A PROOF OF CONVERGENCE OF MULTI-CLASS LOGISTIC REGRESSION NETWORK

MAREK RYCHLIK
UNIVERSITY OF ARIZONA
DEPARTMENT OF MATHEMATICS, 617 N SANTA RITA RD, P.O. BOX 210089
TUCSON, AZ 85721-0089, USA

Abstract. This paper revisits the special type of a neural network known under two names. In the statistics and machine learning community it is known as a multi-class logistic neural network. In the neural network community, it is simply the soft-max layer. The importance is underscored by its role in deep learning: as the last layer, whose autput is actually the classification of the input patterns, such as images. Our exposition focuses on mathematically rigorous derivation of the key equation expressing the gradient. The fringe benefit of our approach is a fully vectorized expression, which is a basis of an efficient implementation. The second result of this paper is the positivity of the second derivative of the cross-entropy loss function as function of the weights. This result proves that optimization methods based on convexity may be used to train this network. As a corollary, we demonstrate that no L^2-regularizer is needed to guarantee convergence of gradient descent.

1. Notation and definitions

The multi-class logistic regression network is a neural network which takes an input vector $\mathbf{x} \in \mathbb{R}^D$ and produces an activation vector $\mathbf{a} \in \mathbb{R}^C$ by a linear transformation

$$\mathbf{a} = \mathbf{W}\,\mathbf{x},$$

where $\mathbf{W} = [w_{jk}]$ is an $C \times D$ matrix of weights. The vector space of such matrices will be denoted by $L\left(\mathbb{R}^D, \mathbb{R}^C\right)$ and identified with the space of linear transformations

$$\mathbf{W} : \mathbb{R}^D \to \mathbb{R}^C.$$

The activations are subsequently transformed by the *soft-max function* $\boldsymbol{\sigma} : \mathbb{R}^C \to \mathbb{R}^C$ given by the formula

$$\sigma_i(\mathbf{u}) = \frac{e^{u_i}}{\sum_{j=1}^{C} e^{u_j}}.$$

Thus,

$$\mathbf{y} = \boldsymbol{\sigma}\left(\mathbf{W}\mathbf{x}\right)$$

The number C is the number of classes into which the input vectors will be classified into. For a *training vector* $\mathbf{x}$ its classification is known and it is given by a target vector $\mathbf{t} \in \{0,1\}^C \subset \mathbb{R}^C$, where $t_k = 1$ iff vector $\mathbf{x}$ belongs to class k. Thus

$$\sum_{k=1}^{C} t_k = 1.$$

In order to train the network, we need the *training set* consisting of N sample input vectors

$$\mathbf{x}^{(n)} \in \mathbb{R}^D, \qquad n = 1, 2, \ldots, N$$

Date: September 21, 2018.

and N corresponding target vectors

$$\mathbf{t}^{(n)} \in \mathbb{R}^C.$$

The loss function to be minimized is the *cross-entropy* loss function, given by the formula:

$$L(\mathbf{W}) = -\sum_{n=1}^{N}\sum_{i=1}^{D} t_i^{(n)} \log y_i^{(n)}. \tag{1}$$

The problem of training is the problem of finding the optimal weight matrix $\widehat{\mathbf{W}}$ which minimizes $L(\mathbf{W})$:

$$\widehat{\mathbf{W}} = \operatorname*{arg\,min}_{\mathbf{W}\in L(\mathbb{R}^D,\mathbb{R}^C)} L(\mathbf{W}).$$

The natural question arises, whether the minimum exists and whether it is unique. It is known that the simplistic answer to this question is "no" because shifting the weights by a constant depending on n only does not change the value of L. More precisely, if

$$\tilde{\mathbf{W}} = \mathbf{W} + \mathbb{1}\,\mathbf{c}^\intercal$$

then $L(\tilde{\mathbf{W}}) = L(\mathbf{W})$, and, more strongly

$$\tilde{\mathbf{y}}^{(n)} = \mathbf{y}^{(n)}$$

where

$$\mathbf{y}^{(n)} = \boldsymbol{\sigma}\left(\mathbf{W}\,\mathbf{x}^{(n)}\right).$$
$$\tilde{\mathbf{y}}^{(n)} = \boldsymbol{\sigma}\left(\tilde{\mathbf{W}}\,\mathbf{x}^{(n)}\right).$$

This is the consequence of the following identity: for every $c \in \mathbb{R}$

$$\boldsymbol{\sigma}(\mathbf{u} + c\,\mathbb{1}) = \boldsymbol{\sigma}(\mathbf{u}).$$

This brief paper answers this question in the most satisfactory fashion, giving sufficient conditions for L to be a strictly convex function on the subspace of those weight matrices $\mathbf{W}$ for which the sum of every column is 0, i.e.

$$\mathbb{1}^\intercal\,\mathbf{W} = 0.$$

This condition guarantees both the existence and uniqueness of the solution, and convergence of optimization algorithms which depend on strict convexity.

