Logistic Regression

Sandro Cumani

sandro.cumani@polito.it

Politecnico di Torino

Discriminative Linear Models – Logistic Regression

Logistic regression, despite its name, is a discriminative approach for classification

Rather than modeling the distribution of observed samples $\boldsymbol{X}|C$, we directly model the class posterior distribution $C|\boldsymbol{X}$

We need to define a model for the class posterior distribution $P(C = c|\boldsymbol{X} = \boldsymbol{x})$

Discriminative Linear Models – Logistic Regression

For a 2–class problem, we have seen that the Gaussian model with tied covariances provides log-likelihood ratios that are linear functions of our data

$$l(\boldsymbol{x}) = \log \frac{f_{X|C}(\boldsymbol{x}|h_1)}{f_{X|C}(\boldsymbol{x}|h_0)} = \boldsymbol{w}^T\boldsymbol{x} + c$$

and the class log-posterior probability ratio is

$$\log \frac{P(C = h_1|\boldsymbol{x})}{P(C = h_0|\boldsymbol{x})} = \log \frac{f_{X|C}(\boldsymbol{x}|h_1)}{f_{X|C}(\boldsymbol{x}|h_0)} + \log \frac{\pi}{1-\pi} = \boldsymbol{w}^T\boldsymbol{x} + b$$

The prior information has been absorbed in the bias term b.

Logistic Regression

Given $\boldsymbol{w}$ and b, we can compute the expression for the posterior class probability as

$$\begin{aligned} P(C = h_1|\boldsymbol{x}, \boldsymbol{w}, b) &= e^{(\boldsymbol{w}^T\boldsymbol{x}+b)} P(C = h_0|\boldsymbol{x}, \boldsymbol{w}, b) \\ &= e^{(\boldsymbol{w}^T\boldsymbol{x}+b)} (1 - P(C = h_1|\boldsymbol{x}, \boldsymbol{w}, b)) \end{aligned}$$

Solving for $P(C = h_1|\boldsymbol{x}, \boldsymbol{w}, b)$ we obtain

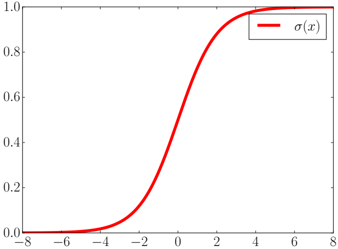
$$P(C = h_1|\boldsymbol{x}, \boldsymbol{w}, b) = \frac{e^{(\boldsymbol{w}^T\boldsymbol{x}+b)}}{1 + e^{\boldsymbol{w}^T\boldsymbol{x}+b}} = \frac{1}{1 + e^{-(\boldsymbol{w}^T\boldsymbol{x}+b)}} = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$$

where

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

is called *sigmoid* function (or *logistic* function)

Sigmoid function:



Some properties of $\sigma(x)$ that will come useful later

- $1 - \sigma(x) = \sigma(-x)$
- $\frac{d\sigma(x)}{dx} = \sigma(x)\left(1 - \sigma(x)\right)$

Logistic Regression

The equation $P(C = h_1|\boldsymbol{x}, \boldsymbol{w}, b) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$ provides a model that allows computing the posterior probabilities for h_1 and h_0

The model assumes that decision rules are linear surfaces (hyperplanes) orthogonal to $\boldsymbol{w}$. The model parameters are $(\boldsymbol{w}, b)$

If we knew $(\boldsymbol{w}, b)$ then we could compute the predictive distribution for the class labels $P(C = h_1|\boldsymbol{x}, \boldsymbol{w}, b)$

We have seen how we can compute estimates for $(\boldsymbol{w}, b)$ from a generative Gaussian model

However, in the following, we ignore the generative model for $\boldsymbol{X}$ and concentrate on the form of the class posterior probabilities

Again, we will follow a frequentist approach, i.e. compute an estimate for $\boldsymbol{w}$ and b from a set of training samples

Logistic Regression

We assume we have a labeled dataset

$$\mathcal{D} = [(\boldsymbol{x}_1, c_1), \ldots (\boldsymbol{x}_n, c_n)]$$

We also assume classes are independently distributed as

$$C_i | \boldsymbol{x}_i, \boldsymbol{w}, b \sim C | \boldsymbol{x}_i, \boldsymbol{w}, \boldsymbol{b}$$

The class-posterior model allows expressing the likelihood for the observed labels as

$$P(C_1 = c_1, \ldots C_n = c_n | \boldsymbol{x}_1, \ldots \boldsymbol{x}_n, \boldsymbol{w}, b) = \prod_{i=1}^{n} P(C_i = c_i | \boldsymbol{x}_i, \boldsymbol{w}, b)$$

We can apply a ML approach to estimate the model parameters that best describe the observed labels $(c_1 \ldots c_n)$

Rather than estimating $\boldsymbol{w}$ from class–conditional likelihoods, we estimate the value of $\boldsymbol{w}$ that maximizes the likelihood of our training labels.

