Supervised Learning Coursework

John Shawe-Taylor Tom Diethe Dorota Glowacka

November 30, 2009; submission date: noon December 18, 2009

Abstract

Using a series of synthetic examples, in this exercise session you will acquaint yourselves with kernel ridge regression, fisher discriminant analysis, gaussian process regression, and normalising and centering of kernel matrices, the kernel adatron, which is an easy implementation of the support vector machine (it does not require a quadratic programming solver), and the bag of words kernel
Keywords: kernel ridge regression (kRR) — fisher discriminant analysis (FDA) — gaussian process regression (GPR) — normalising, centering — Kernel adatron — bag of words kernel.

1 Duality, the kernel trick, and nonlinear regression

1.1 Duality

As you proved in the last coursework, the optimal value $\mathbf{w}^*$ as obtained by Ridge Regression is given by $\mathbf{X}'\mathbf{X}\mathbf{w}^* + \gamma l\mathbf{w}^* = \mathbf{X}'\mathbf{y}$. ¿From this we can derive that: $\mathbf{w}^* = \mathbf{X}' \cdot \left[\frac{1}{l}(\mathbf{y} - \mathbf{X}\mathbf{w}^*)\right]$, showing that $\mathbf{w}$ can be expressed as a linear combination of the columns of $\mathbf{X}'$:

$$\mathbf{w}^* = \mathbf{X}'\alpha^*. \tag{1}$$

The vector $\alpha \in \Re^n$ will be referred to as the *dual vector*.

Using this dual vector, the so-called dual version of RR can be derived as follows:

$$\begin{aligned} \alpha^* &= \gamma l^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ \Rightarrow \gamma l\alpha^* &= (\mathbf{y} - \mathbf{X}\mathbf{X}'\alpha^*) \\ \Rightarrow (\mathbf{X}\mathbf{X}' + \gamma l\mathbf{I})\alpha^* &= \mathbf{y} \\ \Rightarrow \alpha^* &= (\mathbf{X}\mathbf{X}' + \gamma lI)^{-1}\mathbf{y}, \end{aligned} \tag{2}$$

The evaluation of the function $\mathbf{x}_{\text{test,i}}\mathbf{w}^*$ on a test sample can be rewritten as:

$$\mathbf{x}_{\text{test,i}}\mathbf{w}^* = \mathbf{x}_{\text{test,i}}\mathbf{X}'\alpha^*. \tag{3}$$

Thus far, it is not clear why such a formulation can be useful: the size of $\mathbf{X}\mathbf{X}' + \gamma l\mathbf{I}$ is equal to the number of samples, which is larger than the size of $\mathbf{X}'\mathbf{X} + \gamma l\mathbf{I}$ (equal to the dimensionality n) in all cases above, such that nothing would be gained, on the contrary. However, in the next exercise you will use this formulation in order to show the equivalence.

Exercise 1 *In this exercise you will perform the same Ridge Regression using the dual from equation (3)*

Pick a random weight vector $\mathbf{w} \in \Re^{10}$*, and generate a noisy random data set as* $y_i = \mathbf{x}_i'\mathbf{w} + n_i$ *containing 200 samples. Split the data set in training and test sets of size 100.*

Perform RR on the data set once with $\gamma = 10^{-6}$ *up to* $\gamma = 10^3$*, and all powers of* 10 *in between. What are the training and test mean squared errors? Plot the training and test set errors as a function of* γ *(use a log scale for the* γ *axis). Try different values for the noise* n_i *to see the effect. Can you see the effect of regularisation in high and low noise situations?*

1.2 Nonlinear regression

However, for nonlinear regression, the dual version will prove important. Let's start with the following example:

Exercise 2 *In this exercise, you will try to do regression on a data set that is generated by a nonlinear model. Randomly sample 200 points* $\mathbf{x}_i$ *(100 training samples and 100 test samples) in the interval* $[-4, 4]$ *(using the matlab function* `rand`*). Compute corresponding labels* $y_i = sinc(\mathbf{x}_i) + 0.3 \cdot n_i$*, where* n_i *is a random real number generated by the matlab function* `randn` *as before. Perform LSR and RR on this data set. Give a table of results and explain.*

Obviously, linear regression is not capable of achieving a good predictive performance on such a nonlinear data set. Here, we the dual formulation will prove extremely useful, in combination with the *kernel trick*.

This kernel trick is based on nothing more than the observation that equation to compute α^* (equation (2)) as well as the equation to evaluate the regression function (equation (3)) both only contain the vectors $\mathbf{x}_i$ in inner products with each other. Therefore, it is sufficient to know these inner products only, instead of the actual vectors $\mathbf{x}_i$.