2. Related reading

Many related techniques are described in [2, 3]. The softmax function use for pattern recognition is also discussed in [1], on pp. 215 and pp. 238–245.

3. The critical points of L

In this section we summarize known conditions for $\mathbf{W}$ to be a critical points of L. For simplicity of notation we assume $N = 1$, which allows us to avoid the superscript (n). The summation over all samples yields the result for the entire training set. The manner in which we derive the formulas may differ from a typical derivation in machine learning texts, in that it relies upon the Chain Rule for *Fréchet derivatives*, rather than calculation of partials with respect of individual weights w_{jk}.[1]

[1]It should be remembered that Fréchet derivatives are closely related to *Jacobi matrices*, but they are **not the same**, as it will be quite apparent in our calculations. Fréchet derivative is a linear transformation, and upon the choice of a basis, or, in the case of $\mathbb{R}^n$, when the standard basis is used, this linear transformation is identified with the Jacobi matrix.

With this approach, we automatically derive *vectorized* formulas for the gradient of L, which subsequently leads to very efficient implementation of the training algorithm, as all critical operations are simply matrix products.

We also allow the target vector $\mathbf{t} \in \mathbb{R}^C$, without necessarily requiring that $t_k \in \{0, 1\}$, but we still require

$$\sum_{k=1}^{C} t_k = 1.$$

From the point of view of applications, this generalization is useful when the classification of the inputs is ambiguous. For instance, we can use several humans to classify the inputs, in which case the humans may classify the inputs differently. Then we could assign to t_k the fraction of humans who assign the input to class k.

We use the following representation of the loss function L:

$$L(\mathbf{W}) = Q_{\mathbf{t}}(-\log(\boldsymbol{\sigma}(P_{\mathbf{x}}(\mathbf{W}))))$$

where

$$P_{\mathbf{x}}(\mathbf{W}) = \mathbf{W}\,\mathbf{x},$$
$$Q_{\mathbf{t}}(\mathbf{z}) = \mathbf{t}^{\intercal}\,\mathbf{z}.$$

is a vector-valued *linear* operator *on matrices*, consisting in multiplying a matrix by $\mathbf{x}$ on the right. Thus

$$L = Q_{\mathbf{t}} \circ ((-\log) \circ \boldsymbol{\sigma}) \circ P_{\mathbf{x}},$$

which is a composition involving linear operators $Q_{\mathbf{t}}$ and $P_{\mathbf{x}}$ and two non-linear transformations, log (coordinatewise) and $\boldsymbol{\sigma}$. Let $\boldsymbol{\rho} = (-\log) \circ \boldsymbol{\sigma}$. It will be beneficial to think of L as

$$L = Q_{\mathbf{t}} \circ \boldsymbol{\rho} \circ P_{\mathbf{x}}$$

which is a composition with only one non-linear term.

Remark 1. *This form is particularly useful for calculating higher derivatives of L, specifically, the Hessian of L.*

The Chain Rule yields the Fréchet derivative of the composition:

$$DL = DQ_{\mathbf{t}}\, D\boldsymbol{\rho}\, DP_{\mathbf{x}}$$

where the intermediate Fréchet derivatives are evaluated at respective intermediate values of the composition. The more detailed version of the formula is:

$$DL = (DQ_{\mathbf{t}} \circ \boldsymbol{\rho} \circ P_{\mathbf{x}})\ (D\boldsymbol{\rho} \circ P_{\mathbf{x}})\ DP_{\mathbf{x}}$$

It should be noted that for a function $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^m$,

$$D\mathbf{F} : \mathbb{R}^n \to L(\mathbb{R}^n, \mathbb{R}^m),$$

i.e. it is an operator-valued function on $\mathbb{R}^n$.

The Fréchet derivative of a linear transformation is the transformation itself. Thus

$$DQ_{\mathbf{t}} = Q_{\mathbf{t}} \tag{2}$$

$$DP_{\mathbf{x}} = P_{\mathbf{x}} \tag{3}$$

regardless of the argument.

We state one more useful formula. For any vectorized scalar function, evaluated elementwise, i.e. function $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ given by:

$$\mathbf{F}(\mathbf{x}) = (f(x_1), f(x_2), \ldots, f(x_n))$$

is a diagonal matrix:

$$D\mathbf{F}(\mathbf{x}) = \begin{bmatrix} f'(x_1) & 0 & \dots & 0 \\ 0 & f'(x_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & f'(x_n) \end{bmatrix} = \operatorname{diag}(f'(\mathbf{x})).$$

where diag is a MATLAB-like operator converting a vector to a diagonal matrix. We make an observation that diag itself is a linear operator from vectors to matrices.

