.5$.

The logistic sigmoid function $\sigma(\mathbf{w}^\top \mathbf{x})$



- We have $z = \mathbf{w}^\top \mathbf{x}$. For simplicity, assume $\mathbf{x} = x$, then $\sigma(wx)$.
- Recall $\sigma(wx) = \dfrac{1}{1 + \exp(-wx)}$.
- So $\dfrac{d\sigma(wx)}{dx}\Big|_{x=0} = \dfrac{w}{4}$.

The logistic sigmoid function in 2d



Plot of $\sigma(w_1 x_1 + w_2 x_2)$. Here $\mathbf{w} = [w_1\ w_2]^\top$.

Decision boundary

- After the training phase, we will have an estimator for $\mathbf{w}$.
- For a test input vector $\mathbf{x}_*$, we compute $p(y = 1|\mathbf{w}, \mathbf{x}_*) = \sigma(\mathbf{w}^\top \mathbf{x}_*)$.
- This will give us a value between 0 and 1.
- We define a threshold of 0.5 to decide to which class we assign $\mathbf{x}_*$.
- With this threshold we induce a linear decision boundary in the input space.

Outline

Contents

Cross-entropy error function

- Let $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$.
- We write $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_N]^\top$, and $\mathbf{y} = [y_1 \cdots y_N]^\top$.
- Assuming IID observations

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}) = \prod_{n=1}^{N} p(y_n|\mathbf{w}, \mathbf{x}_n) = \prod_{n=1}^{N} \mathrm{Ber}(y_n|\sigma(\mathbf{w}^\top \mathbf{x}_n)).$$

- The cross-entropy function or negative log-likelihood is given as

$$\begin{aligned} NLL(\mathbf{w}) &= -\log p(\mathbf{y}|\mathbf{w}, \mathbf{X}) \\ &= -\sum_{n=1}^{N} \{y_n \log[\sigma(\mathbf{w}^\top \mathbf{x}_n)] + (1 - y_n) \log[1 - \sigma(\mathbf{w}^\top \mathbf{x}_n)]\}, \end{aligned}$$

which can be minimised with respect to **w**.

Gradient and Hessian of $NLL(\mathbf{w})$

- It can be shown that the gradient $\mathbf{g}(\mathbf{w})$ of $NLL(\mathbf{w})$ is given as

$$\mathbf{g}(\mathbf{w}) = \frac{d}{d\mathbf{w}} NLL(\mathbf{w}) = \sum_{n=1}^{N} [\sigma(\mathbf{w}^\top \mathbf{x}_n) - y_n]\mathbf{x}_n = \mathbf{X}^\top (\boldsymbol{\sigma} - \mathbf{y}),$$

  where $\boldsymbol{\sigma} = [\sigma(\mathbf{w}^\top \mathbf{x}_1) \cdots \sigma(\mathbf{w}^\top \mathbf{x}_N)]^\top$.

- It can also be shown that the Hessian $\mathbf{H}(\mathbf{w})$ of $NLL(\mathbf{w})$ is given as

$$\mathbf{H}(\mathbf{w}) = \frac{d}{d\mathbf{w}} \mathbf{g}(\mathbf{w})^\top = \sum_{n=1}^{N} \sigma(\mathbf{w}^\top \mathbf{x}_n)[1 - \sigma(\mathbf{w}^\top \mathbf{x}_n)]\mathbf{x}_n \mathbf{x}_n^\top = \mathbf{X}^\top \boldsymbol{\Sigma} \mathbf{X},$$

  where $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma(\mathbf{w}^\top \mathbf{x}_n)[1 - \sigma(\mathbf{w}^\top \mathbf{x}_n)])$.

Contents

Logistic loss (I)

- We could use $y \in \{-1, +1\}$ instead of $y \in \{0, 1\}$.