We assume that the label for class h_1 is 1, and the label for class h_0 is 0

Let

$$y_i = P(C_i = 1 | \boldsymbol{x}_i, \boldsymbol{w}, b) = \sigma(\boldsymbol{w}^T \boldsymbol{x}_i + b)$$

It follows that

$$P(C_i = 0 | \boldsymbol{x}_i, \boldsymbol{w}, b) = 1 - y_i = 1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}_i + b) = \sigma(-\boldsymbol{w}^T \boldsymbol{x} - b)$$

Logistic Regression

The distribution for $C_i|\boldsymbol{x}_i, \boldsymbol{w}, b$ is a Bernoulli distribution

$$C_i|\boldsymbol{x}_i, \boldsymbol{w}, \boldsymbol{b} \sim \text{Ber}\left(\sigma(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right) = \text{Ber}(y_i)$$

The likelihood for our label set is

$$\begin{aligned}
\mathcal{L}(\boldsymbol{w}, b) &= P(C_1 = c_1, \ldots C_n = c_n|\boldsymbol{x}_1, \ldots \boldsymbol{x}_n, \boldsymbol{w}, b) \\
&= \prod_{i=1}^{n} P(C_i = c_i|\boldsymbol{x}_i, \boldsymbol{w}, b) \\
&= \prod_{i} y_i^{c_i}(1 - y_i)^{(1-c_i)}
\end{aligned}$$

Again, it's more practical to work with the log-likelihood

$$\ell(\boldsymbol{w}, b) = \log \mathcal{L}(\boldsymbol{w}, b) = \sum_{i=1}^{n} \left[c_i \log y_i + (1 - c_i)\log(1 - y_i)\right]$$

Our goal is the maximization of ℓ. We thus seek $\boldsymbol{w}^*, b^*$ that maximize $\ell(\boldsymbol{w}, b)$:

$$\boldsymbol{w}^*, b^* = \arg\max_{\boldsymbol{w},b} \ell(\boldsymbol{w}, b) = \arg\max_{\boldsymbol{w},b} \sum_{i=1}^{n} \left[c_i \log y_i + (1 - c_i) \log(1 - y_i)\right]$$

The ML solution is also the solution that minimizes the average cross-entropy between the distribution of observed and predicted labels

Rather than maximizing $\ell(\boldsymbol{w}, b)$, we can minimize

$$J(\boldsymbol{w}, b) = -\ell(\boldsymbol{w}, b) = \sum_{i=1}^{n} -\left[c_i \log y_i + (1 - c_i) \log(1 - y_i)\right]$$

The expression

$$H(c_i, y_i) = -\left[c_i \log y_i + (1 - c_i) \log (1 - y_i)\right]$$

represents the binary *cross–entropy* between the distribution of observed and predicted labels for the i-th sample

More in general, let P and Q be two distributions over the same domain

The cross-entropy between the two distributions is defined as

$$H(P, Q) = -\mathbb{E}_{P(x)}\left[\log Q(x)\right]$$

For discrete distributions, this can be expressed as

$$H(P, Q) = -\sum_{x \in \mathcal{S}} P(x) \log Q(x)$$

In our case, P is the empirical distribution of class labels, from the point of view of an observer $\mathcal{E}$ who knows the actual label:

$$P(C_i = 1|\boldsymbol{X}_i = \boldsymbol{x}_i, \mathcal{E}) = \begin{cases} 1 & \text{if } c_i = 1 \\ 0 & \text{if } c_i = 0 \end{cases}$$

$$P(C_i = 0|\boldsymbol{X}_i = \boldsymbol{x}_i, \mathcal{E}) = \begin{cases} 0 & \text{if } c_i = 1 \\ 1 & \text{if } c_i = 0 \end{cases}$$

or, equivalently

$$P(C_i = 1|\boldsymbol{X}_i = \boldsymbol{x}_i, \mathcal{E}) = c_i \ , \quad P(C_i = 0|\boldsymbol{X}_i = \boldsymbol{x}_i, \mathcal{E}) = 1 - c_i$$

i.e., a Bernoulli distribution with parameter c_i

Logistic Regression

Distribution Q is the distribution for the predicted labels according to our recognizer $\mathcal{R}$

$$Q(c) = P(C_i = c | \boldsymbol{X}_i = \boldsymbol{x}_i, \mathcal{R}(\boldsymbol{w}, b))$$

i.e.

$$Q(1) = P\left(C_i = 1 | \boldsymbol{X}_i = \boldsymbol{x}_i, \mathcal{R}(\boldsymbol{w}, b)\right) = y_i = \sigma(\boldsymbol{w}^T \boldsymbol{x}_i + b)$$
$$Q(0) = P\left(C_i = 0 | \boldsymbol{X}_i = \boldsymbol{x}_i, \mathcal{R}(\boldsymbol{w}, b)\right) = 1 - y_i = 1 - \sigma(\boldsymbol{w}^T \boldsymbol{x}_i + b)$$