As a result, we can also work with inner products between *nonlinear* mappings $\phi : \mathbf{x}_i \rightarrow \phi(\mathbf{x}_i) \in \mathcal{F}$ of $\mathbf{x}_i$ into a so-called *feature space* $\mathcal{F}$, as long as the inner product $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)'\phi(\mathbf{x}_j)$ can be evaluated efficiently. In many cases, this inner product (from now on called the *kernel function*) can be evaluated much more efficiently than the feature vector itself, which can even be

infinite dimensional in principle. A commonly used kernel function for which this is the case is the radial basis function (RBF) kernel, which is defined as:

$$K_{\mathrm{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2L^2}\right). \tag{4}$$

Using the notation $\mathbf{K} = \mathbf{X}\mathbf{X}'$ for the *kernel matrix* or *Gram matrix* containing the inner products between all training points, we can rewrite equation (2) in its kernel formulation (kRR):

$$\alpha^* = (\mathbf{K} + \gamma l \mathbf{I})^{-\mathbf{1}}\mathbf{y}. \tag{5}$$

The evaluation of the regression function on a test point can be reformulated as:

$$y_{\mathrm{test}} = \sum_{i=1}^{l} K(\mathbf{x}_i, \mathbf{x}_{\mathrm{test}}) \cdot \alpha_i^*. \tag{6}$$

where the K is the kernel function.

Effectively, by using such a kernel function corresponding to a nonlinear feature map, a linear regression method such as RR can be carried out in the feature space, amounting to a nonlinear regression function (equation 6) in the original input space.

Exercise 3 (kRR) *Kernel RR applied to the sinc problem.*

a. *Perform kRR on the data set generated in the previous exercise, with* $L = 0.5$. *Use* 5*-fold cross validation to tune the regularization parameter. Plot the regression function in the interval* $[-4, 4]$. *Note that in fact we can also tune* L *in this way.*

b. *Why is regularization so important in this case?*

In summary, by using the kernel trick, we can apply simple techniques for linear regression in the *feature space* to perform nonlinear regression in the *input space*.

1.3 Gaussian Process (GP) regression

We can also tackle the regression problem in the Bayesian way. We assume that the outputs $\mathbf{y}$ are generated by corrupting an (unknown) true function f with independent Gaussian noise ξ, ie

$$y_i = f(\mathbf{x}_i) + \xi_i \tag{7}$$

for $i = 1, \ldots, l$. The (also unknown) variance of the noise is called σ^2. Hence, for a fixed vector of function values $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_l))$, the data $\mathbf{y}$ have the likelihood

$$p(\mathbf{y}|\mathbf{f}) \propto \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{l}(y_i - f(\mathbf{x}_i))^2\right]. \tag{8}$$

To regularise in the Bayesian approach we assume a Gaussian process prior with covariance kernel K over the space of functions f. This induces a joint Gaussian prior density $p(\mathbf{f}_+)$ over augmented function values $\mathbf{f}_+ = (\mathbf{f}, f(\mathbf{x}_{\text{test}}))^T$. Here $\mathbf{x}_{\text{test}}$ is the test point where we would like to make a prediction.

$$p(\mathbf{f}_+) \propto |\det(\mathbf{K}_+)|^{-1/2} \exp\left[-\frac{1}{2}\mathbf{f}_+^T\mathbf{K}_+^{-1}\mathbf{f}_+\right] \tag{9}$$

with $\mathbf{k} = (K(\mathbf{x}_{\text{test}}, \mathbf{x}_1), K(\mathbf{x}_{\text{test}}, \mathbf{x}_2), \ldots, K(\mathbf{x}_{\text{test}}, \mathbf{x}_l))^T$ and the augmented kernel matrix

$$\tilde{\mathbf{K}} = \begin{pmatrix} \mathbf{K} & \mathbf{k} \\ \mathbf{k}^T & K(\mathbf{x}_{\text{test}}, \mathbf{x}_{\text{test}}) \end{pmatrix}, \tag{10}$$

using the definition of the kernel matrix $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.

We would like to base our prediction on the posterior density

$$p(\mathbf{f}_+|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}_+)}{p(\mathbf{y})} . \tag{11}$$

by calculating the *maximum posterior* (MAP) prediction, ie

$$\mathbf{f}_+^{\text{MAP}} = \text{argmax} \ln p(\mathbf{f}_+|\mathbf{y}) . \tag{12}$$

Since $p(\mathbf{f}_+|\mathbf{y})$ is a quadratic form in $\mathbf{f}_+$ one can get the prediction explicitly. With a bit of matrix algebra one ends up with the result

$$f^{\text{MAP}}(\mathbf{x}_{\text{test}}) = \mathbf{k}^T \left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1} \mathbf{y} . \tag{13}$$

A comparison with (6) shows that this is the same expression as found for kernel RR, when we identify $\gamma l \equiv \sigma^2$.