- In this case,

$$p(y|\mathbf{w}, \mathbf{x}) = \begin{cases} \frac{1}{1+\exp(+\mathbf{w}^\top \mathbf{x})} & y = -1, \\ \frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x})} & y = +1. \end{cases}$$

- Using one expression,

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

Logistic loss (II)

- Let $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, but each $y_n \in \{-1, +1\}$.
- Assuming IID observations $p(\mathbf{y}|\mathbf{w}, \mathbf{X}) = \prod_{n=1}^N \sigma(y_n \mathbf{w}^\top \mathbf{x}_n)$.
- The logistic loss is defined as

$$\begin{aligned} NLL(\mathbf{w}) &= -\log p(\mathbf{y}|\mathbf{w}, \mathbf{X}) \\ &= \sum_{n=1}^{N} \log[1 + \exp(-y_n \mathbf{w}^\top \mathbf{x}_n)]. \end{aligned}$$

which can be minimised with respect to $\mathbf{w}$.

Gradient and Hessian of $NLL(\mathbf{w})$

- It can be shown that the gradient $\mathbf{g}(\mathbf{w})$ of $LL(\mathbf{w})$ is given as

$$\mathbf{g}(\mathbf{w}) = \frac{d}{d\mathbf{w}} NLL(\mathbf{w}) = \sum_{n=1}^{N} [y_n \sigma(y_n \mathbf{w}^\top \mathbf{x}_n) - y_n] \mathbf{x}_n = \mathbf{X}^\top (\boldsymbol{\sigma} - \mathbf{y}),$$

where $\boldsymbol{\sigma} = [y_1 \sigma(y_1 \mathbf{w}^\top \mathbf{x}_1) \cdots y_N \sigma(y_N \mathbf{w}^\top \mathbf{x}_N)]^\top$.

- It can also be shown that the Hessian $\mathbf{H}(\mathbf{w})$ of $NLL(\mathbf{w})$ is given as

$$\mathbf{H}(\mathbf{w}) = \frac{d}{d\mathbf{w}} \mathbf{g}(\mathbf{w})^\top = \sum_{n=1}^{N} \sigma(y_n \mathbf{w}^\top \mathbf{x}_n)[1 - \sigma(y_n \mathbf{w}^\top \mathbf{x}_n)] \mathbf{x}_n \mathbf{x}_n^\top = \mathbf{X}^\top \boldsymbol{\Sigma} \mathbf{X},$$

where $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma(y_n \mathbf{w}^\top \mathbf{x}_n)[1 - \sigma(y_n \mathbf{w}^\top \mathbf{x}_n)])$.

Contents

General problem

- We are given a function $f(\mathbf{w})$, where $\mathbf{w} \in \mathbb{R}^p$.
- Aim: to find a value for $\mathbf{w}$ that minimises $f(\mathbf{w})$.
- Use an interative procedure

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \eta \mathbf{d}_k,$$

  where $\mathbf{d}_k$ is known as the search direction and it is such that

$$f(\mathbf{w}_{k+1}) < f(\mathbf{w}_k).$$

- The parameter η is known as the **step size** or **learning rate**.

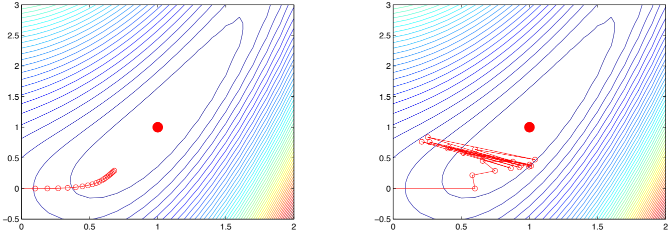
Gradient descent

- Perhaps, the simplest algorithm for unconstrained optimisation.
- It assumes that $\mathbf{d}_k = -\mathbf{g}_k$, where $\mathbf{g}_k = \mathbf{g}(\mathbf{w}_k)$.
- Also known as **steepest descent**.
- It can be written like

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \mathbf{g}_k.$$

Step size

- The main issue in gradient descent is how to set the step size.
- If it is too small, convergence will be very slow. If it is too large, the method can fail to converge at all.