Logistic regression looks for the minimizer of the average cross-entropy between the distributions for the training set labels of an evaluator $\mathcal{E}$ who knows the real label and the distributions for the training set labels as predicted by the model $\mathcal{R}(\boldsymbol{w}, b)$ itself

The cross-entropy is a measure of goodness of the predictions, and the evaluation is performed over the training data itself

The cross-entropy, as a function of Q, is minimized when $Q = P$

The cross–entropy can also be interpreted as a measure of the difference between P and Q

In our case, it measures how different is the predicted distribution $\mathrm{Ber}(y_i)$ from the empirical label distribution $\mathrm{Ber}(c_i)$ (the distribution of the evaluator $\mathcal{E}$)

Minimization of the average cross-entropy means we are looking for a label distribution that is (on average) as similar as possible to the empirical one

Alternatively, as we have seen, we can regard the process as maximization of the likelihood for the observed labels

Another interesting interpretation of the logistic regression objective can be obtained by rewriting the cross-entropy in terms of $z_i = 2c_i - 1$, i.e.

The terms z_i still represent class labels, however for samples of class h_1 we have $z_i = 1$, whereas for samples of class h_0 we have $z_i = -1$:

$$z_i = \begin{cases} 1 & \text{if } c_i = 1 \\ -1 & \text{if } c_i = 0 \end{cases}$$

Logistic Regression

The objective function that we want to minimize corresponds to the sum of n terms

$$J(\boldsymbol{w}, b) = \sum_i H(c_i, y_i)$$

where

$$H(c_i, y_i) = -\left[c_i \log y_i + (1 - c_i) \log(1 - y_i)\right]$$

Note that $H(c_i, y_i)$ is a function of c_i, but also of $\boldsymbol{w}, b$ and $\boldsymbol{x}_i$, since

$$y_i = \sigma(\boldsymbol{w}^T \boldsymbol{x}_i + b)$$

Let $s_i = \boldsymbol{w}^T \boldsymbol{x}_i + b$. In terms of z_i we can rewrite H as

$$
\begin{aligned}
H(c_i, y_i) &= -\left[c_i \log y_i + (1 - c_i) \log(1 - y_i)\right] \\
&= \begin{cases} -\log \sigma(s_i) & \text{if } c_i = 1 \ (z_i = 1) \\ -\log\left(1 - \sigma(s_i)\right) = -\log \sigma(-s_i) & \text{if } c_i = 0 \ (z_i = -1) \end{cases} \\
&= -\log \sigma(z_i s_i) \\
&= -\log \sigma\left(z_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)\right) \\
&= \log\left(1 + e^{-z_i(\boldsymbol{w}^T \boldsymbol{x}_i + b)}\right)
\end{aligned}
$$

The objective function can thus be rewritten as

$$
\begin{aligned}
J(\boldsymbol{w}, b) &= \sum_{i=1}^{n} H(c_i, y_i) \\
&= \sum_{i=1}^{n} \log\left(1 + e^{-z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}\right) \\
&= \sum_{i=1}^{n} l\left(z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)\right)
\end{aligned}
$$

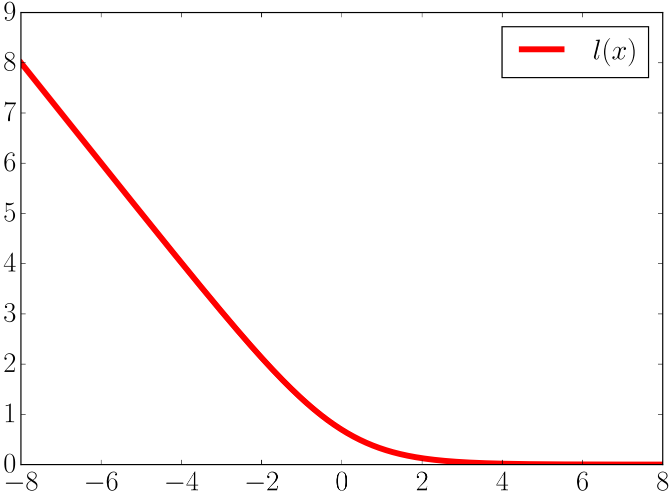
where

$$
l(x) = \log\left(1 + e^{-x}\right)
$$

is the logistic loss function.