It is also possible to find the *posterior uncertainty* for this prediction given by the posterior variance (Bayesian error bar)

$$\sigma^2_{\text{error}}(\mathbf{x}_{\text{test}}) = K(\mathbf{x}_{\text{test}}, \mathbf{x}_{\text{test}}) - \mathbf{k}^T \left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1} \mathbf{k} \tag{14}$$

The Bayesian approach allows us to give a sensible estimate for σ^2 without cross validation by choosing it to maximise the probability of the data (evidence) which is given by

$$p(\mathbf{y}) = \int d\mathbf{f} p(\mathbf{f}) \; p(\mathbf{y}|\mathbf{f}) . \tag{15}$$

This is a big multidimensional Gaussian integral which can be performed analytically by hand, but also easier, when one thinks a bit probabilistically. If we put a Gaussian prior over $\mathbf{f}$ with covariance $\mathbf{K}$ and also adding white Gaussian noise to $\mathbf{f}$ (with covariance $\sigma^2\mathbf{I}$) according to (7), the vector $\mathbf{y}$ becomes jointly Gaussian with zero mean and covariance matrix $\mathbf{K} + \sigma^2\mathbf{I}$. Hence

$$p(\mathbf{y}) = \frac{1}{(2\pi)^{l/2}|\det(\mathbf{K} + \sigma^2\mathbf{I})|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}\right] \tag{16}$$

Exercise 4 (GP) *GP regression applied to the sinc problem.*

a. *Perform GP regression on the data set generated in the previous exercise, with a true noise variance* $\sigma_0^2 = 0.3^2$. *Again use an RBF kernel of the form*

$$K_{RBF}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2L^2}\right) , \tag{17}$$

with $L = 0.5$. *This time tune the unknown parameter* σ *by calculating the log evidence* $\ln p(\mathbf{y})$ *using (16). Plot this for a range of values for* σ^2 *(with fixed data, ie fixed noise* σ_0*).*

b. *Plot the regression function in the interval* $[-4, 4]$. *Give a measure for the (Bayesian) uncertainty of your prediction by computing* σ^2_{error} *using (14). Plot two extra lines which are one standard deviation* σ_{error} *above and below your prediction. Don't confuse the different* σ*'s!*

c. *(bonus) Repeat the experiment described in (a) but varying the width* L *of the Gaussian kernel while leaving* σ *fixed at the value maximising the log evidence in (a).*

2 Normalizing and centering

Normalizing a kernel matrix In some cases, it is a good idea to normalize all samples to unit norm. This can be done easily on the sample matrix $\mathbf{X}$ itself:

$$\mathbf{X}_n = \begin{pmatrix} \mathbf{x}_1'\mathbf{x}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{x}_2'\mathbf{x}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{x}_l'\mathbf{x}_l \end{pmatrix}^{-1/2} \cdot \mathbf{X}. \tag{18}$$

Note that the diagonal matrix has $\mathbf{x}_i'\mathbf{x}_i = \mathbf{K}_{ii}$ on its diagonal. Again, it turns out we can also compute the normalized kernel matrix $\mathbf{K}_n$ corresponding to the kernel $\mathbf{K}$, without ever needing to know the data matrix $\mathbf{X}$ itself:

$$\begin{aligned} \mathbf{K}_n &= \mathbf{X}_n\mathbf{X}_n' \\ &= \operatorname{diag}(\mathbf{x}_1\mathbf{x}_1'\ \mathbf{x}_2\mathbf{x}_2'\ \ldots\ \mathbf{x}_l\mathbf{x}_l')^{-1/2} \cdot \mathbf{X}\mathbf{X}' \cdot \operatorname{diag}(\mathbf{x}_1\mathbf{x}_1'\ \mathbf{x}_2\mathbf{x}_2'\ \ldots\ \mathbf{x}_l\mathbf{x}_l')^{-1/2} \\ &= \operatorname{diag}(\mathbf{K}_{11}\,\mathbf{K}_{22}\ \ldots\ \mathbf{K}_{ll})^{-1/2} \cdot \mathbf{K} \cdot \operatorname{diag}(\mathbf{K}_{11}\,\mathbf{K}_{22}\ \ldots\ \mathbf{K}_{ll})^{-1/2} \end{aligned} \tag{19}$$

It's easy to check that all diagonal entries of $\mathbf{K}_n$, which are the norms of the normalized samples $\mathbf{X}_n$, are equal to 1 indeed.

