UNIVERSITY OF OXFORD

Machine Learning
Michaelmas Term 2017
Week 6

Problem Sheet 3

1 Optimisation Methods for ℓ_1-regularisation

1. Show that if you use the absolute loss function with the regularisation term corresponding to Lasso (called ℓ_1 regularization as the penalty is on the ℓ_1 norm of the parameter vector $\mathbf{w}$), the optimization problem can be solved using linear programming. The objective function is:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{N} |\mathbf{w}^\mathsf{T}\mathbf{x}_i - y_i| + \lambda \sum_{i=1}^{D} |w_i| \tag{1.1}$$

2. If we use the squared loss instead of absolute loss, we are optimising the Lasso objective. We can no longer use linear programming because of the quadratic term in the objective. Write the sub-gradient descent update rule with step size η, *i.e.,* write how you would obtain $\mathbf{w}_{t+1}$ using $\mathbf{w}_t$ and an (explicitly computed) subgradient of the objective function at $\mathbf{w}_t$ and step-size η. The objective function is:

$$\mathcal{L}_{\text{lasso}}(\mathbf{w}) = \sum_{i=1}^{N} (\mathbf{w}^\mathsf{T}\mathbf{x}_i - y_i)^2 + \lambda \sum_{i=1}^{D} |w_i| \tag{1.2}$$

2 Maximum Likelihood for Logistic Regression

Consider the sigmoid function, defined as $\sigma(z) = \frac{1}{1+e^{-z}}$. Note that $\lim_{z\to-\infty} \sigma(z) = 0$ and $\lim_{z\to\infty} \sigma(z) = 1$. Thus, for binary classification problems, we can compose a linear function with the sigmoid function to model the probability that a given input $\mathbf{x}$ belongs to one of the two classes $\{0, 1\}$. More precisely, for parameter vector $\mathbf{w} \in \mathbb{R}^D$, and input vector $\mathbf{x} \in \mathbb{R}^D$,[1] the label $y \in \{0, 1\}$ is given by the following model:

$$\Pr(y = 1 \mid \mathbf{w}, \mathbf{x}) = \sigma(\mathbf{x}^\mathsf{T}\mathbf{w}) \tag{2.1}$$

$$\Pr(y = 0 \mid \mathbf{w}, \mathbf{x}) = 1 - \sigma(\mathbf{x}^\mathsf{T}\mathbf{w}) \tag{2.2}$$

1. Show that the derivative of σ, $\sigma'(z) = \sigma(z)(1 - \sigma(z))$

2. Suppose you have i.i.d. data $\mathcal{D} = \langle(\mathbf{x}_i, y_i)\rangle_{i=1}^N$. Logistic regression is a discriminative model and we don't explicitly model the $\mathbf{x}_i$, the class labels y_i are considered as random variables. Write the likelihood of observing the labels $y_1, \ldots, y_N$, given the model parameters $\mathbf{w}$ and the inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N$.

[1] As always we will assume that an extra dimension which takes value 1 on every data point has been added to avoid dealing with the constant term separately.

3. Compute the gradient and Hessian of the negative log-likelihood. Show that the Hessian is positive semi-definite.

4. What algorithm would you use to find the maximum likelihood estimate and what can you say about the obtained solution?

3 Support Vector Machines

3.1 SVM Formulation

Let us look at support vector machines (without kernels) and assume that the data is *linearly separable*. In order to maximize the margin, a more natural formulation would be the following: Fix $\|\mathbf{w}\|_2 = 1$, so the distance of $\mathbf{x}$ from hyperplane defined by $(\mathbf{w}, w_0)$ is exactly $|\mathbf{x} \cdot \mathbf{w} + w_0|$. Then, we can define a mathematical program:

$$\begin{aligned} \text{maximize} \quad & \alpha \\ \text{subject to} \quad & y_i(\mathbf{x}_i \cdot \mathbf{w} + w_0) \geq \alpha \text{ for } i = 1, \ldots, N \\ & \|\mathbf{w}\| = 1 \end{aligned}$$

Unfortunately, the condition $\|\mathbf{w}\| = 1$ implies that the set of admissible $\mathbf{w}$ do not form a convex set. Argue that relaxing the constraint to be $\|\mathbf{w}\| \leq 1$ does not change the optimal solution of the above program. Then show that this formulation is equivalent to the one we considered in class, *i.e.,* show that an optimal solution for one can be used to obtain an optimal solution for the other.