Our goal is to find the minimizer of $J(\boldsymbol{w}, \boldsymbol{b})$

Logistic loss



Logistic Regression

We can interpret the function as the cost of the prediction made with model $(\boldsymbol{w}, b)$ for each sample

Remember that class log-posterior probability ratio is

$$\log \frac{P(h_1|\boldsymbol{x}_i)}{P(h_0|\boldsymbol{x}_i)} = \boldsymbol{w}^T\boldsymbol{x}_i + b = s_i$$

Decision rules take the form $s_i \lessgtr t$

Since $s_i = \boldsymbol{w}^T\boldsymbol{x}_i + b$, decision rules are linear hyperplane orthogonal to the vector $\boldsymbol{w}$

s_i is related to the distance of the sample $\boldsymbol{x}_i$ from the separating surface

When s_i is positive, our classifier is favoring class h_1, whereas negative s_i means we are classifying the sample as belonging to class h_0.

The cost we pay for each sample is $l(z_i s_i)$

- The prediction and the actual class agree: $z_i = 1, s_i > 0$ or $z_i = -1, s_i < 0$. Then $z_i s_i > 0$, and we pay a low cost. The cost becomes exponentially smaller (asymptotically) as the absolute value of s_i increases (we move away from the separation surface)
- The prediction and the actual class disagree: $z_i = 1, s_i < 0$ or $z_i = -1, s_i > 0$. Then $z_i s_i < 0$, and we pay a cost that increases (asymptotically) linearly with s_i

We can thus interpret the logistic regression objective as a measure of an empirical risk[1]. Our goal is minimizing the empirical risk

More in general, empirical risk minimization is a framework for the estimation of classification models which aims at minimizing an empirical risk function over our training data

Generalized risk minimization problem: minimize the risk $R(\boldsymbol{\theta})$

$$R(\boldsymbol{\theta}) = \sum_i l(\boldsymbol{w}, \boldsymbol{x}_i, z_i)$$

where l is called loss (or cost) function, and $\boldsymbol{\theta}$ are the parameters of the classification model, e.g. $\boldsymbol{\theta} = (\boldsymbol{w}, b)$ in our case.

[1] Empirical because it's computed on the observed samples

Logistic regression solutions cannot be computed in closed form

We will resort to numerical solvers

A numerical solver iteratively looks for the minimizer of a function

We will use the L-BFGS algorithm

The algorithm requires a function that computes the loss and its gradient with respect to $\boldsymbol{w}$ and b

In the laboratory we will see how to implement the minimization

Logistic Regression

If classes are linearly separable, the logistic regression solution is not defined

Linearly separable classes: there exist $\boldsymbol{w}$ and b such that all training samples lie on the correct side of the corresponding separation surface ($z_i > 0 \iff s_i > 0$)

In this case, we can make the values of s_i arbitrarily high by simply increasing the norm of $\boldsymbol{w}$ (and changing accordingly the value of b)

As we increase $\|\boldsymbol{w}\|$, the loss becomes lower, thus we are decreasing the objective function

The function does not have a minimum, but has an infimum $\inf J(\boldsymbol{w}, b) = 0$, corresponding to $\|\boldsymbol{w}\| \to \infty$

To make the problem solvable again, we can look for solutions with small norm by introducing a norm penalty to the objective function

The penalty is called regularization term

The objective function that we minimize is

$$\tilde{R}(\boldsymbol{w}, b) = \frac{\tilde{\lambda}}{2}\|\boldsymbol{w}\|^2 + \sum_{i=1}^{n} \log\left(1 + e^{-z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}\right)$$

where $\tilde{\lambda}$ is a hyper-parameter that allows specifying the relative weight of the regularization term

Alternatively, we look for the minimizer of

$$R(\boldsymbol{w}, b) = \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + e^{-z_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}\right)$$

where the risk is *averaged* over all samples, and λ is the regularization coefficient.

λ is a hyper-parameter, and should be selected as to optimize the performance of the classifier

Note that λ cannot be computed by minimizing R with respect to λ, as we would obtain the trivial solution $\lambda = 0$

The selection of good values for λ should thus be based on other approaches, such as cross-validation

The model is called regularized Logistic Regression, and is an example of a regularized risk minimization problem

$$R(\boldsymbol{w}, b) = \Omega(\boldsymbol{w}, b) + \frac{1}{n}\sum_{i=1}^{n} l(\boldsymbol{x}_i, z_i, \boldsymbol{w}, \boldsymbol{b})$$

Logistic Regression

The regularization term Ω (in our case $\frac{\lambda}{2}\|\boldsymbol{w}\|^2$) can be interpreted as a term that favors simpler solutions (we will see explicitly why small norm of $\boldsymbol{w}$ can be interpreted as a simpler solution when discussing Support Vector Machines)

Regularization allows reducing the risk of over-fitting the training data

Of course, if λ is too large, we will obtain a solution that has small norm, but is not able to well separate the classes

On the other hand, if λ is too small, we will get a solution that has good separation on the training set, but may have poor classification accuracy for unseen data (i.e. poor generalization)

Some considerations:

- The non-regularized model is invariant to linear transformations of the feature vectors.
- The regularized version of the model, on the other hand, is not invariant.
- It is therefore useful, in some cases, to pre-process data so that dynamic ranges of different features are similar
- Cross-validation can help in identifying good pre-processing strategies