In summary,

$$DL(\mathbf{W}) = Q_{\mathbf{t}}\, D\rho(\mathbf{a})\, P_{\mathbf{x}} \tag{4}$$

Let us compute the derivative of ρ. Firstly,

$$\rho(\mathbf{u}) = -\mathbf{u} + \left(\log\left(\sum_{k=1}^{C} e^{u_i}\right)\right)\, \mathbb{1}$$

where $\mathbb{1} = (1, 1, \dots, 1)$ is a (column) vector of 1's. This implies easily, using a combination of techniques already mentioned:

$$\begin{aligned} D\boldsymbol{\rho}(\mathbf{u})\mathbf{h} &= -\mathbf{h} + \left(\frac{1}{\sum_{k=1}^{C} e^{u_i}}\, (e^{\mathbf{u}})^{\intercal}\ \mathbf{h}\right) \mathbb{1} \\ &= -\mathbf{h} + (\boldsymbol{\sigma}(\mathbf{u})^{\intercal}\mathbf{h})\mathbb{1} = -\mathbf{h} + \mathbb{1}\,(\boldsymbol{\sigma}(\mathbf{u})^{\intercal}\mathbf{h}) = (-I + \mathbb{1}\,\boldsymbol{\sigma}(\mathbf{u})^{\intercal})\,\mathbf{h} \end{aligned}$$

where I is the $C \times C$ identity matrix. Thus, in short,

$$D\boldsymbol{\rho}(\mathbf{u}) = -I + \mathbb{1}\,\boldsymbol{\sigma}(\mathbf{u})^{\intercal}.$$

Finally, carfully looking at the formula (4) and equating matrix product with composition of linear transformations (this convention should be familiar from linear algebra), we obtain

$$\begin{aligned} DL(\mathbf{W})\mathbf{V} &= \mathbf{t}^{\intercal}\,(-I + \mathbb{1}\,\boldsymbol{\sigma}(\mathbf{W}\,\mathbf{x})^{\intercal})\,\mathbf{V}\,\mathbf{x} \\ &= -\mathbf{t}^{\intercal}\,\mathbf{V}\,\mathbf{x} + (\mathbf{t}^{\intercal}\,\mathbb{1})\,\mathbf{y}^{\intercal}\,\mathbf{V}\,\mathbf{x} \\ &= -\mathbf{t}^{\intercal}\,\mathbf{V}\mathbf{x} + \mathbf{y}^{\intercal}\,\mathbf{V}\,\mathbf{x} \\ &= -(\mathbf{t} - \mathbf{y})^{\intercal}\,\mathbf{V}\mathbf{x}. \end{aligned}$$

We note that we used $\mathbf{t}^{\intercal}\mathbb{1} = 1$ because $\sum_{k=1}^{C} t_k = 1$ was assumed.

This formula generalizes easily to $N \geq 1$, when L is given by (1):

$$DL(\mathbf{W})\mathbf{V} = -\sum_{n=1}^{N} \left(\mathbf{t}^{(n)} - \mathbf{y}^{(n)}\right)^{\intercal}\,\mathbf{V}\mathbf{x}^{(n)}.$$

In order to use methods such as gradient descent, we need to find the gradient $\nabla L(\mathbf{W})$ from the formula for $DL(\mathbf{W})$. We should note that the gradient belongs to the same vector space as the argument $\mathbf{W}$, while $DL(\mathbf{W})$ is a functional on the same space. In our situation:

$$\nabla L(\mathbf{W}) \in L(\mathbb{R}^D, \mathbb{R}^C) \tag{5}$$

$$DL(\mathbf{W}) \in L(L(\mathbb{R}^D, \mathbb{R}^C), \mathbb{R}) = L(\mathbb{R}^C, \mathbb{R}^C)^* \tag{6}$$

where the notation $X^* = L(X, \mathbb{R})$ defines the *dual space* of the vector space X. Thus, an expression $\mathbf{W} + \eta\nabla L(\mathbf{W})$ can be evaluated for $\eta \in \mathbb{R}$, but $\mathbf{W} + \eta\, DL(\mathbf{W})$ makes no sense.

