DATA12001 Advanced Course in Machine Learning

Exercise 2 **Due April 16, 23:59**

In this exercise set we focus on linear classifiers and regressors (lecture slides 5).

Submit pen&paper (as JPG or PDF) and application exercises (as PDF) to Moodle. Submit TMC exercises to TMC.

---

1 Logistic regression

pen&paper, 5p

Logistic regression is a discriminative model for classification, where the output y is modelled with a Bernoulli distribution

$$p(y = 1|\mathbf{w}, \mathbf{x}) = sigm\big(\mathbf{w}^T\mathbf{x}\big)\,,$$

where

$$sigm(z) = \frac{1}{1 + e^{-z}} \quad .$$

Here, we assume that labels are either 1 or -1.

(a) Prove that the negative log-likelihood is

$$-\sum_n \log p(y_n|\mathbf{w}, \mathbf{x}_n) = \sum_n \log(1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n}) \quad .$$

(b) We define the right-hand side of the above equation as the loss we wish to minimize. Prove that the gradient $\boldsymbol{g}$ and the Hessian matrix $\boldsymbol{H}$ of the loss is equal to

$$\mathbf{g} = \mathbf{X}^T(\boldsymbol{\mu} - (\mathbf{y} + 1)/2)$$
$$\mathbf{H} = \mathbf{X}^T\mathbf{S}\mathbf{X},$$

where $\boldsymbol{\mu}$ is a vector of length N, $\mathbf{y}$ is a vector containing entries y_n, $\boldsymbol{\mu}_n = sigm\big(\mathbf{w}^T\mathbf{x}_n\big)$, $\boldsymbol{S}$ is a diagonal matrix $\boldsymbol{S}_{nn} = \boldsymbol{\mu}_n(1 - \boldsymbol{\mu}_n)$, and $\boldsymbol{X}$ is a matrix with rows as inputs. Hint: Solve $\boldsymbol{g}$ by separating the cases $y = 1$ and $y = -1$.

(c) Prove that $\boldsymbol{H}$ is positive semidefinite. This makes the loss convex. Why this is important?

Break S into S^1/2 and (S^1/2)^T

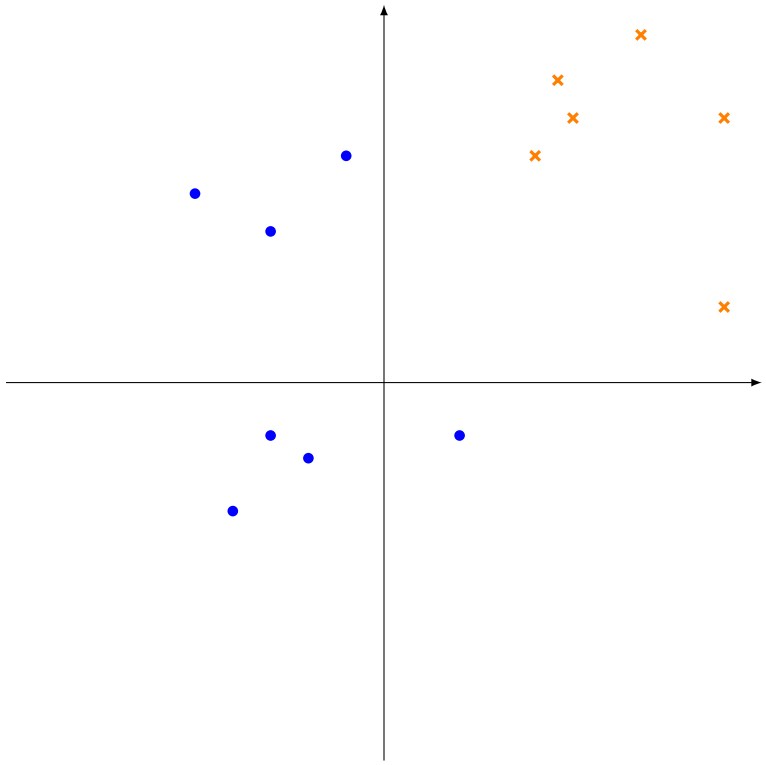
2 Support Vector Machines

pen&paper, 5p

Answer the following questions related to support vector machines (SVM).

