Inclass 08: Classification

[SCS4049] Machine Learning and Data Science

Seongsik Park (s.park@dgu.edu)

AI Department, Dongguk University

Contents

Contents

MNIST dataset

- 70,000 images of 28×28 handwritten digits from 0 to 9
- “Hello, World!” of machine learning

Figure 3-1. A few digits from the MNIST dataset

Classification

The goal in classification

- Take an input vector $\mathbf{x}$
- Assign it to one-of-K discrete class $\mathcal{C}_k$ where $k = 1, 2, \cdots, K$
- Assign each input to one and only one class (disjoint)

The input space divided into decision region, whose boundaries are called **decision boundaries** or **decision surfaces**.

- (D-1)-dimensional hyperplanes within the D-dimensional input space
- Linearly separable dataset that can be separated exactly by linear decision surface

Classification

Binary representation $\mathbf{t}$

- Two-class problem, $t \in \{0, 1\}$ ($t = 1$ represents $\mathcal{C}_1$)
- $K > 2$ classes, $\mathbf{t} \in \{0, 1\}^K$ and $\sum_k t_k = 1$
- t_k is probability of $\mathcal{C}_k$

Approaches

- Direct assign (discriminant function)
- Model $p(\mathcal{C}_k|\mathbf{x})$ with class-conditional distribution $p(\mathbf{x}|\mathcal{C}_k)$ and prior distribution $p(\mathcal{C}_k)$ (generative model)
- Model $p(\mathcal{C}_k|\mathbf{x})$ directly (parametric model)

Classification

Generalized linear model

- $y(\mathbf{x}) = f(\mathbf{w}^T\mathbf{x} + w_0)$
- Activation function $f : \Re \to (0, 1)$
- Even if the function $f(\cdot)$ is nonlinear, the decision surfaces $\mathbf{w}^T\mathbf{x} + w_0 = const$ are linear functions of $\mathbf{x}$

Confusion matrix

Confusion matrix (contingency matrix)

		Predicted	
		Negative	Positive
Ground truth	Negative	**T**rue **N**egative	**F**alse **P**ositive (Type I Error)
	Positive	**F**alse **N**egative (Type II Error)	**T**rue **P**ositive

True positive rate (TPR): $\mathrm{TPR} = \dfrac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$ (recall, sensitivity, hit rate)

Positive predictive value (PPV): $\mathrm{PPV} = \dfrac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$ (precision)

Accuracy (ACC): $\mathrm{ACC} = \dfrac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FN} + \mathrm{TN} + \mathrm{FP}}$

True negative rate (TNR): $\mathrm{TNR} = \dfrac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$ (specificity)

False negative rate (FNR): $\mathrm{FNR} = 1 - \mathrm{TPR}$

False positive rate (FPR): $\mathrm{FPR} = 1 - \mathrm{TNR}$

Contents

Contents

Linear discriminant function

Two classes

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

$$\begin{aligned} \mathbf{x} \text{ is assigned to class } \mathcal{C}_1 \quad & \text{if } y(\mathbf{x}) \geq 0 \\ \text{to class } \mathcal{C}_2 \quad & \text{otherwise} \end{aligned}$$

Decision surface

- $\mathbf{w}$ determines the orientation
- w_0 determines the location
- $-w_0/\|\mathbf{w}\|$ is the normal distance from the origin