Logistic Regression

Common preprocessing strategies that may be worth trying:

- Center the data (either using the training set, or a weighted mean — e.g. the average of class means): $\boldsymbol{x}_i' = \boldsymbol{x}_i - \boldsymbol{\mu}$
- Standardize variances (e.g. divide each feature by its own standard deviation computed over the training set): $\boldsymbol{x}_{i,[j]}' = \boldsymbol{x}_{i,[j]} / \sigma_{[j]}$
- Whiten the covariance matrix (i.e. normalize variances while making features uncorrelated): $\boldsymbol{x}_i' = \boldsymbol{A}\boldsymbol{x}_i$, where $\boldsymbol{A} = \boldsymbol{\Sigma}^{\frac{1}{2}}$ and $\boldsymbol{\Sigma}$ is the training set covariance (a variation consists in replacing $\boldsymbol{\Sigma}$ with the within-class covariance)
- L2 (or length) normalization: $\boldsymbol{x}_i' = \frac{\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|}$ (often after centering and whitening)

Some considerations:

- The Logistic Regression score can be interpreted as the logarithm of the ratio between class posterior probabilities
- It reflects the empirical class prior of the training data
- We can simulate different empirical priors π_T using a prior-weighted version of the model

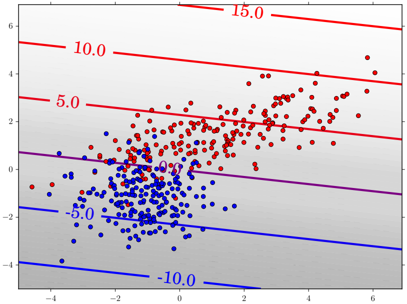
$$R(w) = \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{\pi_T}{n_T}\sum_{i|z_i=1} l(z_i s_i) + \frac{1-\pi_T}{n_F}\sum_{i|z_i=-1} l(z_i s_i)$$

Some considerations:

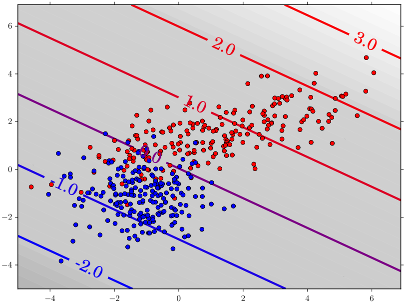
- If our application is characterized by the same effective prior π_T, the optimal decision should correspond to $\boldsymbol{w}^T\boldsymbol{x} + b \lessgtr 0$
- In theory, we can also recover log-likelihood ratios by subtracting from the score s the empirical prior log-odds of the training set $\log \frac{n_T}{n_F}$, where n_T is the number of samples of class $h_1 (c_i = 1, z_i = +1)$ and n_F is the number of samples of class $h_0 (c_i = 0, z_i = -1)$
- In some cases, especially if the dimensionality of the space is large, the score s may not provide the correct probabilistic interpretation, and optimal decisions may require either re-calibration of the scores or selection of an optimal threshold based on a validation set

Logistic Regression

$\lambda = 0$



$\lambda = 1$



Logistic Regression

We test the model on MNIST digit pairs (e.g. 0 vs 1, 0 vs 2, ...)

MNIST — Average Pairwise EER for Logistic Regression

DimRed	$\lambda = 0$	$\lambda = 0.00001$	$\lambda = 0.001$	$\lambda = 0.1$	Tied Gau
RAW [768]	1.7%	1.4%	$\boxed{1.2\%}$	2.0%	—
PCA [50]	1.4%	1.4%	1.4%	2.1%	1.7%
PCA [100]	1.3%	$\boxed{1.2\%}$	$\boxed{1.2\%}$	2.0%	1.5%

LogReg obtains better performance than the Gaussian model

Regularization is important, especially when we do not reduce the dimensionality (we have more parameters to estimate, so over-fitting is more sever)

If we regularize too much the model performs poorly again

Multiclass Logistic Regression

We now consider a problem with K classes, labeled from 1 to K

To extend the Logistic Regression model to multiclass tasks we start again from the form of the posterior likelihood ratios of the Linear Gaussian classifier with uniform priors

$$\log \frac{P(C = j|\boldsymbol{x})}{P(C = r|\boldsymbol{x})} = (\boldsymbol{w}_j - \boldsymbol{w}_r)^T \boldsymbol{x} + (b_j - b_r)$$

corresponding to pair–wise linear classification surfaces between class j and a reference class r

Notice that we over-parametrized the model by introducing an extra set of parameters $(\boldsymbol{w}_r, b_r)$, so that we do not have to explicitly enforce $\log \frac{P(C=r|\boldsymbol{x})}{P(C=r|\boldsymbol{x})} = 0$