We recall that the notion of gradient depends on the inner product in the underlying vector space. In our case, it is the space of weight matrices $\mathbf{W}$, i.e. $L(\mathbb{R}^D, \mathbb{R}^C)$. We assume the most simple form of the inner product: the Frobenius inner product:

$$\langle \mathbf{U}, \mathbf{V} \rangle = \sum_{j=1}^{C} \sum_{k=1}^{D} U_{jk} V_{jk} = \operatorname{tr} \mathbf{U}^T \mathbf{V}. \tag{7}$$

We then have the definition of gradient by duality:

$$DL(\mathbf{W})\mathbf{V} = \langle \nabla L(\mathbf{W}), \mathbf{V} \rangle. \tag{8}$$

for all matrices $\mathbf{V}$. Then, for $N = 1$ we look for

$$-(\mathbf{t} - \mathbf{y})^\intercal \mathbf{V}\, \mathbf{x} = \operatorname{tr} \nabla L(\mathbf{W})^\intercal \mathbf{V}$$

Using the identity $\operatorname{tr}(\mathbf{A}\,\mathbf{B}) = \operatorname{tr}(\mathbf{B}\mathbf{A})$, and for vectors $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{a}^\intercal \mathbf{b} = \operatorname{tr}(\mathbf{b}\,\mathbf{a}^\intercal)$, we obtain:

$$-\operatorname{tr} \{\mathbf{V}\, \mathbf{x}(\mathbf{t} - \mathbf{y})^\intercal\} = \operatorname{tr} \{\mathbf{V}(\nabla L(\mathbf{W}))^\intercal\}$$

or, more invariantly,

$$-\langle \mathbf{V}, (\mathbf{t} - \mathbf{y})\mathbf{x}^\intercal \rangle = \langle \mathbf{V}, \nabla L(\mathbf{W}) \rangle.$$

Because $\mathbf{V}$ is arbitrary:

$$\nabla L(\mathbf{W}) = -(\mathbf{t} - \mathbf{y})\mathbf{x}^\intercal.$$

For arbitrary N we obtain the following gradient formula:

$$\nabla L(\mathbf{W}) = -\sum_{n=1}^{N} \left(\mathbf{t}^{(n)} - \mathbf{y}^{(n)}\right) \left(\mathbf{x}^{(n)}\right)^\intercal = -(\mathbf{T} - \mathbf{Y})\, \mathbf{X}^\intercal. \tag{9}$$

where

$$\begin{aligned} \mathbf{T} &= \begin{bmatrix} \mathbf{t}^{(1)} & \mathbf{t}^{(2)} & \dots & \mathbf{t}^{(N)} \end{bmatrix}, \\ \mathbf{Y} &= \begin{bmatrix} \mathbf{y}^{(1)} & \mathbf{y}^{(2)} & \dots & \mathbf{y}^{(N)} \end{bmatrix}, \\ \mathbf{X} &= \begin{bmatrix} \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(N)} \end{bmatrix}, \end{aligned}$$

are matrices containing the elements of the training data in their columns. Thus, calculating the gradient can be expressed through simple matrix arithmetic, which leads to very efficient implementations of the gradient method.

Corollary 1 (Characterization of critical points)**.** *A weight matrix* $\mathbf{W}$ *is a critical point of* L *iff the sample uncentered covariance matrix of the errors*

$$\mathbf{e}^{(n)} = \mathbf{t}^{(n)} - \mathbf{y}^{(n)}$$

and of the input vectors $\mathbf{x}^{(n)}$ *is zero, where by definition the sample uncentered covariance matrix is the right-hand side of* (9)*.*

Let us address the matter of uniqueness of the critical point. If we have two critical points $\mathbf{W}$ and $\tilde{\mathbf{W}}$, the corresponding matrices $\mathbf{Y}$ and $\tilde{\mathbf{Y}}$ satisfy

$$0 = -(\mathbf{T} - \mathbf{Y})\, \mathbf{X} = -(\mathbf{T} - \tilde{\mathbf{Y}})\, \mathbf{X}^\intercal$$

Thus

$$0 = (\mathbf{Y} - \tilde{\mathbf{Y}})\mathbf{X}^\intercal$$

Let $\Delta\mathbf{Y} = \mathbf{Y} - \tilde{\mathbf{Y}}$. Thus

$$\Delta\mathbf{Y}\, \mathbf{X}^\intercal = 0.$$

Hence

$$\mathbf{X}\Delta\mathbf{Y}^\intercal = 0.$$

Furthermore, every row of $\mathbf{Y}$ is in the nullspace of $\mathbf{X}$. The matrix $\mathbf{X}$ has the elements of the training set laid out in columns and it has N columns. If the rank of $\mathbf{X}$ is maximal (equal to D, assuming $N \gg D$) then the nullspace is trivial. Thus $\mathbf{Y} = \tilde{\mathbf{Y}}$. Let $\mathbf{y} = \mathbf{y}^{(n)}$ and $\tilde{\mathbf{y}} = \tilde{\mathbf{y}}^{(n)}$ for some n. Then $\mathbf{y} = \tilde{\mathbf{y}}$. Let $\mathbf{w}_i$ and $\tilde{\mathbf{w}}_i$ be the i-th rows of $\mathbf{W}$ and $\tilde{\mathbf{W}}$. Let $\mathbf{x} = \mathbf{x}^{(n)}$ be the corresponding element of the training set. Then

$$\frac{\exp(\mathbf{w}_i^\intercal \mathbf{x})}{\exp(\tilde{\mathbf{w}}_i^\intercal \mathbf{x})} = \exp(\Delta \mathbf{w}_i^\intercal \mathbf{x})$$

is independent of i, and equals

$$\frac{\sum_{j=1}^{C} \exp(\tilde{\mathbf{w}}_j^\intercal \mathbf{x})}{\sum_{j=1}^{C} \exp(\mathbf{w}_j^\intercal \mathbf{x})}.$$

