Exercises
Machine Learning
AS 2012

Machine Learning Laboratory
Dept. of Computer Science, ETH Zürich
Prof. Dr. Joachim M. Buhmann
Web http://ml2.inf.ethz.ch/courses/ml/

Email questions to: Alberto Giovanni Busetto
alberto.busetto@inf.ethz.ch

Series 4, Nov 6th, 2012 (SVMs)

Problem 1 (Support Vector Machines):

The objective of this exercise is to implement the 1-norm soft margin support vector machine. This SVM is defined by the optimization problem

$$\text{minimize } \langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{i=1}^{l} \xi_i$$

with respect to $\xi, \mathbf{w}, b$ and subject to the constraints

$$\begin{aligned} y_i \left(\langle \mathbf{w}, \mathbf{x}_i \rangle + b \right) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \, . \end{aligned}$$

Here, $\mathbf{w}$ denotes the weight vector of the hyperplane, $\mathbf{x_i}$ are the training data points and ξ_i the corresponding slack variables. The dual optimization problem is given by

$$\text{maximize} \qquad W(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \ ,$$

with respect to $\boldsymbol{\alpha}$ and subject to the constraints

$$\begin{aligned} &\sum_{i=1}^{l} \alpha_i y_i = 0 \\ &0 \leq \alpha_i \leq C \, . \end{aligned}$$

$y_1, \ldots, y_l$ denote the class labels for the training vectors $\mathbf{x}_i$. The α_i are the Lagrange parameters, and K denotes the kernel function. We write α_i^* for the optimal values of the Lagrange parameters computed as solution of the above dual problem.

The offset (bias) b^* can be computed by noting that any support vector $\mathbf{x}_i$ for which $0 < \alpha_i < C$ satisfies $y_i f(x_i) = 1$, resulting in

$$b^* := y_i - \sum_{j \in \text{SV}} y_j \alpha_j^* K(\mathbf{x}_i, \mathbf{x}_j) \ .$$

Whilst we can solve this for any support vector x_i, for numerical stability, we compute this offset b by averaging over the individual offset value given by all the support vectors.

From the solution of the maximization problem and the bias, the classification function is constructed as

$$f(\mathbf{x}) := \sum_{i=1}^{l} y_i \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) + b^* \ .$$

The predicted class label $hyp \in \{-1, 1\}$ of a test value $\mathbf{x}$ is determined as

$$hyp := \text{sign}\left(f(\mathbf{x}) \right) \ .$$

The SVM implementation consists of two matlab functions, one for training

```
[yALPHA, B, SV] = svmtrain(SAMPLES, CLASSES, C, KERNEL, PARAM),
```

and one for classification:

```
hyp = svmclass(X, yALPHA, B, SV, KERNEL, PARAM).
```

The parameters are:

`yALPHA`	Lagrange parameters of the dual problem weighted by the label $y_i\alpha_i$.
`B`	Bias.
`SV`	The support vectors.
`SAMPLES`	The matrix of input vectors from the training data set, with each row corresponding to one data vector.
`CLASSES`	Vector of training class labels; `CLASSES(i)` specifies the class label of `SAMPLES(i,:)`.
`C`	Soft margin parameter (C in the optimization problem above)
`KERNEL`	We want to be able to specify different types of kernels (see below), so we hand over this string with possible values `'linear'`, `'polynomial'` or `'rbf'`.
`PARAM`	The kernel parameters (d and σ, in the notation used below).
`X`	Test data set: a matrix containing test data points as column vectors.
`hyp`	The vector of predicted class labels $hyp \in \{-1, 1\}$.

The kernels are defined as:

$$\begin{aligned} \text{Linear: } K(\mathbf{x},\mathbf{y}) &= \mathbf{x}^t\mathbf{y} \\ \text{Polynomial: } K(\mathbf{x},\mathbf{y}) &= \left(\mathbf{x}^t\mathbf{y}+1\right)^d \\ \text{RBF: } K(\mathbf{x},\mathbf{y}) &= \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|}{\sigma}\right) . \end{aligned}$$

The set of support vectors is, in theory, given by $\{\, i \,|\alpha_i > 0\}$. In practice, we bound this away from zero by using $\{\, i \,|\alpha_i > \epsilon\}$, where $\epsilon \approx 10^{-8}$. We assume that everything closer to zero cannot be reliably labeled a support vector due to the limited resolution of machine arithmetics.