Linear discriminant function

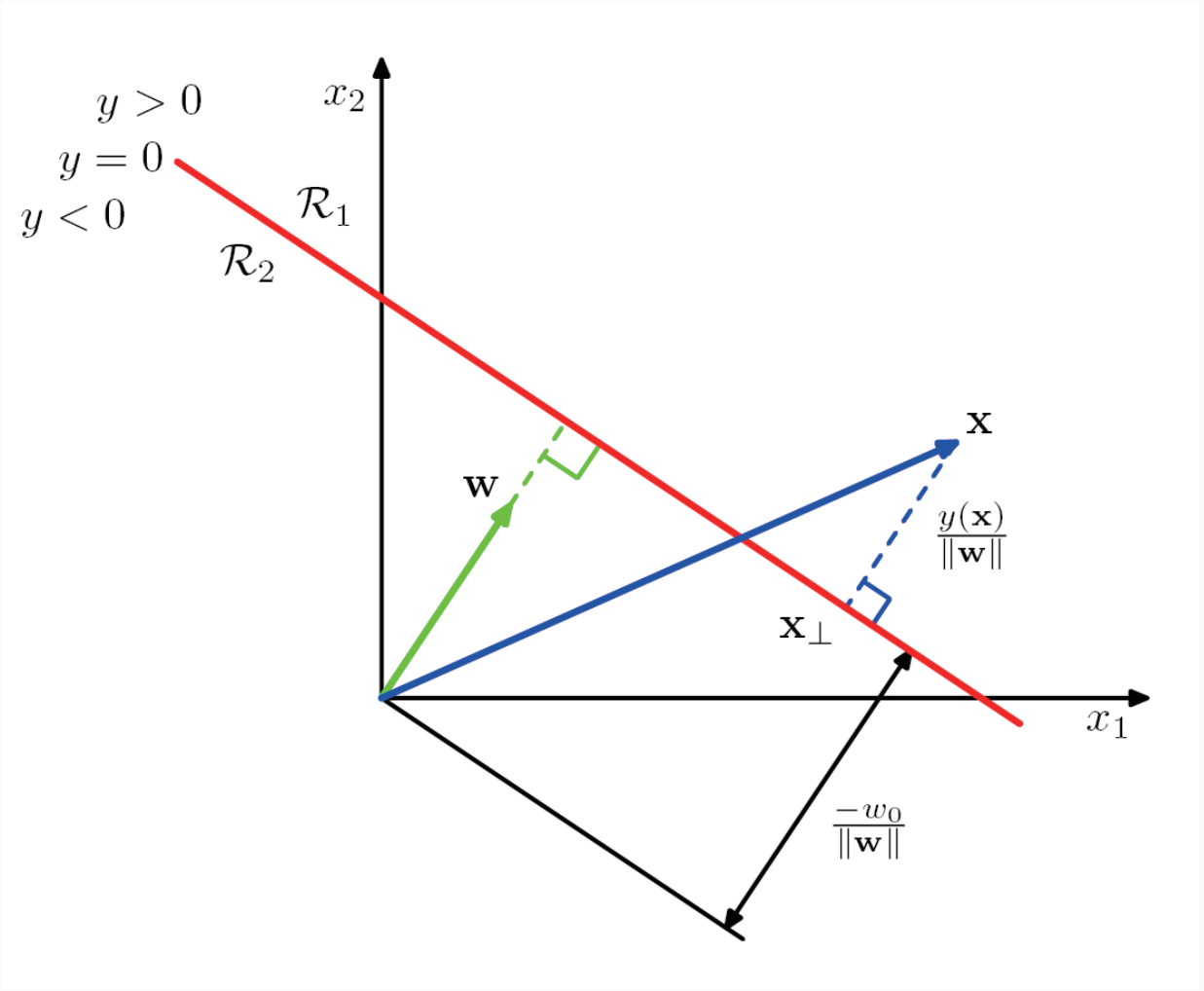


Figure 1: Decision surface of linear discriminant function

Linear discriminant function

Multiple classes

- (K-1) classifier: one-against-the rest
- $K(K\text{-}1)/2$ classifier: one-against-one
- K-class discriminant