Hence,

$$\Delta \mathbf{w}_i^\intercal \, \mathbf{x} = c$$

where $c = c_n$ is a scalar depending on the element of the training set. Therefore

$$\Delta \mathbf{W} \, \mathbf{x} = c \, \mathbb{1}.$$

Combining all training set elements,

$$\Delta \mathbf{W} \, \mathbf{X} = \mathbb{1} \, \mathbf{c}^\intercal$$

where $\mathbf{c} = (c_1, c_2, \ldots, c_N)$.

Let $\tilde{\mathbf{X}}$ be a $D \times D$ non-singular submatrix of $\mathbf{X}$ and let $\tilde{\mathbf{c}}$ be the corresponding vector of coefficients c_n. Then

$$\Delta \mathbf{W} \, \tilde{\mathbf{X}} = \mathbb{1} \, \tilde{\mathbf{c}}^\intercal$$

and

$$\Delta \mathbf{W} = \mathbb{1} \, \tilde{\mathbf{c}}^\intercal \, \tilde{\mathbf{X}}^{-1} = \mathbb{1} \, \tilde{\tilde{\mathbf{c}}}^\intercal.$$

Thus, all columns of $\Delta \mathbf{W}$ must be multiples of $\mathbb{1}$, i.e. the weights $\tilde{\mathbf{W}}$ are obtained from $\mathbf{W}$ by a shift depending only on the element of the training set. Such a shift does not change the activations $\mathbf{y}^{(n)}$ and thus produces the same value of the cross-entropy loss function. Thus, the weights differing by such a shift are equivalent in all respects.

One way to choose the matrix $\mathbf{W}$ is to shift every column by subtracting the mean, which will have the effect of choosing the equivalent weight matrix with miniumum Frobenius norm:

$$\sum_{j=1}^{C} \sum_{k=1}^{D} w_{jk}^2 = \min.$$

A practical application of the above considerations is that we can subtract the mean from columns as part of the iteration process, when implementing an optimization method, such as gradient descent, which should help with stability of the method.

4. The second derivative and the Hessian

In this section we consider the second (Fréchet) derivative of the loss function which is closely related to the Hessian. We recall the expression:

$$L = Q_{\mathbf{t}} \circ \boldsymbol{\rho} \circ P_{\mathbf{x}}.$$

In view of linearity of the $Q_{\mathbf{t}}$ and $P_{\mathbf{x}}$, we have:

$$D^2 L(\mathbf{W})(\mathbf{U}, \mathbf{V}) = Q_{\mathbf{t}} \, D^2 \boldsymbol{\rho}(\mathbf{a})(P_{\mathbf{x}} \mathbf{U}, P_{\mathbf{x}} \mathbf{V})$$

where $\mathbf{a} = P_{\mathbf{x}}(W) = \mathbf{W} \, \mathbf{x}$. We recall that the second derivative $D^2 L(\mathbf{W})$ is a bi-linear, symmetric function on pairs of vectors $\mathbf{U}$, $\mathbf{W}$, which in our case are also $C \times D$ matrices.

We also have the formula for $D\boldsymbol{\rho}$:

$$D\boldsymbol{\rho}(\mathbf{u}) = -I + \mathbb{1}\,\boldsymbol{\sigma}(\mathbf{u})^\intercal.$$

Differentiating again, we obtain

$$D^2\boldsymbol{\rho}(\mathbf{a})\,(\mathbf{g},\mathbf{h}) = \mathbb{1}\,(D\boldsymbol{\sigma}(\mathbf{a})\,\mathbf{g})^\intercal\mathbf{h} = \mathbb{1}\,\mathbf{g}^\intercal D\boldsymbol{\sigma}(\mathbf{a})^\intercal\,\mathbf{h}.$$

Therefore,

$$Q_{\mathbf{t}}D^2\boldsymbol{\rho}(\mathbf{a})\,(\mathbf{g},\mathbf{h}) = (\mathbf{t}^\intercal\mathbb{1})\mathbf{g}^\intercal D\boldsymbol{\sigma}(\mathbf{a})^\intercal\,\mathbf{h} = \mathbf{g}^\intercal D\boldsymbol{\sigma}(\mathbf{a})^\intercal\,\mathbf{h}$$