Here is what we want you to do:

1. Download the archive file `svmfiles.zip` from the course webpage. The archive contains the following files:
   - `data1.txt`, `data2.txt`, `data3.txt` - Data sets.
   - `cl1.txt`, `cl2.txt`, `cl3.txt` - Class labels for the data sets.
   - `uspsdata.txt`, `uspscl.txt` - US Postal Service data set and class labels files.
   - `pr_loqo2.m` - Quadratic optimization matlab code by Robert J. Vanderbei, Princeton University.
   - `svmplot.m` - Matlab plot code.
2. Implement the functions `svmtrain` and `svmclass`. Remember that for training the classifier, we have to solve the dual (!) optimization problem, which is what the `pr_loqo2.m` code is for.
3. Train your implementation on the data sets `data*.txt`. Visualize the resulting decision functions using `svmplot`. For each choice of the kernel, try different values for the kernel parameters and the margin parameter C.
4. Cross-validate your SVM on the USPS (US Postal service) data set: Use a random subset of the data (100 samples) for training. Then apply the classifier to the remaining points (make sure you delete the training points from the test set!) and compare the results with the corresponding class labels. Once again, use different kernels and parameters and try to determine a configuration which works well on the data. If you are interested to visualize the i-th datapoint in the data set, you can do this by: `figure; colormap('gray'); imagesc(reshape(uspsdata(i,:),16,16)');`

Please hand in the following:

1. A hardcopy of your code for `svmtrain`.
2. The plots you created using `svmplot` for the `data*.txt` data sets.
3. A short discussion of the performance of the different kernels for the different data sets.

Problem 2 (Lagrange Multipliers):

We consider a constrained optimization problem in standard form, for $f_i : \mathbb{R}^n \to \mathbb{R}$ and $g_j : \mathbb{R}^n \to \mathbb{R}$:

$$\begin{array}{ll} \min_{x \in \mathbb{R}^n} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0 \text{ for } i = 1, \ldots, m \\ & g_j(x) = 0 \text{ for } j = 1, \ldots, p \end{array},$$

$f_0(x)$ is called the objective function, $f_i(x) \leq 0$ are called the inequality constraints and $g_j(x) = 0$ are called the equality constraints. Such problems arise in many applications including machine learning.

The basic idea in Lagrangian duality is to take the constraints into account by augmenting the objective function with a weighted sum of the constraint functions. The Lagrangian is defined as

$$\mathcal{L}(x, \lambda, \gamma) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} \gamma_j g_j(x),$$

where λ and γ are real valued vectors called the Lagrange multipliers with $\lambda_i \geq 0$ and γ_i free. Hence λ_i is the Lagrange multiplier associated with the ith inequality constraint $f_i(x) \leq 0$ and similarly γ_j is the Lagrange multiplier associated with the equality constraint $g_j(x) = 0$. The vectors λ and γ are also known as dual variables.