3.2 Simple Observations

1. Suppose we use the SVM formulation for separable data, and that the data indeed is linearly separable. Recall that in this case, support vectors are those points $\mathbf{x}_i$ in the dataset those for which $y_i(\mathbf{w}^* \cdot \mathbf{x}_i + w_0^*) = 1$, where $\mathbf{w}^*, w_0^*$ is the max-margin hyperplane. If your dataset consists of N points in D dimensional space, what is the *maximum* number of support vectors possible? the minimum?

2. Suppose you use the formulation for the non-separable case, *i.e.,* with slack variables ζ_i, but your data is actually linearly separable. Do you always recover the "true" max-margin separating hyperplane?

4 Reducing the cost of linear regression for large D, small N

The ridge method is a regularized version of least squares with objective function:

$$\min_{\mathbf{w} \in \mathbb{R}^D} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \tag{4.1}$$

Here λ is a scalar, the input matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ and the output vector $\mathbf{y} \in \mathbb{R}^N$. The parameter vector $\mathbf{w} \in \mathbb{R}^D$ is obtained by differentiating the cost function, yielding the *normal equations*

$$(\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I}_D)\mathbf{w} = \mathbf{X}^T\mathbf{y}, \tag{4.2}$$

where $\mathbf{I}_D$ is the $D \times D$ identity matrix. The predictions $\widehat{\mathbf{y}} = \widehat{\mathbf{y}}(\mathbf{X}_*)$ for new test points $\mathbf{X}_* \in \mathbb{R}^{N^* \times D}$ are obtained by evaluating the hyperplane

$$\widehat{\mathbf{y}} = \mathbf{X}_*\mathbf{w} = \mathbf{X}_*(\mathbf{X}^T\mathbf{X} + \lambda \mathbf{I}_D)^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{H}\mathbf{y}. \tag{4.3}$$

The matrix $\mathbf{H}$ is known as the *hat matrix* because it puts a "hat" on y.

1. Show that the solution can be written as $\mathbf{w} = \mathbf{X}^T\widetilde{\mathbf{w}}$, where $\widetilde{\mathbf{w}} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$.

2. Show that $\widetilde{\mathbf{w}}$ can also be written as follows: $\widetilde{\mathbf{w}} = (\mathbf{X}\mathbf{X}^T + \lambda \mathbf{I}_N)^{-1}\mathbf{y}$ and, hence the predictions can be written as follows:

$$\widehat{\mathbf{y}} = \mathbf{X}_*\mathbf{w} = \mathbf{X}_*\mathbf{X}^T\widetilde{\mathbf{w}} = [\mathbf{X}_*\mathbf{X}^\mathsf{T}]([\mathbf{X}\mathbf{X}^T] + \lambda \mathbf{I}_N)^{-1}\mathbf{y}. \tag{4.4}$$

   (This an *awesome trick* because if $N = 20$ patients with $D = 10,000$ gene measurements, the computation of $\widetilde{\mathbf{w}}$ only requires inverting the $N \times N$ matrix, while the direct computation of $\mathbf{w}$ would have required the inversion of a $D \times D$ matrix.)

Remark: Observe that this is the same idea as the kernel trick used in the context of SVMs. In general, the kernel trick is widely applicable and should be used whenever the amount of data is significantly smaller than the dimension of the data, either to begin with or because of basis expansion.

5 Linear algebra revision: Singular Value Decomposition (SVD)

Note: This question is intended for self-study and will only be discussed in class if time permits. We will use SVD frequently in Lectures 17, 18, so please make sure you revise this before then.