Hence,

$$\begin{aligned} D^2L(\mathbf{W})(\mathbf{U},\mathbf{V}) &= (P_{\mathbf{x}}\mathbf{U})^\intercal\,D\boldsymbol{\sigma}(\mathbf{a})^\intercal\,(P_{\mathbf{x}}\mathbf{V}) \\ &= (\mathbf{U}\mathbf{x})^\intercal D\boldsymbol{\sigma}(\mathbf{a})^\intercal(\mathbf{V}\mathbf{x}) \\ &= \mathbf{x}^\intercal\,\mathbf{U}^\intercal\,D\boldsymbol{\sigma}(\mathbf{a})^\intercal\,\mathbf{V}\,\mathbf{x}. \end{aligned}$$

We also find

$$\begin{aligned} D\boldsymbol{\sigma}(\mathbf{a}) &= D\exp(-\boldsymbol{\rho}(\mathbf{a})) \\ &= -\mathrm{diag}(\exp(-\boldsymbol{\rho}(\mathbf{a})))D\boldsymbol{\rho}(\mathbf{a}) \\ &= -\mathrm{diag}(\boldsymbol{\sigma}(\mathbf{a}))(-I + \mathbb{1}\,\boldsymbol{\sigma}(\mathbf{a})^\intercal) \\ &= \mathrm{diag}(\boldsymbol{\sigma}(\mathbf{a})) - \mathrm{diag}(\boldsymbol{\sigma}(\mathbf{a}))\mathbb{1}\,\boldsymbol{\sigma}(\mathbf{a})^\intercal. \end{aligned}$$

Hence,

$$D\boldsymbol{\sigma}(\mathbf{a})^\intercal = \mathrm{diag}(\boldsymbol{\sigma}(\mathbf{a})) - \boldsymbol{\sigma}(\mathbf{a})\,\mathbb{1}^\intercal\,\mathrm{diag}(\boldsymbol{\sigma}(\mathbf{a})).$$

Therefore, taking into account that $\mathrm{diag}(\mathbf{s})\,\mathbb{1} = \mathbf{s}$ and $\mathbb{1}^\intercal\,\mathrm{diag}(\mathbf{s}) = \mathbf{s}^\intercal$, we obtain:

$$D\boldsymbol{\sigma}(\mathbf{a})^\intercal = \mathrm{diag}(\boldsymbol{\sigma}(\mathbf{a})) - \boldsymbol{\sigma}(\mathbf{a})\,\boldsymbol{\sigma}(\mathbf{a})^\intercal.$$

$$\begin{aligned} D^2L(\mathbf{W})(\mathbf{U},\mathbf{V}) &= \mathbf{x}^\intercal\mathbf{U}^\intercal\mathrm{diag}(\boldsymbol{\sigma}(\mathbf{a}))\mathbf{V}\mathbf{x} - \mathbf{x}^\intercal\mathbf{U}^\intercal\boldsymbol{\sigma}(\mathbf{a})\,\boldsymbol{\sigma}(\mathbf{a})^\intercal\mathbf{V}\mathbf{x} \\ &= \mathbf{x}^\intercal\mathbf{U}^\intercal(\mathrm{diag}(\boldsymbol{\sigma}(\mathbf{a})) - \boldsymbol{\sigma}(\mathbf{a})\,\boldsymbol{\sigma}(\mathbf{a})^\intercal)\mathbf{V}\mathbf{x} \\ &= \mathbf{x}^\intercal\mathbf{U}^\intercal(\mathrm{diag}(\mathbf{y}) - \mathbf{y}\,\mathbf{y}^\intercal)\mathbf{V}\mathbf{x}. \end{aligned}$$

This bi-linear form is positive definite, as

$$D^2L(\mathbf{W})(\mathbf{U},\mathbf{U}) = (\mathbf{U}\mathbf{x})^\intercal(\mathrm{diag}(\mathbf{y}) - \mathbf{y}\,\mathbf{y}^\intercal)(\mathbf{U}\mathbf{x}) \geq 0.$$

Indeed, the matrix $\mathrm{diag}(\mathbf{y}) - \mathbf{y}\mathbf{y}^\intercal$ is symmetric and thus has real spectrum. It suffices to show that the spectrum is non-negative.

Proposition 1. *Let $\mathbf{y} \in \mathbb{R}^C$ be a vector satisfying*

(1) $y_i > 0$*;*
(2) $\sum_{i=1}^C y_i = 1$.

Then the eigenvalues of the symmetric matrix

$$\mathrm{diag}(\mathbf{y}) - \mathbf{y}\,\mathbf{y}^\intercal$$

are all non-negative. Furthermore, the only eigenvector with eigenvalue 0 is $\mathbb{1}$ up to a scalar factor.