1. For $\mathbf{x} \in \mathbb{R}^d$, solve the following optimization problem:

$$\arg\max_{\mathbf{x}} f_0(\mathbf{x}) = 1 - \|\mathbf{x}\|_2$$

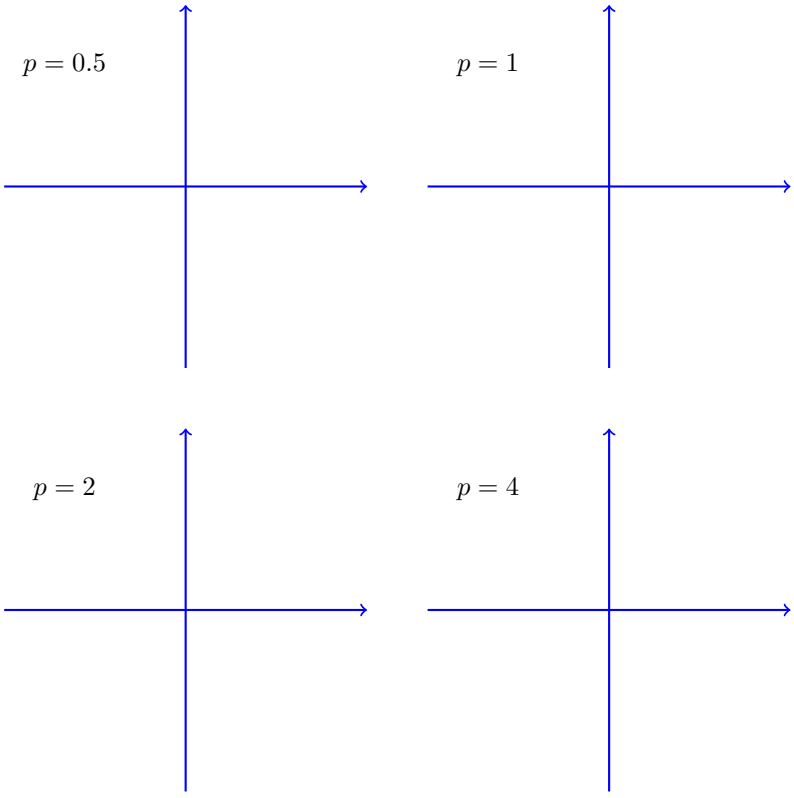
such that

$$\sum_{i=1}^{d} x_i = 1$$

where $\| \cdot \|_p$ denotes the L_p norm.

(a) Look up the definition for the L_p norm, and write it down mathematically. What is the common name for the L_2 norm?

(b) For a two dimensional input, draw the contours for $\| \cdot \|_p$ for the different values of p specified.



(c) Using the contours for $p = 2$, derive a geometrical argument for the solution to the optimization problem in two dimensions, i.e., $d = 2$.

(d) Now solve the optimization problem for any arbitrary dimension d using Lagrange multipliers. Solve this optimization problem and comment on the form of the solution vector.

2. Recall the online Perceptron training algorithm. The perceptron learns a hyperplane defined by $\mathbf{w}_k$ at the k-th iteration, after observing the k-th sample $(\mathbf{x}_k, y_k)$. Initially, $\mathbf{w}_0$ is the zero vector. Classification for a new datapoint $\mathbf{x}$ is obtained by observing on which side of the hyperplane defined by $\mathbf{w}_k$ that $\mathbf{x}$ lies on, which can be read off the sign of the inner product $\mathbf{w_k}'\mathbf{x}$.

   At the $(k+1)$-st sample, the hyperplane $\mathbf{w}_{k+1}$ is updated from $\mathbf{w}_k$. Here, we consider an update scheme that is based on minimizing:

$$\mathbf{w}_{k+1} = \arg\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w} - \mathbf{w}_k\|_2^2$$

   subject to the constraint

$$\ell(\mathbf{w}, (\mathbf{x}_{k+1}, y_{k+1})) \leq 0$$

   where we consider ℓ as the *hinge loss*:

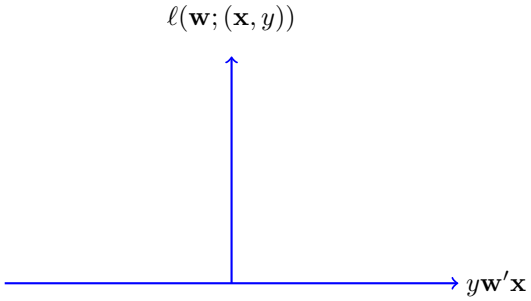
$$\ell(\mathbf{w}; (\mathbf{x}, y)) = \begin{cases} 0 & y\mathbf{w}'\mathbf{x} \geq 1 \\ 1 - y\mathbf{w}'\mathbf{x} & \text{otherwise.} \end{cases}$$

   (a) Look at the L_2 term in the optimization function. State in one sentence what it is trying to geometrically minimize.