Multiclass Logistic Regression

It follows that, for all classes $j \in 1, \dots K$,

$$P(C = j|\boldsymbol{x}) = P(C = r|\boldsymbol{x})e^{(\boldsymbol{w}_j - \boldsymbol{w}_r)^T \boldsymbol{x} + (b_j - b_r)}$$

Remembering that

$$\sum_{j=1}^{K} P(C = j|\boldsymbol{x}) = 1$$

We have

$$P(C = r|\boldsymbol{x}) = 1 - \sum_{j \neq r} P(C = j|\boldsymbol{x}) = 1 - \sum_{j \neq r} P(C = r|\boldsymbol{x})e^{(\boldsymbol{w}_j - \boldsymbol{w}_r)^T \boldsymbol{x} + (b_j - b_r)}$$

Therefore

$$P(C = r|\boldsymbol{x}) = \frac{1}{1 + \sum_{j \neq r} e^{(\boldsymbol{w}_j - \boldsymbol{w}_r)^T \boldsymbol{x} + (b_j - b_r)}}$$
$$= \frac{1}{\sum_j e^{(\boldsymbol{w}_j - \boldsymbol{w}_r)^T \boldsymbol{x} + (b_j - b_r)}}$$
$$= \frac{e^{\boldsymbol{w}_r^T \boldsymbol{x} + b_r}}{\sum_j e^{\boldsymbol{w}_j^T \boldsymbol{x} + b_j}}$$

Note that the model is completely specified (and actually over-parametrized) once we know the K vectors $\boldsymbol{w}_k$ and the terms b_k. Repeating the operations for all classes we obtain

$$P(C = k|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}_k^T \boldsymbol{x} + b_k}}{\sum_j e^{\boldsymbol{w}_j^T \boldsymbol{x} + b_j}}$$

Function $f_i(\boldsymbol{s}) = \frac{e^{s_i}}{\sum_j e^{s_j}}$ is called softmax

Multiclass Logistic Regression

Given the model parameters

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{w}_1 & \dots & \boldsymbol{w}_K \end{bmatrix} \ , \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_K \end{bmatrix}$$

the logistic regression model allows computing the probability of each class

$$P(C = k | \boldsymbol{W}, \boldsymbol{b}, \boldsymbol{x}) = \frac{e^{\boldsymbol{w}_k^T \boldsymbol{x} + b_k}}{\sum_j e^{\boldsymbol{w}_j^T \boldsymbol{x} + b_j}}$$

If we consider sample $\boldsymbol{x}_i$, its class posterior distribution is thus a categorical distribution

$$C_i | \boldsymbol{W}, \boldsymbol{b}, \boldsymbol{X}_i = \boldsymbol{x}_i \sim \mathrm{Cat}(\boldsymbol{y}_i)$$

where

$$\boldsymbol{y}_{ik} = \frac{e^{\boldsymbol{w}_k^T \boldsymbol{x}_i + b_k}}{\sum_j e^{\boldsymbol{w}_j^T \boldsymbol{x}_i + b_j}}$$

Multiclass Logistic Regression

As for the binary case, we can express the log-likelihood for the training class labels as

$$\ell(\boldsymbol{W}, \boldsymbol{b}) = \sum_{i=1}^{n} \log P(C_i = c_i | \boldsymbol{X}_i = \boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{b})$$

Remember that the categorical density $P(C_i = c_i | \boldsymbol{X}_i = \boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{b})$ can be expressed using a 1-of-K encoding, as

$$\log P(C_i = c_i | \boldsymbol{X}_i = \boldsymbol{x}_i, \boldsymbol{W}, \boldsymbol{b}) = \log P(C_i = c_i | \boldsymbol{y}_i) = \sum_{k=1}^{K} \boldsymbol{z}_{ik} \log \boldsymbol{y}_{ik}$$

where $\boldsymbol{z}_i$ is a vectors that has all component equal to 0, except for the index c_i which is equal to 1

$$\boldsymbol{z}_i = [0 \dots 0, 1, 0 \dots 0] \ , \quad \boldsymbol{z}_{ik} = \begin{cases} 1 & \text{if } c_i = k \\ 0 & \text{otherwise} \end{cases}$$

Multiclass Logistic Regression

The log-likelihood can thus be expressed as

$$\ell(\boldsymbol{W}, \boldsymbol{b}) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \log \boldsymbol{y}_{ik}$$

The terms $\boldsymbol{y}_{ik}$ are, again,

$$\boldsymbol{y}_{ik} = \frac{e^{\boldsymbol{w}_k^T \boldsymbol{x}_i + b_k}}{\sum_j e^{\boldsymbol{w}_j^T \boldsymbol{x}_i + b_j}}$$

and represent the distribution for the class labels according to the Logistic Regression model

Multiclass Logistic Regression

As for the binary case, the expression

$$H(\boldsymbol{z}_i, \boldsymbol{y}_i) = -\sum_{k=1}^{K} z_{ik} \log \boldsymbol{y}_{ik}$$

represents the (multiclass) cross-entropy between the observed and predicted label distributions for sample $\boldsymbol{x}_i$