Proof. **Method 1:** Let λ be an eigenvalue and $\mathbf{z}$ be an eigenvector for eigenvalue λ; thus

$$\mathrm{diag}(\mathbf{y})\,\mathbf{z} - \mathbf{y}\,(\mathbf{y}^\intercal\mathbf{z}) = \lambda\,\mathbf{z}.$$

Therefore,

$$\begin{aligned} y_i\,z_i - y_i\,\langle\mathbf{y},\mathbf{z}\rangle &= \lambda\,z_i. \\ (y_i - \lambda)z_i &= y_i\,\langle\mathbf{y},\mathbf{z}\rangle \end{aligned}$$

We know that $0 < y_i < 1$. Therefore, unless $0 < \lambda < 1$, $y_i \neq \lambda$, and

$$z_i = \frac{y_i}{y_i - \lambda}\langle \mathbf{y}, \mathbf{z}\rangle.$$

Since $\mathbf{z} \neq 0$, we assume WLOG that $\langle \mathbf{y}, \mathbf{z}\rangle \neq 0$. Hence

$$\langle \mathbf{y}, \mathbf{z}\rangle = \sum_{i=1}^{C} \frac{y_i^2}{y_i - \lambda}\langle \mathbf{y}, \mathbf{z}\rangle.$$

and

$$\sum_{i=1}^{C} \frac{y_i^2}{y_i - \lambda} = 1.$$

We need to show that all roots λ of this equation are non-negative. Indeed, if $\lambda < 0$ then

$$\sum_{i=1}^{C} \frac{y_i^2}{y_i - \lambda} < \sum_{i=1}^{C} \frac{y_i^2}{y_i} = \sum_{i=1}^{C} y_i = 1$$

which is a contradition. Thus $\lambda \geq 0$.

It remains to see that the eigenvector $\mathbf{z}$ for the eigenvalue $\lambda = 0$ is $\mathbf{z} = \mathbb{1}$ up to a multiplicative constant. Indeed, if $\lambda = 0$ then for $i = 1, 2, \ldots, C$:

$$y_i\, z_i = y_i\, \langle \mathbf{y}, \mathbf{z}\rangle.$$

Dividing by $y_i > 0$, we obtain:

$$z_i = \langle \mathbf{y}, \mathbf{z}\rangle.$$

Hence z_i is independent of i, i.e. proportional to $\mathbb{1}$.

Method 2: (Gershgorin Circle Theorem) The matrix in question has diagonal entry $y_i - y_i^2$ at position i, and the off-diagonal entries in row i are $y_i y_j$, $j = 1, 2, \ldots, C$, $j \neq i$. Hence, for every eigenvalue λ there esists i such that:

$$|\lambda - (y_i - y_i^2)| \leq \sum_{\substack{j=1 \\ j \neq i}}^{C} y_i y_j.$$

In particular

$$\lambda \geq (y_i - y_i^2) - \sum_{\substack{j=1 \\ j \neq i}}^{C} y_i y_j = y_i - y_i \sum_{j=1}^{C} y_j = y_i - y_i = 0.$$

Method 3: Let $\mathbf{M} = \mathrm{diag}(\sqrt{\mathbf{y}})$. Obviously this is a symmetric diagonal matrix. Then we have the following factorization:

$$\mathrm{diag}(\mathbf{y}) - \mathbf{y}\,\mathbf{y}^\intercal = \mathbf{M}^\intercal (I - \mathbf{u}\mathbf{u}^\intercal)\mathbf{M}$$

where $\mathbf{u} = M^{-1}\mathbf{y} = \sqrt{\mathbf{y}}$. Clearly, $\sum_{i=1}^{C} u_i^2 = \sum_{i=1}^{C} y_i = 1$. Thus $\mathbf{u}$ is a unit vector. The matrix $I - \mathbf{u}\,\mathbf{u}^\intercal$ is non-negative definite as it is an orthogonal projection on the hyperplane normal to $\mathbf{u}$, and therefore its eigenvalues are 0 (multiplicity 1) and 1 (multiplicity $C-1$). Moreover, the eigenvector with eigenvalue 0 is $\mathbf{u}$. Hence, the eigenvector $\mathbf{z}$ with eigenvalue 0 for $\mathrm{diag}(\mathbf{y}) - \mathbf{y}\,\mathbf{y}^\intercal$ satisfies (up to a scalar multiple):

$$\mathbf{M}\,\mathbf{z} = \mathbf{u}.$$

and $\mathbf{z} = \mathbf{M}^{-1}\,\mathbf{u} = \mathbb{1}$. □

Let us consider arbitrary N (the size of the training set). We then have

$$D^2L(\mathbf{W})(\mathbf{U},\mathbf{U}) = \sum_{n=1}^{N} \left(\mathbf{U}\mathbf{x}^{(n)}\right)^\intercal \left(\operatorname{diag}(\mathbf{y}^{(n)} - \mathbf{y}^{(n)} \left(\mathbf{y}^{(n)}\right)^\intercal\right) \left(\mathbf{U}\mathbf{x}^{(n)}\right) \geq 0.$$

The only way to get a value of 0 is for every summand to be 0, i.e. to have for $n = 1, 2, \ldots, N$:

$$\mathbf{U}\,\mathbf{x}^{(n)} = c_n\,\mathbb{1}$$

for some scalars c_n, $n = 1, 2, \ldots, N$. These equations can also be written as a single matrix identity:

$$\mathbf{U}\,\mathbf{X} = \mathbb{1}\,\mathbf{c}^\intercal \tag{10}$$

where $\mathbf{c} = (c_1, c_2, \ldots, c_N)$.