(a) The figure below presents a two-dimensional data set with two classes, indicated by blue dots (class -1) and orange crosses (class +1). Figure out the solution of a linear SVM classifier for this data set, by simply thinking about the properties of SVMs—no need to run any algorithm. Then mark in the figure the following things:

- The decision surface that separates the two classes
- The weight vector $\mathbf{w}$
- The support vectors corresponding to $\boldsymbol{\alpha}_i > 0$
- The margin (=quantity that SVM maximizes) of the classifier



Then draw two new samples that violate the margin (and assume the decision surface does not change—this would not be guaranteed in real case). Do this so that for one of the new samples the slack variable ξ_i would be between 0 and 1 and for the other it would be bigger than 1.

(b) The *primal problem* of SVM for non-separable data is given by

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_n \xi_n$$
$$\text{s.t.} \quad (\mathbf{w}^T\mathbf{x}_n + b)y_n \geq 1 - \xi_n \quad \text{and} \quad \xi_n \geq 0$$

and solving it is equivalent to solving the *dual problem*

$$\max_{\boldsymbol{\alpha}} \left[\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j\right]$$
$$\text{s.t.} \quad 0 \leq \alpha_i \leq C \quad \text{for all } i$$
$$\sum_i \alpha_i y_i = 0.$$

Here x_n are the the inputs y_n are the outputs, and C is the parameter given by the user penalizing the violations.

Prove this by first writing the Lagrangian for the the primal problem and then solving for $\nabla_{\mathbf{w}}\Lambda = 0$, $\nabla_b\Lambda = 0$, and $\nabla_\xi\Lambda = 0$ to obtain the solution to the minimization problem. Then simplify the maximization problem.

Hints: the constraints $\xi_n \geq 0$ will introduce additional multipliers, say β_n. These multipliers can and should be eliminated when simplifying the maximization problem.

3 Naive Bayes

TMC, 5p

Implement the NaiveBayes algorithm (TMC exercise: `part2/01.nb`).

Consider a binary classification problem, where the input x is a *binary* vector of length k. Naive Bayes is a generative model that assumes that features in x are independent given y, and models each feature with a Bernoulli distribution, $p(\mathbf{x} \mid y) = \prod_i p(x_i \mid y)$. Note that the features are *not* identically distributed.

Using the training data, the algorithm estimates the model parameters by maximixing the log-likelihood. Then given a new data point $\mathbf{x}$, the algorithm selects the label that has the highest probability

$$\log p(y \mid \mathbf{x}) = \log p(y) + \log p(\mathbf{x} \mid y) - \log p(\mathbf{x}) \quad .$$

Note that we can ignore $p(\mathbf{x})$ since it is constant.

This classifier is a linear classifier that is, one can write it as $sign(\mathbf{w}^T\mathbf{x} + b)$.

Implement `nb(X, labels)` that computes the weight vector $\mathbf{w}$ and b given the training data.

Hints: The model has $2k + 1$ parameters, figure out what they are, and what are their ML estimates. Once you figure out the estimates, derive the weights $\mathbf{w}$ and b from these estimates. A convenient way of expressing probability mass function for a single Bernoulli random variable is $p(a) = \theta^a(1-\theta)^{1-a}$, where $a = 0, 1$.

Submit your code to the TMC system.

4 Iterative Reweighted Least Squares

TMC, 5p

Implement the IRLS algorithm (TMC exercise: `part2/02.irls`).

See the slides for the algorithm description.

The algorithm should perform Newton's step `itercnt` times using the given weights `w` as the starting point.

The algorithm should return the updated weights `w`, and arrays `err` and `misclass` of length `itercnt + 1`.

`misclass[i]` is the misclassification rate (normalized between 0 and 1) after i iterations. `misclass[0]` is the misclassification rate with initial `w`. If $\mathbf{w}^T\mathbf{x} = 0$ (that is $\mathbf{x}$ is at the decision boundary), then the classifier should predict 1. If you use the `sign` function, be aware that `sign(0)` returns 0.

`err[i]` is the loss (=negative log-likelihood) after i iterations. `err[0]` is the loss with initial `w`. See the slides for the exact definition.

Submit your code to the TMC system.

5 Lasso

TMC, 5p

Implement the Lasso algorithm (TMC exercise: `part2/03.lasso`).

See the slides for the algorithm description.

The algorithm loops `itercnt` times. During each iteration, the algorithm updates each coordinate w_d in the weight vector $\mathbf{w}$ (see the slides to the update formula). The coordinates should be updated in order. The first coordinate should not be regularized, that is, the update for the first feature is $w_d = c_d/a_d$ (see the slides), as this feature corresponds to the bias term.

The algorithm should return the updated weights $\mathbf{w}$.

Submit your code to the TMC system.