(a) (b)

The function to optimise is $f(w_1, w_2) = 0.5(w_1^2 - w_2)^2 + 0.5(w_1 - 1)^2$. The minimum is at $(1, 1)$. In (a) $\eta = 0.1$. In (b) $\eta = 0.6$.

Alternatives to choose the step size η

- Line search methods (there are different alternatives).
- Line search methods may use search directions other than the steepest descent direction.
- Conjugate gradient (method of choice for quadratic objectives $f(\mathbf{w}) = \mathbf{w}^\top \mathbf{A}\mathbf{w}$).

Newton's method

- Another important search direction is the *Newton* direction.

- It derives a faster optimisation algorithm by taking the curvature of the space (i.e., the Hessian) into account.

- Optimisation methods that include the Hessian are also known as second order optimisation methods.

- In Newton's algorithm

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \mathbf{H}_k^{-1} \mathbf{g}_k,$$

  where $\mathbf{g}_k = \mathbf{g}(\mathbf{w}_k)$, and $\mathbf{H}_k = \mathbf{H}(\mathbf{w}_k)$.

- Usually $\eta = 1$, and $\mathbf{d}_k = -\mathbf{H}_k^{-1} \mathbf{g}_k$.

Newton's method and convex functions

- Newton's method requires that $\mathbf{H}_k$ be positive definite.
- Otherwise, we could not compute $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$.
- The condition holds if the function is strictly convex.
- Alternatives:
  - If $\mathbf{H}_k$ is not positive definite, use $\mathbf{d}_k = -\mathbf{g}_k$ instead.
  - Use the *Levenberg Marquardt* algorithm, which compromises between the Newton direction and the steepest direction.
  - Rather than computing $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$, find $\mathbf{d}_k$ that solves the linear system $\mathbf{H}_k\mathbf{d}_k = -\mathbf{g}_k$ using an iterative procedure. Stop when needed (**truncated Newton**).

Quasi-Newton methods

- The main drawback of the Newton direction is the need for the Hessian.
- Explicit computation of this matrix of second derivatives can sometimes be a cumbersome, error-prone, and expensive process.
- Quasi-Newton search directions provide an attractive alternative to Newton's method.
- They do not require computation of the Hessian and yet still attain a faster rate of convergence.
- In place of the true Hessian $\mathbf{H}_k$, they use an approximation $\mathbf{B}_k$, which is updated after each step to take account of the additional knowledge gained during the step.

BFGS formula for $\mathbf{H}_k$

- In the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) formula

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{\mathbf{z}_k \mathbf{z}_k^\top}{\mathbf{z}_k^\top \mathbf{s}_k} - \frac{(\mathbf{B}_k \mathbf{s}_k)(\mathbf{B}_k \mathbf{s}_k)^\top}{\mathbf{s}_k^\top \mathbf{B}_k \mathbf{s}_k},$$

  where $\mathbf{s}_k = \mathbf{w}_k - \mathbf{w}_{k-1}$ and $\mathbf{z}_k = \mathbf{g}_k - \mathbf{g}_{k-1}$.

- This is a rank-two update to the matrix, and ensures that the matrix remains positive definite.

- Ussually $\mathbf{B}_0 = \mathbf{I}$.

- The search direction is then $\mathbf{d}_k = -\mathbf{B}_k^{-1} \mathbf{g}_k$.

BFGS formula for $\mathbf{H}_k^{-1}$

- BFGS can alternatively update $\mathbf{C}_k = \mathbf{H}_k^{-1}$ using

$$\mathbf{C}_{k+1} = \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{z}_k^\top}{\mathbf{z}_k^\top \mathbf{s}_k}\right) \mathbf{C}_k \left(\mathbf{I} - \frac{\mathbf{z}_k \mathbf{s}_k^\top}{\mathbf{z}_k^\top \mathbf{s}_k}\right) + \frac{\mathbf{s}_k \mathbf{s}_k^\top}{\mathbf{z}_k^\top \mathbf{s}_k}.$$

- The search direction is then $\mathbf{d}_k = -\mathbf{C}_k \mathbf{g}_k$.