The degeneracy condition (10) is rather hard to fulfill for a randomly chosen training set. If the coordinates of $\mathbf{x}^{(n)}$ are not linearly dependent as random variables then $\mathbf{X}$ has maximum rank D with probability 1 if the variables are continuous, or with probability asymptotically converging to 1 if the variables are discrete (or mixed). Let $\tilde{\mathbf{X}}$ be a $D \times D$ non-singular submatrix of $\mathbf{X}$. Then

$$\mathbf{U} = \mathbb{1}\,\mathbf{c}^\intercal\,\tilde{\mathbf{X}}^{-1} = \mathbb{1}\,\tilde{\mathbf{c}}^\intercal.$$

Thus, all columns of $\mathbf{U}$ must be multiples of $\mathbb{1}$. We are already familiar with this condition. If we restrict L to only matrices $\mathbf{W}$ for which the mean of every column is 0 then the second derivative $D^2L(\mathbf{W})$ is strictly-positive definite, thus ensuring convergence of optimization methods, such as gradient descent.

Corollary 2 (On convergence and regularization)**.** *The loss function L for the multi-class logistic regression is strictly convex on the set of weight matrices $\mathbf{W}$ with column mean zero, as long as the training set spans the vector space $\mathbb{R}^D$, or, equivalently, when* $\operatorname{rank}\mathbf{X} = D$*. Therefore, it is possible to find the optimal weight without using a regularizer.*

5. A sample training algorithm

Algorithm 1 contains the basic training loop for the logistic regression network we described in previous sections. Many modifications are possible. For instance, one can implement variable training rate, utilizing, for example, the Barzilai-Bowdain update rule. Also, we can break out of the loop if no progress is made. We could evaluate the loss function and see if it decays, as a test of progress, etc.

It should be noted tat

$$\mathbb{1}^\intercal\,\mathbf{T} = \mathbb{1}^\intercal\,\mathbf{Y} = 1$$

and therefore

$$\mathbb{1}^\intercal\,(\mathbf{T} - \mathbf{Y})^\intercal = 0$$

Hence, in view of $\nabla L(\mathbf{W}) = (\mathbf{T} - \mathbf{Y})\,\mathbf{X}^\intercal$,

$$\mathbb{1}^\intercal\,\nabla L(\mathbf{W}) = 0$$

and the condition

$$\mathbb{1}^\intercal\,\mathbf{W} = 0$$

is maintained during the iteration process, at least if round-off error is ignored. However, we may want to subtract the mean from each column of $\mathbf{W}$ every so often to prevent round-off error acting as diffusion in the weight space, which clearly prevents convergence.

Algorithm 1 The algorithm implements training of the logistic regression network by gradient descent method.

Require:
$\mathbf{X}$ is an $D \times N$ matrix of rank D, containing the N training vectors as columns;
$\mathbf{W}$ is a $C \times D$ weight matrix initialized at random;
$\eta \in \mathbb{R}$ is the learning rate;
$NumEpochs$ is the number of epochs;
Ensure:
$\mathbf{W}$ approximates the optimal weights minimizing the cross-entropy loss function.
for $epoch = 1, 2, \ldots, NumEpochs$ **do**
  $\mathbf{A} \leftarrow \mathbf{W} \cdot \mathbf{X}$ ▷ Compute activations.
  $\mathbf{Y} \leftarrow \textsc{softmax}(\mathbf{A})$ ▷ Compute softmax activity.
  $\mathbf{E} \leftarrow \mathbf{T} - \mathbf{Y}$ ▷ Compute errrors.
  $\boldsymbol{\nabla} L \leftarrow -\mathbf{E} \cdot \mathbf{X}^{\intercal}$ ▷ Find the gradient.
  $\mathbf{W} \leftarrow \mathbf{W} - \eta \cdot \boldsymbol{\nabla} L$ ▷ Update weights.
end for

References

[1] Christopher M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, Inc. New York, NY, USA, 1996.

[2] Logistic regression. `https://en.wikipedia.org/wiki/Logistic_regression`.

[3] Multinomial logistic regression. `https://en.wikipedia.org/wiki/Multinomial_logistic_regression`.