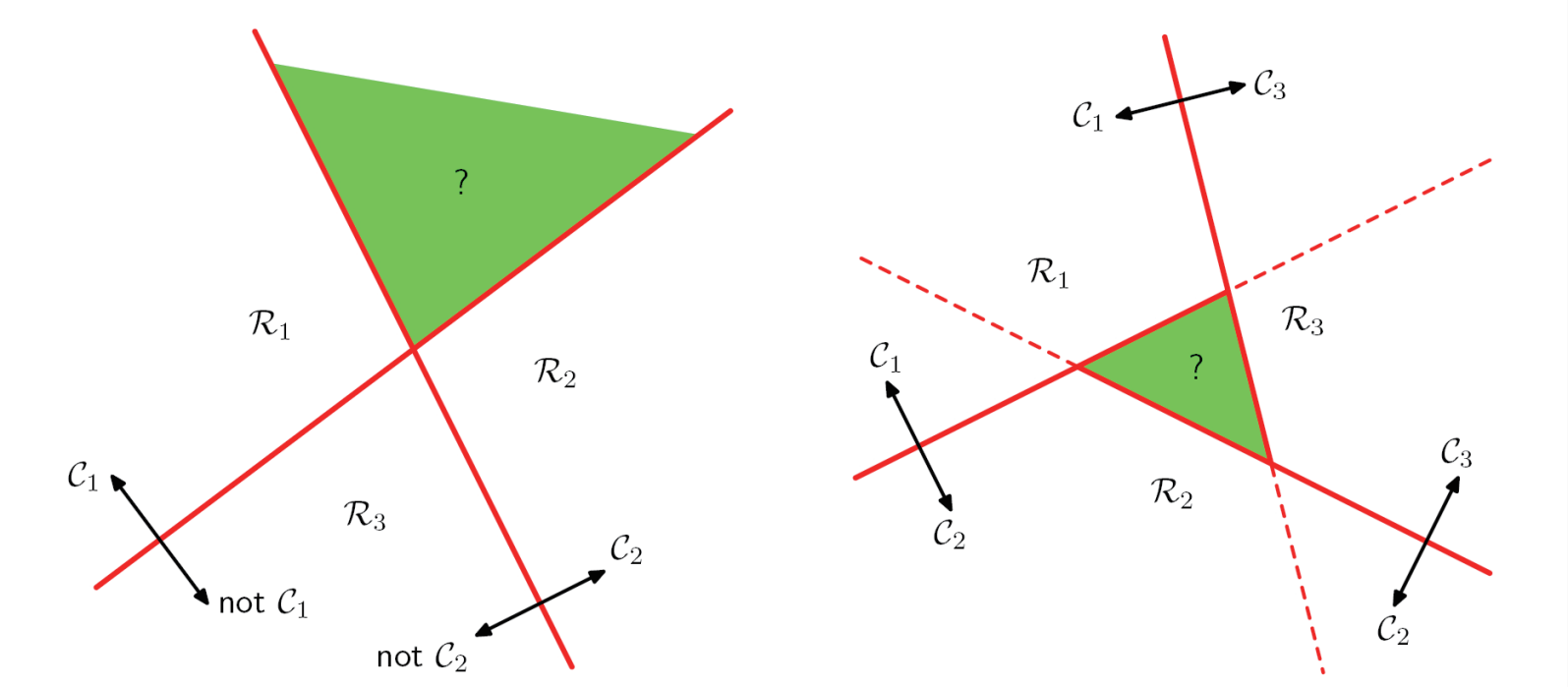


Figure 2: (K-1) classifier and $K(K\text{-}1)/2$ classifier

Linear discriminant function

K-class discriminant

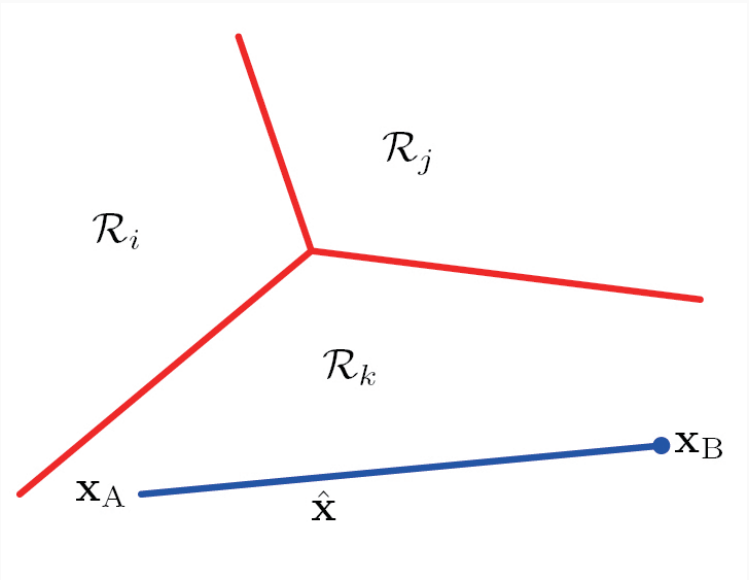
$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

$$\text{assign } \mathcal{C}_k \text{ if } y_k(\mathbf{x}) > y_j(\mathbf{x}) \text{ for all } j \neq k$$