Once you start looking at raw data, one of the first things you notice is how redundant it often is. In images, it's often not necessary to keep track of the exact value of every pixel; in text, you don't always need the counts of every word. Correlations among variables also create redundancy. For example, if every time a gene, say A, is expressed another gene B is also expressed, then to build a tool that predicts patient recovery rate from gene expression data, it seems reasonable to remove either A or B. Most situations are not as clear-cut.

In this question, we'll look at eigenvalue methods for factoring and projecting data matrices (images, document collections, image collections), with an eye to one of the most common uses: Converting a high-dimensional data matrix to a lower-dimensional one, while minimizing the loss of information.

The Singular Value Decomposition (SVD) is a matrix factorization that has many applications in information retrieval, collaborative filtering, least-squares problems and image processing.

Let $\mathbf{X}$ be an $n \times n$ matrix of real numbers; that is $\mathbf{X} \in \mathbb{R}^{n \times n}$. Assume that $\mathbf{X}$ has n eigenvalue-eigenvector pairs $(\lambda_i, \mathbf{q}_i)$:

$$\mathbf{X}\mathbf{q}_i = \lambda_i \mathbf{q}_i \qquad i = 1, \ldots, n$$

If we place the eigenvalues $\lambda_i \in \mathbb{R}$ into a diagonal matrix $\mathbf{\Lambda}$ and gather the eigenvectors $\mathbf{q}_i \in \mathbb{R}^n$ into a matrix $\mathbf{Q}$, then the eigenvalue decomposition of $\mathbf{X}$ is given by

$$\mathbf{X} \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \ldots & \mathbf{q}_n \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \ldots & \mathbf{q}_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \tag{5.1}$$

or, equivalently,

$$\mathbf{X} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}.$$

For a symmetric matrix, i.e. $\mathbf{X} = \mathbf{X}^\mathsf{T}$, one can show that $\mathbf{X} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$. But what if $\mathbf{X}$ is not a square matrix? Then the SVD comes to the rescue. Given $\mathbf{X} \in \mathbb{R}^{m \times n}$, the SVD of $\mathbf{X}$ is a factorization of the form

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}.$$

These matrices have some interesting properties:

- $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$ is diagonal with positive entries (singular values σ in the diagonal).
- $\mathbf{U} \in \mathbb{R}^{m \times n}$ has orthonormal columns: $\mathbf{u}_i^\mathsf{T}\mathbf{u}_j = 1$ only when $i = j$ and 0 otherwise.
- $\mathbf{V} \in \mathbb{R}^{n \times n}$ has orthonormal columns and rows. That is, $\mathbf{V}$ is an orthogonal matrix, so $\mathbf{V}^{-1} = \mathbf{V}^\mathsf{T}$.

Often, $\mathbf{U}$ is m-by-m, not m-by-n. The extra columns are added by a process of orthogonalization. To ensure that dimensions still match, a block of zeros is added to $\mathbf{\Sigma}$. For our purposes, however, we will only consider the version where $\mathbf{U}$ is m-by-n, which is known as the *thin-SVD*.

It will turn out useful to introduce the vector notation:

$$\mathbf{X}\mathbf{v}_j = \sigma_j \mathbf{u}_j \qquad j = 1, 2, \ldots, n$$

where $\mathbf{u} \in \mathbb{R}^m$ are the left *singular vectors*, $\sigma \in [0, \infty)$ are the *singular values* and $\mathbf{v} \in \mathbb{R}^n$ are the right singular vectors. That is,

$$\mathbf{X} \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_n \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \ldots & \mathbf{u}_n \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} \tag{5.2}$$

or $\mathbf{XV} = \mathbf{U\Sigma}$. Note that there is no assumption that $m \geq n$ or that $\mathbf{X}$ has full rank. In addition, all diagonal elements of $\mathbf{\Sigma}$ are non-negative and in non-increasing order:

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$$

where $p = \min(m, n)$.

Question: Outline a procedure for computing the SVD of a matrix $\mathbf{X}$. Hint: assume you can find the eigenvalue decompositions of the symmetric matrices $\mathbf{X}^\mathsf{T}\mathbf{X}$ and $\mathbf{X}\mathbf{X}^\mathsf{T}$.