Centering a kernel matrix Given a sample $\mathbf{X}$, centering can be performed by subtracting the means of the features of $\mathbf{X}$ (corresponding to columns of $\mathbf{X}$):

$$\mathbf{X}_c = \mathbf{X} - \frac{1}{l}\mathbf{1}\mathbf{1}'\mathbf{X} \tag{20}$$

Thus, the matrix containing all inner products of the centered vectors $\mathbf{X}_c$ is

$$\mathbf{X}_c\mathbf{X}_c' = (\mathbf{X} - \frac{1}{l}\mathbf{1}\mathbf{1}'\mathbf{X})(\mathbf{X} - \frac{1}{l}\mathbf{1}\mathbf{1}'\mathbf{X})'$$

which is equal to

$$\mathbf{X}_c\mathbf{X}_c' = \mathbf{X}\mathbf{X}' - \frac{1}{l}\mathbf{1}\mathbf{1}'\mathbf{X}\mathbf{X}' - \frac{1}{l}\mathbf{X}\mathbf{X}'\mathbf{1}\mathbf{1}' + \frac{1}{l^2}\mathbf{1}\mathbf{1}'\mathbf{X}\mathbf{X}'\mathbf{1}\mathbf{1}'.$$

Also here we can use the kernel trick and replace $\mathbf{X}\mathbf{X}'$ by the kernel matrix $\mathbf{K}$. Thus, the centered kernel matrix $\mathbf{K}_c$ can be computed directly from the uncentered kernel matrix as:

$$\mathbf{K}_c' = \mathbf{K} - \frac{1}{l}\mathbf{1}\mathbf{1}'\mathbf{K} - \frac{1}{l}\mathbf{K}\mathbf{1}\mathbf{1}' + \frac{1}{l^2}\mathbf{1}\mathbf{1}'\mathbf{K}\mathbf{1}\mathbf{1}' \tag{21}$$

Exercise 5 (Normalizing and centering) *Load the file* `docs.mat` *into matlab. As you can see, it contains 4 cell arrays, each of which contains 195 little texts in either english, french, german or italian (articles from the Swiss constitution...). You can verify that the ith text in one language is a translation of the ith text in the other languages. In this practical session we will work with this data set.*

Now, there exist kernels specially designed for strings and texts that compute a certain similarity measure between two texts. Two of these kernels are the bag-of-words kernel *and the* 2-mer kernel. *Since computing them on that* $780 = 4{\cdot}195$ *texts would take a while, you don't have to do that here. Just load the two kernels K_bow and K_2mer into matlab from the file Ks.mat. These are the two kernel matrices evaluated on all documents of all languages, and the ordering in which the documents appear is: english, french, german and italian, each language in the same ordering (so for an english document at position* $1 \leq i \leq 195$*, the german translation occurs at position* $2 \times 195 + i$*).*

a. *Have a look at both kernels by printing part of them on the screen. Do you notice that small texts generally give rise to smaller kernel values? This is undesirable in many applications: very often, the similarity between two texts should not be based on their length, but rather on how much they overlap relative to their length. To get rid of this artefact, you can normalize the kernels. Do this, and save the resulting normalized kernels as K_bow_n and K_2mer_n.*

b. *As a second preprocessing step, center the kernels and save the results as K_bow_nc and K_2mer_nc. Show that the preprocessing has been successful by calculating the norms and means for the kernels.*

3 The kernel adatron

The kernel adatron is a variant of the support vector machine for classification. The object function is exactly the same, but the bias term is omitted from the

constraints. The resulting dual problem is

$$\begin{array}{ll} \max\limits_{\alpha} & f(\alpha) = \mathbf{1}'\alpha - \dfrac{1}{2}\alpha'(\mathbf{K} \odot (\mathbf{y}\mathbf{y}'))\alpha \\ \text{s.t.} & 0 \leq \alpha \leq C \end{array} \tag{22}$$

where $\odot$ is the elementwise matrix product. The result is that the optimization problem is much simpler, and can be carried out using simple coordinate ascent:

- Equate $\alpha = \mathbf{0}$.
- Iterate (something like $10 \cdot l$ times if l is the number of samples):
  1. Compute the derivatives $\mathbf{g}_i = \frac{\mathrm{d}f(\alpha)}{\mathrm{d}\alpha_i} = 1 - (\mathbf{K} \odot (\mathbf{y}\mathbf{y}'))\alpha$ of the object function with respect to all i.
  2. For all i, compute
  $$\beta_i = \min\left(\max\left(\alpha_i + \eta \mathbf{g}_i, 0\right), C\right).$$
  3. Find α_i, for which $|\alpha_i - \beta_i|$ is maximal.
  4. Equate this (and only this) α_i to β_i.