- Decision surface

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} - (w_{k0} - w_{j0}) = 0$$

- Singly connected and convex



Contents

Least squares for classification

Min of sum-of-squares error $\rightarrow$ closed form solution

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

$$\mathbf{y}(\mathbf{x}) = \widetilde{\mathbf{W}}^T \widetilde{\mathbf{x}} \quad \text{where } \widetilde{\mathbf{w}}_k = (w_{k0}, \mathbf{w}_k^T)^T \quad \widetilde{\mathbf{x}} = (1, \mathbf{x}^T)^T$$

$$\text{assign} \ \ \arg\max_k y_k$$

Given a training dataset $\{\mathbf{x}_n, \mathbf{t}_n\}$ $(n = 1, 2, \cdots, N)$

$$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2} \mathrm{Tr} \left\{ (\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^T (\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T}) \right\}$$

$$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^T \widetilde{\mathbf{X}})^{-1} \widetilde{\mathbf{X}}^T \mathbf{T}$$

$$\mathbf{T} \in \{0, 1\}^{N \times K} \quad \widetilde{\mathbf{X}} \in \Re^{N \times (D+1)}$$

Least squares for classification

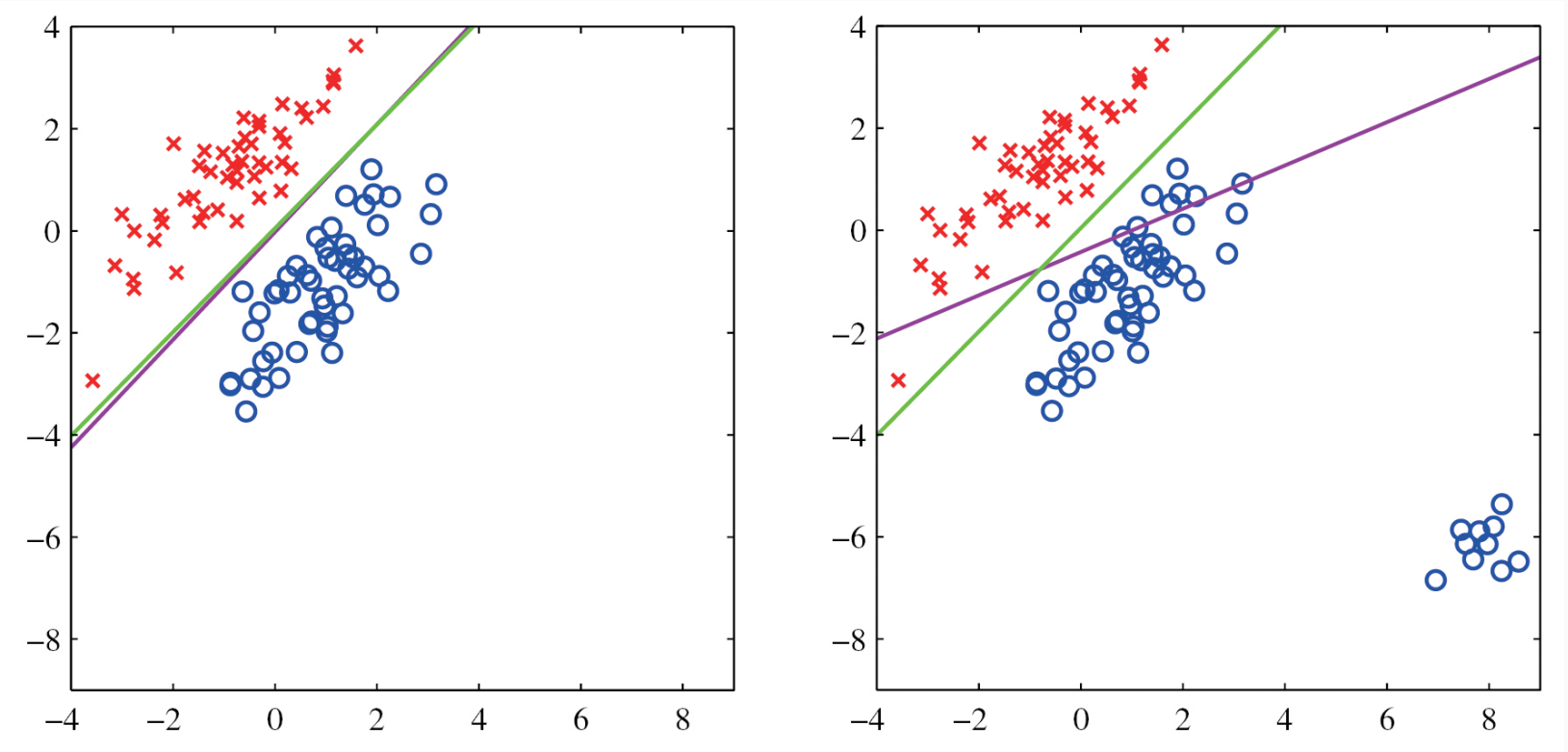


Figure 3: Two classes: least square and logistic regression

Least squares for classification

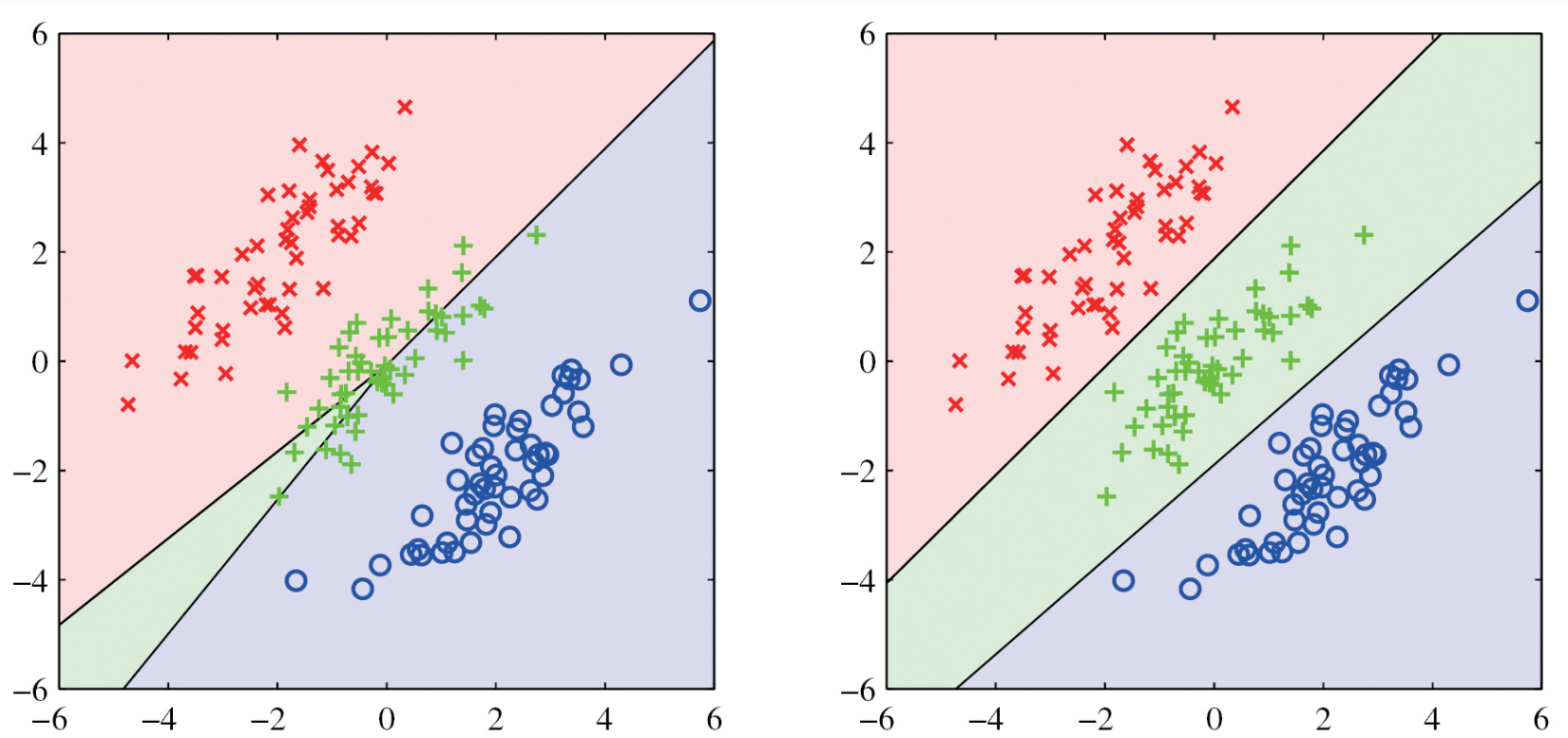


Figure 4: Multiple classes: least square and logistic regression

Contents

The perceptron algorithm

Generalized linear model

$$y(\mathbf{x}) = f(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}))$$

Nonlinear activation function given by a step function

$$f(a) = \begin{cases} +1, & a \geq 0 \\ -1, & a < 0 \end{cases}$$

Target values $t = +1$ for class $\mathcal{C}_1$ and $t = -1$ for class $\mathcal{C}_2$.