As for the binary case, we estimate $\boldsymbol{W}$ and $\boldsymbol{b}$ as to maximize the likelihood for the training labels

The ML solution is again the solution that minimizes the (average) cross-entropy:

$$\arg\max_{\boldsymbol{W},\boldsymbol{b}} \ell(\boldsymbol{W}, \boldsymbol{b}) = \arg\max_{\boldsymbol{W},\boldsymbol{b}} \left[-\sum_{i=1}^{n} H(\boldsymbol{z}_i, \boldsymbol{y}_i) \right] = \arg\min_{\boldsymbol{W},\boldsymbol{b}} \sum_{i=1}^{n} H(\boldsymbol{z}_i, \boldsymbol{y}_i)$$

Multiclass Logistic Regression

Compared to the binary case, the model is over-parametrized (i.e., we can add a constant vector to all terms $\boldsymbol{w}_i$ without changing the model)

In particular, for a 2-class problem, if we subtract $\boldsymbol{w}_2$ from both $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, we recover exactly the binary logistic regression objective.

Multiclass Logistic Regression

Finally, as for the binary class, we can cast the problem as a minimization of a loss function

We rewrite the objective in terms of class labels c as

$$
\begin{aligned}
J(\boldsymbol{W}, \boldsymbol{b}) &= -\sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \log \boldsymbol{y}_{ik} \\
&= -\sum_{i=1}^{n} \log \frac{e^{\boldsymbol{w}_{c_i}^T \boldsymbol{x}_i + b_{c_i}}}{\sum_{c'=1}^{K} e^{\boldsymbol{w}_{c'}^T \boldsymbol{x} + b_{c'}}} \\
&= \sum_{i=1}^{n} \left[\log \left(\sum_{c'=1}^{K} e^{\boldsymbol{w}_{c'}^T \boldsymbol{x} + b_{c'}} \right) - \boldsymbol{w}_{c_i}^T \boldsymbol{x}_i - b_{c_i} \right] \\
&= \sum_{i=1}^{n} l(\boldsymbol{x}_i, c_i, \boldsymbol{W}, \boldsymbol{b})
\end{aligned}
$$

l is also called softmax loss

Multiclass Logistic Regression

Again, we can add a regularization term to reduce over-fitting. We thus look for the minimizer of

$$R(\boldsymbol{W}, \boldsymbol{b}) = \Omega(\boldsymbol{W}) + \frac{1}{n} J(\boldsymbol{W}, \boldsymbol{b})$$

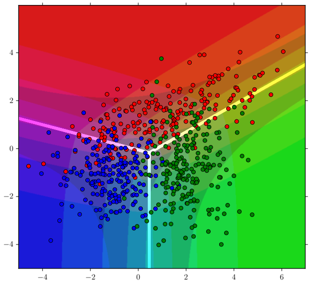
Different regularizers can be used, for example

$$\Omega(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N) = \frac{1}{2} \sum_i \|\boldsymbol{w}_i\|^2$$

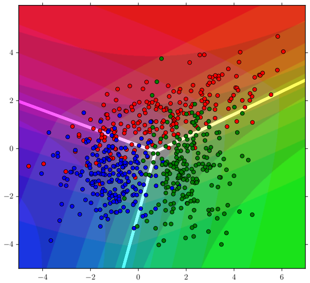
Again, we can replace the average cross-entropy with prior-weighted average cross entropy to account for priors that are different from the empirical training set prior

Multiclass Logistic Regression

Logistic Regression



Gaussian



Multiclass Logistic Regression

MNIST — Error rates for Logistic Regression

DimRed	$\lambda = 0$	$\lambda = 0.00001$	$\lambda = 0.001$	$\lambda = 0.1$	Tied Gau
RAW [768]	8.0%	$\boxed{7.4\%}$	7.9%	12.9%	—
PCA [50]	8.8%	8.8%	8.9%	13.3%	12.6 %
PCA [100]	7.8%	7.8%	8.2%	12.9%	12.3%
PCA+LDA [9]	10.9%	10.9%	11.0%	12.4%	12.3 %

The multiclass logistic regression performs better than the Gaussian model — indeed, the Gaussian assumption is not very accurate for the features we are considering. LogReg only assumes linear separation, but does not impose a distribution over the features

Again, regularization is important, especially when the feature space is large

Multiclass Logistic Regression

Linear logistic regression on MNIST performs better than our Tied-Covariance Gaussian classifier, however it's far worse than our non-linear Gaussian classifier