   (b) Assume $\mathbf{w}_k$ classifies the $(k+1)$-th sample $(\mathbf{x}_{k+1}, y_{k+1})$ with $\ell(\mathbf{w}_k, (\mathbf{x}_{k+1}, y_{k+1})) = 0$, i.e., zero hinge loss. What is the choice for $\mathbf{w}_{k+1}$ and why?

   (c) How do we have to modify the feature space such that this classification model does not require an offset bias?

   (d) Is it possible for the classification hyperplane $\mathbf{w}_k$ to classify a point $\mathbf{x}$ correctly, but yet induce a non-zero hinge loss? To help justify your solution, please plot the hinge loss curve.



   (e) Compute the update rule for $\mathbf{w}_{k+1}$.

   (f) How can this algorithm be kernelized? (Hint: Expand your optimization solution $\mathbf{w}_{k+1}$, such that it only depends on $\mathbf{w}_0$ and the respective updates. What is the necessary and sufficient condition for an algorithm, such that it can be kernelized? Observe the structure of your solution and draw similarities from this to a kernelized solution.)

Problem 3 (Kernels and Computational Complexity):

A popular argument in favor of using kernels is that they can, in a certain sense, save a lot of computational effort. In this problem, we will convince ourselves that this may actually be the case.

First recall that a kernel function computes a scalar product in a high-dimensional space in the following sense: consider vectors in an input space $\mathbb{R}^d$. We define a mapping ϕ into another space, called the feature space, with dimension h:

$$\phi : \mathbb{R}^d \to \mathbb{R}^h \ . \tag{1}$$

The kernel k of two vectors $\mathbf{x}, \mathbf{y}$ in the input space is defined as

$$k(\mathbf{x}, \mathbf{y}) := \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle_{\mathbb{R}^h} \ . \tag{2}$$

Note:

- The kernel evaluates *both* the mapping ϕ *and* the scalar product.
- The scalar product is computed in the feature space $\mathbb{R}^h$.

The argument mentioned above is, in detail, the following: Assume that we wish to work with a high-dimensional feature space $\mathbb{R}^h$ (to improve the separability of the data). Using a kernel can save us a lot of computational effort, as compared to using the mapping ϕ and the scalar product $\langle\,.\,,\,.\,\rangle_{\mathbb{R}^h}$ explicitly.

To illustrate the argument, we will use a polynomial kernel, because this particular kernel allows us to derive explicit formulæ for the mapping ϕ. The kernel is defined as

$$k(\mathbf{x},\mathbf{y}) := \left(\langle \mathbf{x},\mathbf{y}\rangle_{\mathbb{R}^d} + 1\right)^m \ . \tag{3}$$

1. For $d = 2, m = 3$, find a formula for the mapping ϕ. What is the dimension h of the feature space defined by k?
2. For the general polynomial kernel (3), the number h of dimensions of the feature space is the number of all possible monomials of degree $\leq m$ in the d input entries $x_1, ..., x_d$. Derive the formula for this number of dimensions as a function of m and d. Whether you derive the formula by means of mathematics, literature research or search engine literacy is up to you.
   Hint: The number of monomials of *exactly* degree m in d input variables is given by the binomial coefficient $\binom{d+m-1}{m}$. The formula we are looking for is something quite similar.
3. Recall the digit classification problem: Each handwritten digit is scanned and rescaled to have size 16×16, so the input vector space is of dimension $d = 256$. How many monomial entries would we have to compute for each vector $\phi(\mathbf{x})$ if we choose $m = 2$ or $m = 4$? Compare your results to the number of operations necessary to evaluate the kernel function.

Kernel functions help to find a high dimensional linear discriminant where there is no linear discriminant in the original space.

4. Illustrate why the exclusive OR cannot be solved using a single linear discriminant operating on the features.
5. Define a suitable polynomial (quadratic) kernel to solve this problem.
6. Illustrate how the proposed kernel solves the problem.