Limited memory BFGS (L-BFGS)

- Since storing the Hessian takes $O(p^2)$ space, for very large problems, one can use limited memory BFGS, or L-BFGS.
- In L-BFGS, $\mathbf{H}_k$ or $\mathbf{H}_k^{-1}$ is approximated by a diagonal plus low rank matrix.
- In particular, the product $\mathbf{B}_k^{-1}\mathbf{g}_k$ can be obtained by performing a sequence of inner products with $\mathbf{s}_k$ and $\mathbf{z}_k$, using only the m most recent $(\mathbf{s}_k, \mathbf{z}_k)$ pairs, and ignoring older information.
- The storage requirements are therefore $O(mp)$. Typically $m \sim 20$ suffices for good performance.

Optimisation methods applied to logistic regression

All the methods described above can be applied to find the parameter vector $\mathbf{w}$ that minimises the negative log-likelihood $NLL(\mathbf{w})$ in logistic regression.

Stochastic gradient descent (I)

- Traiditionally in machine learning, the gradient $\mathbf{g}_k$ and the Hessian $\mathbf{H}_k$ are computed using the whole dataset $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$.
- There are settings, though, where only a subset of the data can be used.
- **Online learning**: the instances $(\mathbf{x}_n, y_n)$ appear one at a time.
- **Large datasets**: computing the exact values for $\mathbf{g}_k$ and $\mathbf{H}_k$ would be expensive, if not impossible.

Stochastic gradient descent (II)

- In stochastic gradient descent (SGD), the gradient $\mathbf{g}_k$ is computed using a subset of the instances available.

- The word stochastic refers to the fact that the value for $\mathbf{g}_k$ will depend on the subset of the instances chosen for computation.

Stochastic gradient descent (III)

- In the stochastic setting, a better estimate can be found if the gradient is computed using

$$\mathbf{g}_k = \frac{1}{|S|} \sum_{i \in S} \mathbf{g}_{k,i},$$

where $S \in \mathcal{D}$, $|S|$ is the cardinality of S, and $\mathbf{g}_{k,i}$ is the gradient at iteration k computed using the instance $(\mathbf{x}_i, y_i)$.

- For logistic regression, $\mathbf{g}_{k,i}$ would be the gradient computed for $-\log p(y_i|\mathbf{w}, \mathbf{x}_i)$.

Step size in SGD

- Choosing the value of η is particularly important in SGD since there is no easy way to compute it.

- Usually the value of η will depend on the iteration k, η_k.

- It should follow the **Robbins-Monro** conditions

$$\sum_{k=1}^{\infty} \eta_k = \infty, \quad \sum_{k=1}^{\infty} \eta_k^2 < \infty.$$

- Various formulas for η_k can be used

$$\eta_k = \frac{1}{k}, \quad \eta_k = \frac{1}{(\tau_0 + k)^{\kappa}},$$

  where τ_0 slows down early interations and $\kappa \in (0.5, 1]$.

- Spark's option is $\eta_k = \frac{s}{\sqrt{k}}$, with `s = stepsize`.

Outline

What is regularisation?

- It refers to a technique used for preventing overfitting in a predictive model.

- It consists in adding a term (a regulariser) to the objective function that encourages simple solutions.

- With regularisation, the objective function for logistic regression would be

$$f(\mathbf{w}) = NLL(\mathbf{w}) + \lambda R(\mathbf{w}),$$

  where $R(\mathbf{w})$ is the regularisation term and λ the regularisation parameter.

- If $\lambda = 0$, we get $f(\mathbf{w}) = NLL(\mathbf{w})$.

Regularisation for logistic regression

- Suppose the data is linearly separable.
- The optimum for $f(\mathbf{w})$ would be obtain if $\|\mathbf{w}\| \to \infty$.
- Such a solution is very brittle and will not generalize well.

Different types of regularisation

- The objective function for logistic regression would be

$$f(\mathbf{w}) = NLL(\mathbf{w}) + \lambda R(\mathbf{w}),$$

where $R(\mathbf{w})$ follows as

$$R(\mathbf{w}) = \alpha\|\mathbf{w}\|_1 + (1-\alpha)\frac{1}{2}\|\mathbf{w}\|_2^2,$$

where $\|\mathbf{w}\|_1 = \sum_{m=1}^{p} |w_m|$, and $\|\mathbf{w}\|_2^2 = \sum_{m=1}^{p} w_m^2$.