The perceptron algorithm

Perceptron criterion that we want to minimize

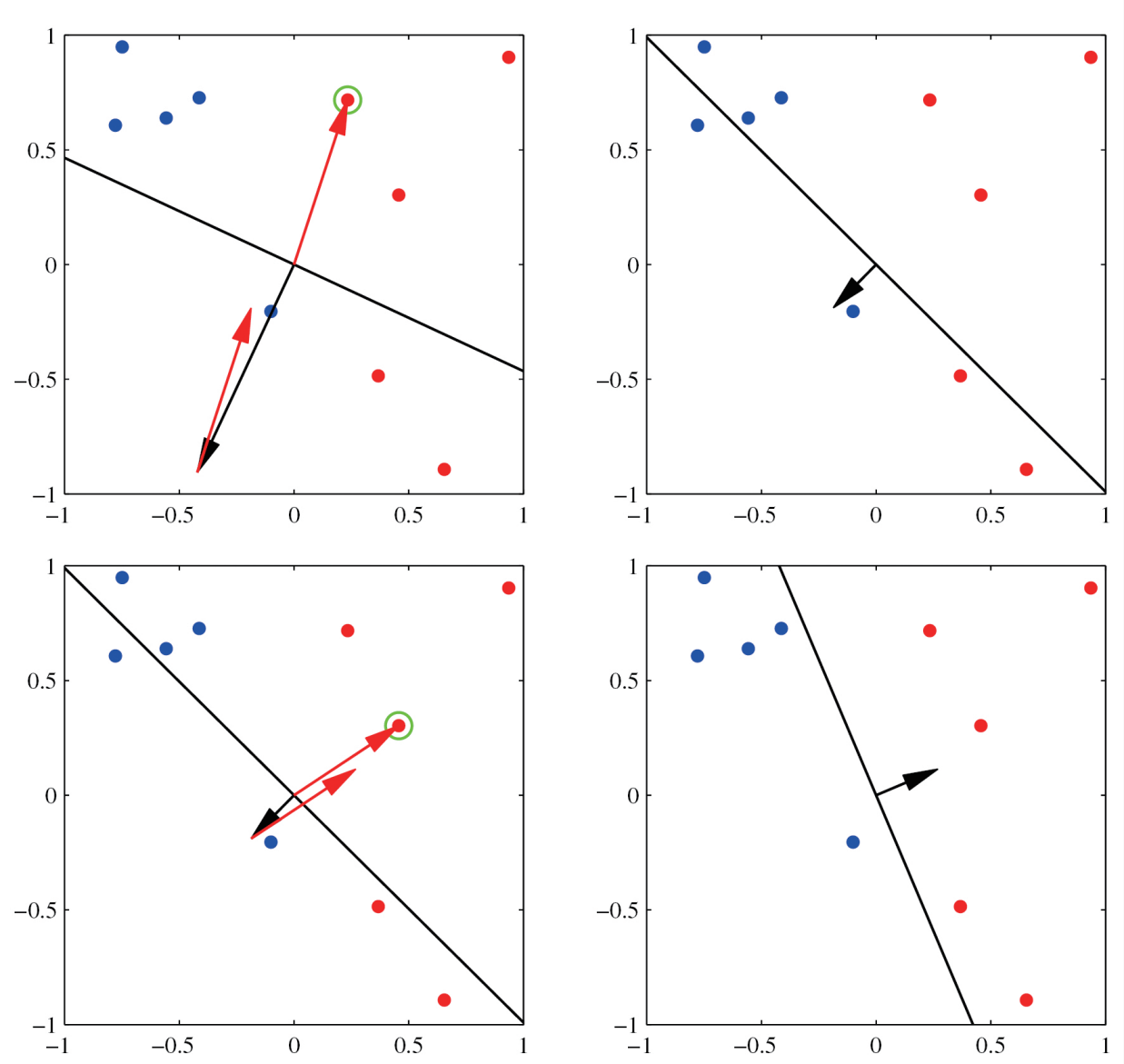
$$E_P(\mathbf{w}) = -\sum_{n \in \mathcal{M}} \mathbf{w}^T \boldsymbol{\phi}_n t_n$$