Even though this standard version of the kernel adatron would work, modifications have been made to accommodate a bias term b. The resulting algorithm is then:

- Equate $\alpha = \mathbf{0}$ and $b = 0$.
- Iterate (something like $10 \cdot l$ times if l is the number of samples):
  1. Compute $\mathbf{z} = \mathbf{K}(\mathbf{y} \odot \alpha) + b$. As you can see, this vector contains the predictions of the training set labels if we would use α and b.
  2. Compute $\mathbf{g} = 1 - \mathbf{y} \odot \mathbf{z}$, containing one minus the predicted labels times the corresponding true labels. (Ideally, when all training labels are predicted correctly, this is the zero vector. If the ith element is negative, this means that α_i should decrease a bit, and vice versa. It corresponds to the gradient in the basic kernel adatron algorithm above.)
  3. For all i, compute
  $$\beta_i = \min\left(\max\left(\alpha_i + \eta g_i, 0\right), C\right).$$
  4. Find α_i, for which $|\alpha_i - \beta_i|$ is maximal. If change less than 0.005 then exit.
  5. Equate this (and only this) α_i to β_i.

6. Then, update b as:

$$b = b + \frac{\mathbf{y}'\alpha}{4}.$$

Exercise 6 *Implement the kernel adatron, and use it to classify english against french languages in the swiss constitution data set, using the 2-mer kernel. Tune the regularization parameter C using cross-validation. Use $\eta = 1$ for the step size.*

4 The bag of words kernel

The bag of words kernel evaluated between two documents is nothing more than:

$$\sum_{\texttt{word} \in \texttt{dictionary}} (\#\text{occ. of } \texttt{word} \text{ in } \texttt{text 1}) \times (\#\text{occ. of } \texttt{word} \text{ in } \texttt{text 2}).$$

Here `dictionary` is the union of all words occurring in any of the texts.

Even though this looks extremely simple, implementing it such that it can be evaluated *efficiently* on a large number of texts is more difficult.

Exercise 7 *Implement the bag of words kernel. Use the matlab help to learn about string processing in matlab (*`help strfun`*). Test it on the documents of the swiss constitution using the kernel adatron. Compare your results with the precomputed kernels and give running times for computing the kernel. (Use the functions* `tic` *and* `toc`*. A good implementation in matlab requires less than 5 minutes on a modern pc... Note: you should obtain a somewhat different kernel from the one you have been using in the previous practical sessions, because that kernel was computed after stemming and stop word removal.)*

5 Exam style question

a) The Perceptron algorithm applied to the training set

$$((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$$

involves the following training loop:

```
  for i = 1 to m do
if y_i⟨w, x_i⟩ ≤ 0 then
w ← w + y_i x_i ;
end
end
```

If $\mathbf{w}$ is initialised to 0, show that $\mathbf{w}$ can be expressed as a linear combination:

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i,$$

of the training data, explaining what the value of α_i will be at any stage of the algorithm.

b) Show we can evaluate the classification $\text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle)$ on a test example $\mathbf{x}$ using the values of α_i and inner products between $\mathbf{x}$ and the training data $\langle \mathbf{x}, \mathbf{x}_i \rangle$, for $i = 1, \ldots, m$. Hence, show how we can run the Perceptron algorithm in a feature space defined by a mapping

$$\phi : \mathbf{x} \longmapsto \phi(\mathbf{x})$$

using only the kernel function

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle.$$

c) For inseparable data the Perceptron algorithm will go into an infinite loop. Assuming the training examples are distinct, consider the modification of a normalised kernel κ to

$$\tilde{\kappa}(\mathbf{x}, \mathbf{z}) = \kappa(\mathbf{x}, \mathbf{z}) + a\delta(\mathbf{x}, \mathbf{z}),$$

where $a > 0$ and

$$\delta(\mathbf{x}, \mathbf{z}) = \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{z}, \\ 0 & \text{otherwise.} \end{cases}$$

Show we can find α such that $\tilde{K}\alpha = \mathbf{y}$ for the label vector $\mathbf{y}$, where $\tilde{K}$ is the kernel matrix of the modified kernel $\tilde{\kappa}$. Hence, argue that the training set is separable in the feature space defined by the mapping $\tilde{\phi}$ of the kernel $\tilde{\kappa}$.

d) What is the functional margin of the weight vector defined in (c)? Hence, give an upper bound on the number of updates the Perceptron algorithm will make using the kernel $\tilde{\kappa}$.