Remember that, for binary LR, we assumed linear separation surfaces

$$\log \frac{P(C = h_1|\boldsymbol{x})}{P(C = h_0|\boldsymbol{x})} = \boldsymbol{w}^T \boldsymbol{x} + b$$

which has the same form as the Gaussian classifier with tied covariances

For Gaussian classifier with non-tied covariances we have

$$\log \frac{P(C = h_1|\boldsymbol{x})}{P(C = h_0|\boldsymbol{x})} = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x} + c = s(\boldsymbol{x}, \boldsymbol{A}, \boldsymbol{b}, c)$$

The expression

$$s(\boldsymbol{x}, \boldsymbol{A}, \boldsymbol{b}, c) = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x} + c$$

is quadratic in $\boldsymbol{x}$, however it is linear in $\boldsymbol{A}$ and $\boldsymbol{b}$

Indeed, we can rewrite $s(\boldsymbol{x}, \boldsymbol{A}, \boldsymbol{b}, c)$ as

$$s(\boldsymbol{x}, \boldsymbol{A}, \boldsymbol{b}, c) = \langle \boldsymbol{x}\boldsymbol{x}^T, \boldsymbol{A} \rangle + \boldsymbol{b}^T \boldsymbol{x} + c$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product

$$\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \sum_i \sum_j \boldsymbol{A}_{ij} \boldsymbol{B}_{ij}$$

We can further express $\langle \boldsymbol{A}, \boldsymbol{x}\boldsymbol{x}^T \rangle$ as

$$\langle \boldsymbol{A}, \boldsymbol{x}\boldsymbol{x}^T \rangle = \mathrm{vec}(\boldsymbol{x}\boldsymbol{x}^T)^T \mathrm{vec}(\boldsymbol{A})$$

$\mathrm{vec}(\boldsymbol{M})$ is the operator that stacks the columns of matrix $\boldsymbol{M}$

Multiclass Logistic Regression

If we define

$$\phi(\boldsymbol{x}) = \begin{bmatrix} \mathrm{vec}(\boldsymbol{x}\boldsymbol{x}^T) \\ \boldsymbol{x} \end{bmatrix}$$

and

$$\boldsymbol{w} = \begin{bmatrix} \mathrm{vec}(\boldsymbol{A}) \\ \boldsymbol{b} \end{bmatrix}$$

then the class log-posterior ratio can be expressed as

$$s(\boldsymbol{x}, \boldsymbol{w}, c) = \boldsymbol{w}^T \phi(\boldsymbol{x}) + c$$

We can thus train a LR model using feature vectors $\phi(\boldsymbol{x})$ rather than $\boldsymbol{x}$

We will obtain a model that has linear separation surface in the space defined by the mapping ϕ

This space is also called expanded feature space

Multiclass Logistic Regression

The LR model (both binary and multiclass) allows computing linear separation rules for the transformed features $\phi(\boldsymbol{x})$

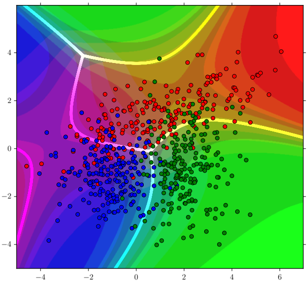
Since expressions $\boldsymbol{w}^T\phi(\boldsymbol{x}) + c$ correspond to quadratic forms in the original feature space, we are actually estimating quadratic separation surfaces in the original space

In general, we can consider a transformation $\phi(\boldsymbol{x})$ of our feature space such that our classes are (approximately) linearly separable in the expanded feature space

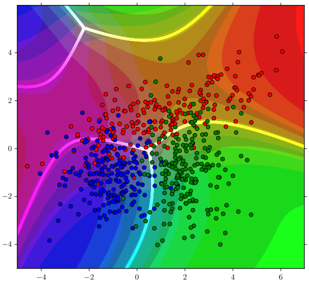
We have to pay attention that the dimensionality of the expanded feature space can grow very quickly — For example, polynomial expansions of degree d result in a feature space of dimensionality $O(M^d)$, where M is the dimensionality of $\boldsymbol{x}$.

Multiclass Logistic Regression

Logistic Regression



Gaussian



Multiclass Logistic Regression

MNIST — Average pairwise EER for LR with quadratic feature expansion

DimRed	$\lambda = 0$	$\lambda = 1e^{-5}$	$\lambda = 1e^{-3}$	$\lambda = 1e^{-1}$	Gaussian
PCA [50]	1.0%	1.0%	0.9%	1.5%	0.8%

MNIST — Multiclass error rates for LR with quadratic feature expansion

DimRed	$\lambda = 0$	$\lambda = 1e^{-5}$	$\lambda = 1e^{-3}$	$\lambda = 1e^{-1}$	Gaussian
PCA [50]	2.3%	1.9%	1.7%	3.1%	3.6%

In the next lectures we will see a different approach for non–linear classification: Support Vector Machines (SVM)

Alternatively, neural networks (not part of this course) allow jointly estimating a parametric transformation $\phi(\boldsymbol{x}, \boldsymbol{\Pi})$ and a classification rule $(\boldsymbol{w}, b)$ in the transformed space