Misclassified pattern $\mathcal{M} = \{\mathbf{x}_n : \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) t_n < 0\}$

The stochastic gradient descent algorithm

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_P(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta \boldsymbol{\phi}_n t_n$$

The perceptron algorithm



Contents

Overview

An alternative approach is to use the functional form of the generalized linear model explicitly and to determine its parameters directly by using maximum likelihood. We shall see that there is an efficient algorithm finding such solutions known as *iterative reweighted least squares*, or *IRLS*.

The indirect approach to finding the parameters of a generalized linear model, by fitting class-conditional densities and class priors separately and then applying Bayes' theorem, represents an example of *generative* modelling, because we could take such a model and generate synthetic data by drawing values of $\mathbf{x}$ from the marginal distribution $p(\mathbf{x})$.

In the direct approach, we are maximizing a likelihood function defined through the conditional distribution $p(\mathcal{C}_k \mid \mathbf{x})$, which represents a from of *discriminative* training. One advantage of the discriminative approach is that there will typically be fewer adaptive parameters to be determined, as we shall see shortly.

Contents

Logistic regression

We begin our treatment of generalized linear models by considering the problem of two-class classification. Under rather general assumptions, the posterior probability of class $\mathcal{C}_1$ can be written as a logistic sigmoid acting on a linear function of the feature vector $\boldsymbol{\phi}$ so that

$$p(\mathcal{C}_1 \mid \boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma\left(\mathbf{w}^T \boldsymbol{\phi}\right) \tag{1}$$

with $p(\mathcal{C}_2 \mid \boldsymbol{\phi}) = 1 - p(\mathcal{C}_1 \mid \boldsymbol{\phi})$. Here, $\sigma(\cdot)$ is the *logistic sigmoid* function defined by

$$\sigma(a) = \frac{1}{1 + \exp(-a)}. \tag{2}$$

In the terminology of statistics, this model is known as *logistic regression*, although it should be emphasized that this is a model for classification rather than regression.

Logistic regression

For a data set $\{\boldsymbol{\phi}_n, t_n\}$, where $t_n \in \{0, 1\}$ and $= (\mathbf{x}_n)$, with $n = 1, ..., N$, the likelihood function can be written

$$p(\mathbf{t} \mid \mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} \{1 - y_n\}^{1-t_n} \tag{3}$$

where $\mathbf{t} = (t_1, ..., t_N)^T$ and $y_n = p(\mathcal{C}_1 \mid \phi_n)$. As usual, we can define an error function by taking the negative logarithm of the likelihood, which gives the *cross-entropy* error function in the form

$$E(\mathbf{w}) = -\log p(\mathbf{t} \mid \mathbf{w}) = -\sum_{n=1}^{N} \{t_n \log y_n + (1 - t_n) \log(1 - y_n)\} \tag{4}$$

where $y_n = \sigma(a_n)$ and $a_n = \mathbf{w}^T \boldsymbol{\phi}_n$. Taking the gradient of the error function with respect to $\mathbf{w}$, we obtain

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n) \boldsymbol{\phi}_n. \tag{5}$$

Logistic regression

The contribution to the gradient from data point n is given by the error $y_n - t_n$ between the target value and the prediction of the model, times the basis function vector $\boldsymbol{\phi}_n$. This takes precisely the same form as the gradient of the sum-of-squares error function for the linear regression model.

Contents

Iterative reweighted least squares

In the case of the linear regression models, the maximum likelihood solution, on the assumption of a Gaussian noise model, leads to a closed-form solution. This was a consequence of the quadratic dependence of the log likelihood function on the parameter vector $\mathbf{w}$. For logistic regression, there is no longer a closed-form solution, due to the nonlinearity of the logistic sigmoid function.