- If $\alpha = 1$, we get ℓ_1 regularisation.
- If $\alpha = 0$, we get ℓ_2 regularisation.
- If $0 < \alpha < 1$, we get the elastic net regularisation.
- In Spark, λ is `regParam` and α is `elasticNetParam`.

Outline

Categorial distribution

- Imagine an random experiment where the output can only take 1 of K outputs.

- The result of the experiment can be coded using a vector $\mathbf{y}$ of dimension K, where only one of its entries is 1, and the rest is zero.

- Each of the K outputs of the experiment has a probability μ_k with $\sum_{k=1}^{K} \mu_k = 1$.

- We say that the vector $\mathbf{y}$ follows a categorial distribution

$$p(\mathbf{y}|\boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_k^{y_k},$$

where $\mathbf{y} = [y_1, y_2, \cdots, y_K]^\top$, and $\boldsymbol{\mu} = [\mu_1, \mu_2, \cdots, \mu_K]^\top$.

Multinomial logistic regression (I)

- The categorial distribution is a particular case of the multinomial distribution.
- The categorial distribution can be used for multi-class logistic regression.
- Each instance in a multi-class calssification problem is made of $(\mathbf{x}, \mathbf{y})$.
- In multinomial logistic regression, each probability μ_k is represented as

$$\mu_k = \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x})},$$

where $\{\mathbf{w}_k\}_{k=1}^{K}$ is a set of weigths, that we jointly refer to as $\mathbf{W}$.

Multinomial logistic regression (II)

- In this way, we can model $p(\mathbf{y}|\mathbf{W}, \mathbf{x})$ using

$$p(\mathbf{y}|\mathbf{W}, \mathbf{x}) = \prod_{k=1}^{K} \mu_k^{y_k} = \prod_{k=1}^{K} \left[\frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x})} \right]^{y_k} .$$

- Suppose we have a dataset $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}$.

- We write $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_N]^\top$, and $\mathbf{Y} = [\mathbf{y}_1 \cdots \mathbf{y}_N]^\top$.

Multinomial logistic regression (III)

- The parameters $\mathbf{W}$ could be estimated by minimising

$$NLL(\mathbf{W}) = -\log p(\mathbf{Y}|\mathbf{W},\mathbf{X}) = -\log \prod_{n=1}^{N} \prod_{k=1}^{K} \mu_{n,k}^{y_{n,k}} = -\sum_{n=1}^{N} \sum_{k=1}^{K} y_{n,k} \log \mu_{n,k},$$

where $\mu_{n,k} = \frac{\exp(\mathbf{w}_k^\top \mathbf{x}_n)}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^\top \mathbf{x}_n)}$.

- We could also add a regularisation term to $NLL(\mathbf{W})$, and minimise

$$f(\mathbf{W}) = NLL(\mathbf{W}) + \lambda R(\mathbf{W}),$$

with respect to $\mathbf{W}$.

Outline

Contents

`LogisticRegression()`

- `LogisticRegression()` works with DataFrames, so the feature vectors should be created using `org.apache.spark.ml.linalg.Vectors`
- It is possible to use regularisation ℓ_1, ℓ_2 or Elastic Net.
- L-BFGS is used as a solver for `LogisticRegression()`, with ℓ_2.
- When ℓ_1, or Elastic Net are used, a variant of L-BFGS, known as Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) is used instead.

Contents

`LogisticRegressionWithLBFGS()`

- `LogisticRegressionWithLBFGS()` uses LabeledPoints.
- It supports both binary and multinomial Logistic Regression.
- By default, it uses ℓ_2 regularisation.
- ℓ_1 regularisation can be used by setting `setUpdater` to `L1Updater`

LogisticRegressionWithSGD()

Deprecated since version 2.0.0.