The Exercises of Mid-term

Yanwei Fu

November 21, 2017

1 Linear Regression

Let $(x_i, y_i)_{i=1}^n$ be our dataset, with $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}^p$. Linear regression can be formulated as empirical risk minimization, where the model is to predict y as $x^T\beta$ and we use the squared loss:

$$R^{emp}(\beta) = \sum_{i=1}^{n} \frac{1}{2} \left(y_i - x_i^T\beta\right)^2$$

1. Previously, we know that optimal parameter is

$$\hat{\beta} = \left(X^T X\right)^{-1} X^T Y \tag{1}$$

where X is a $n \times p$ matrix with ith row given x_i^T, and Y is a $n \times 1$ matrix with ith entry y_i. To warm up, please remind me how to derive the result of Eq (1).

2. Consider regularizing our empirical risk by incorporating a L_2 regularizer. That is, find β minimizing

$$\frac{C}{2} \parallel \beta \parallel_2^2 + \sum_{i=1}^{n} \frac{1}{2} \left(y_i - x_i^T\beta\right)^2$$

We know that the optimal parameter is given by the ridge regression estimator,

$$\hat{\beta} = \left(CI + X^T X\right)^{-1} X^T Y \tag{2}$$

Really? How? Anyone who can help prove Eq (2) would be great.

3. Question: Suppose we wish to introduce nonlinearities into the model, by transforming $x \to \phi(x)$. Show how this transformation may be achieved using the kernel trick. That is, let Φ be a matrix with ith row given by $\phi(x_i)^T$. The optimal parameters $\hat{\beta}$ would then be given by (previous part):

$$\hat{\beta} = \left(CI + \Phi^T \Phi\right)^{-1} \Phi^T Y$$

Express the predicted y values on the training set, $\Phi\hat{\beta}$, only in terms of Y and the Gram matrix $G = \Phi\Phi^T$, with $G_{ij} = \phi(x_i)^T \phi(x_i) = \kappa(x_i, x_j)$ where κ is some kernel function.

Compute an expression for the value of y_0 predicted by the model at a test vector x_0.

You will find the Woodbury matrix inversion formula useful:

$$(A + UBV)^{-1} = A^{-1} - A^{-1}U\left(B^{-1} + VA^{-1}U\right)^{-1} VA^{-1}$$

where A and B are square invertible matrices of size $n \times n$ and $p \times p$ respectively, and U and V are $n \times p$ and $p \times n$ rectangular matrices.

2 SVM –Fitting an SVM classifier by hand

Consider a dataset with 2 points in 1d:$(x_1 = 0, y_1 = -1)$ and $\left(x_2 = \sqrt{2}, y_2 = 1\right)$. Consider mapping each point to 3d using the feature vector $\phi(x) = \left[1, \sqrt{2}x, x^2\right]^T$. (This is equivalent to using a second order polynomial kernel.) The max margin classifier has the form

$$\min \parallel w \parallel^2 \qquad s.t.$$

$$y_1\left(w^T\phi(x_1) + w_0\right) \geqslant 1 \tag{3}$$

$$y_1\left(w^T\phi(x_1) + w_0\right) \geqslant 1 \tag{4}$$

1. Write down a vector that is parallel to the optimal vector w. Hint: Please remember that w is perpendicular to the decision boundary between the two points in the 3d feature space.

2. What is the value of the margin that is achieved by this w? Hint: recall that the margin is the distance from each support vector to the decision boundary. Hint: think about the geometry of 2 points in space, with a line separating one from the other.

3. Solve for w, using the fact the margin is equal to $1/||\mathrm{w}||$.

4. Solve for w_0 using your value for w in Eq (3) and Eq (4) . Hint: the points will be on the decision boundary, so the inequalities will be tight.

5. Write down the form of the discriminant function $f(x) = w_0 + w^T\phi(x)$ as an explicit function of x.

3 Neural Network

Recall the definition of a 1 hidden layer neural network for binary classification. The objective function is:

$$J = -\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i) + \frac{1}{2} \sum_{j=1}^{p} \sum_{k=1}^{m} C \mid W_{jk}^{h} \mid^2 + \frac{1}{2} \sum_{k=1}^{m} C \mid W_{k}^{o} \mid^2$$

and the network definition is

$$\hat{y}_i = s \left(b^o + \sum_{k=1}^{m} W_k^o h_{ik} \right)$$

$$h_{ik} = s \left(b_k^h + \sum_{j=1}^{p} W_{jk}^h x_{ij} \right)$$

1. Verify that the derivatives needed for gradient descent are:

$$\frac{dJ}{dW_k^o} = CW_k^o + \sum_{i=1}^{n} (\hat{y}_i - y_i) h_{ik}$$

$$\frac{dJ}{dW_{jk}^h} = CW_{jk}^h + \sum_{i=1}^{n} (\hat{y}_i - y_i) W_k^o h_{ik} (1 - h_{ik}) x_{ij}$$

2. Suppose instead that you have an L layer neural network for binary classification, with each hidden layer having m neurons with logistic nonlinearity. Define carefully the network, giving the parameterization of each layer, and derive the backpropagation algorithm to compute the derivatives of the objective with respect to the parameters. You may ignore bias terms for simplicity. (Hint: we may want to refer to Chap7 of our slides, or Chap 5.3 (Error Backpropagation) of the book "Pattern Recognition and Machine Learning" for the backpropagation).