Iterative reweighted least squares

The error function is concave and hence has a unique minimum. Furthermore, the error function can be minimized by an efficient iterative technique based on the Newton-Raphson iterative optimization scheme, which uses a local quadratic approximation to the log likelihood function. The Newton-Raphson update, for minimizing a function $E(\mathbf{w})$, takes the form

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - \mathbf{H}^{-1}\nabla E(\mathbf{w}^{(\text{old})}) \tag{6}$$

where $\mathbf{H}$ is the Hessian matrix whose elements comprise the second derivatives of $E(\mathbf{w})$ with respect to the components of $\mathbf{w}$.

Iterative reweighted least squares

Let us first of all apply the Newton-Raphson method to the linear regression model with the sum-of-squares error function. The gradient and Hessian of this error function are given by

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (\mathbf{w}^T \boldsymbol{\phi}_n - t_n)\boldsymbol{\phi}_n = \boldsymbol{\Phi}^T \boldsymbol{\Phi} \mathbf{w} - \boldsymbol{\Phi}^T \mathbf{t} \tag{7}$$

$$\mathbf{H} = \nabla^2 E(\mathbf{w}) = \sum_{n=1}^{N} \boldsymbol{\phi}_n \boldsymbol{\phi}_n^T = \boldsymbol{\Phi}^T \boldsymbol{\Phi} \tag{8}$$

where $\boldsymbol{\Phi}$ is the $N \times N$ design matrix, whose n-th row is given by $\boldsymbol{\phi}_n^T$. The Newton-Raphson update then takes the form

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \left\{ \boldsymbol{\Phi}^T \boldsymbol{\Phi} \mathbf{w}^{(\text{old})} - \boldsymbol{\Phi}^T \mathbf{t} \right\} \tag{9}$$

$$= (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \mathbf{t} \tag{10}$$

which we recognize as the standard least-squares solution. Note that the error function in this case is quadratic and hence the Newton-Raphson formula gives the exact solution in one step.

Iterative reweighted least squares

Now let us apply the Newton-Raphson update to the cross-entropy error function for the logistic regression model. We see that the gradient and Hessian of this error function are given by

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\boldsymbol{\phi}_n = \boldsymbol{\Phi}^T(\mathbf{y} - \mathbf{t}) \tag{11}$$

$$\mathbf{H} = \nabla^2 E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1 - y_n)\boldsymbol{\phi}\boldsymbol{\phi}^T = \boldsymbol{\Phi}^T\mathbf{R}\boldsymbol{\Phi}. \tag{12}$$

Also, we have introduced the $N \times N$ diagonal matrix $\mathbf{R}$ with elements

$$R_{nn} = y_n(1 - y_n). \tag{13}$$

Iterative reweighted least squares

We see that the Hessian is no longer constant but depends on w through the weighting matrix R, corresponding to the fact that the error function is no longer quadratic. Using the property $0 < y_n < 1$, which follows from the form of the logistic sigmoid function, we see that $\mathbf{u}^T\mathbf{H}\mathbf{u} > 0$ for an arbitrary vector $\mathbf{u}$, and so the Hessian matrix $\mathbf{H}$ is positive definite. It follows that the error function is a concave function of $\mathbf{w}$ and hence has a unique minimum.

Iterative reweighted least squares

The Newton-Raphson update formula for the logistic regression model then becomes

$$\begin{aligned}\mathbf{w}^{(\text{new})} &= \mathbf{w}^{(\text{old})} - (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T(\mathbf{y}-\mathbf{t}) && (14)\\ &= (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\{\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi}\mathbf{w}^{(\text{old})} - \mathbf{\Phi}(\mathbf{y}-\mathbf{t})\} && (15)\\ &= (\mathbf{\Phi}^T\mathbf{R}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{R}\mathbf{z} && (16)\end{aligned}$$

where $\mathbf{z}$ is an N-dimensional vector with elements

$$\mathbf{z} = \mathbf{\Phi}\mathbf{w}^{(\text{old})} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t}). \quad (17)$$

Iterative reweighted least squares

We see that the update formula takes the form of a set of normal equations for a weighted least-squares problem. Because the weighing matrix $\mathbf{R}$ is not constant but depends on the parameter vector $\mathbf{w}$, we must apply the normal equations iteratively, each time using the new weight vector $\mathbf{w}$ to compute a revised weighing matrix $\mathbf{R}$. For this reason, the algorithm is known as *iterative reweighted least squares*, or *IRLS*.

Contents

Reference and further reading

- “Chap 4” of C. Bishop, Pattern Recognition and Machine Learning
- “Chap 3” of A. Geron, Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow