

Fisher Kernel

Martin Sewell

Department of Computer Science
University College London

April 2007 (last updated September 2008)

1 Introduction

In common with all kernel methods, the support vector machine technique involves two stages: first non-linearly map the input space into a very high dimensional feature space, then apply a learning algorithm designed to discover linear patterns in that space. This paper concerns the first stage. The basic idea is to train a (generative) hidden Markov model on market data to derive a Fisher kernel for a (discriminative) support vector machine. If each data point is a (possibly varying length) sequence, each may be used to train a probability model (HMM), with the average of the models in the training set used to construct a global HMM. It is then possible to calculate how much a new data point would ‘stretch’ the parameters of the global model.

2 Literature Review

Jaakkola and Haussler (1999a) introduced the Fisher kernel (named in honour of Sir Ronald Fisher), thus creating a generic mechanism for incorporating generative probability models into discriminative classifiers such as SVMs. Jaakkola and Haussler (1999b) introduced a generic class of probabilistic regression models and a parameter estimation technique that can make use of arbitrary kernel functions. Jaakkola, Diekhans and Haussler (1999) applied the Fisher kernel method to detecting remote protein homologies which performed well in classifying protein domains by SCOP superfamily. Jaakkola, Diekhans and Haussler (2000) found that using the Fisher kernel significantly improved on previous methods for the classification of protein domains based on remote homologies. Moreno and Rifkin (2000) used the Fisher kernel method for large scale Web audio classification. Vinokourov and Girolami (2001) successfully employed the Fisher kernel for document classification. Saunders, Shawe-Taylor and Vinokourov (2003) showed how the string kernel can be thought of as a k -stage Markov process, and as a result interpreted as a Fisher kernel. Tsuda, *et al.* (2004) analysed the statistical properties of the Fisher kernel. Nicotra, Micheli and Starita

(2004) extended the Fisher kernel to deal with tree structured data. Kersting and Gärtner (2004) extend the Fisher kernel to logical sequences (sequences over an alphabet of logical atoms). Their experiments showed a considerable improvement over results achieved without Fisher kernels for logical sequences. Holub, Welling and Perona (2005) successfully combined generative models with Fisher kernels to realize performance gains on standard object recognition datasets. Dick and Kersting (2006) developed Fisher kernels for relational data.

3 Markov Chains

Markov chains were introduced by the Russian mathematician Andrey Markov in 1906 (Markov 1906), although the term did not appear for over 20 years when it was used by Bernstein (Bernstein 1927). A *Markov chain* is a discrete-state Markov process (see Sewell (2006)).

Formally, a discrete time Markov chain is a sequence of n random variables $X_n, n \geq 0$ such that for every n $P(X_{n+1} = x | X_0 = x_0, X_1 = x_1, \dots, X_n = x_n) = P(X_{n+1} = x | X_n = x_n)$.

In words, the future of the system depends on the present, but not the past.

4 Hidden Markov Models

A *hidden Markov model* (HMM) is a temporal probabilistic model in which the state of the process is described by a single discrete random variable. Loosely speaking, it is a Markov chain observed in noise. The theory of hidden Markov models was developed in the late 1960s and early 1970s by Baum, Eagon, Petrie, Soules and Weiss (Baum and Petrie 1966; Baum and Eagon 1967; Baum, *et al.* 1970; Baum 1972), whilst the name ‘hidden Markov model’ was coined by L. P. Neuwirth. For more information on HMMs, see the tutorial papers Rabiner and Juang (1986); Poritz (1988); Rabiner (1989); Eddy (2004) and the books MacDonald and Zucchini (1997); Durbin, *et al.* (1999); Elliot, Aggoun and Moore (2004); Cappé, Moulines and Rydén (2005). HMMs have earned their popularity largely from successful application to speech recognition (Rabiner 1989), but have also been applied to handwriting recognition, gesture recognition, musical score following and bioinformatics.

Formally, a hidden Markov model is a bivariate discrete time process $\{X_k, Y_k\}_{k \geq 0}$, where X_k is a Markov chain and, conditional on X_k , Y_k is a sequence of independent random variables such that the conditional distribution of Y_k only depends on X_k .

The successful application of HMMs to markets is referenced as far back as Kemeny, Snell and Knapp (1976) and Juang (1985). The books Bhar and Hamori (2004) and Mamon and Elliott (2007) cover HMMs in finance.

5 Fixed Length Strings Generated by a Hidden Markov Model

Parts of the final chapter of Shawe-Taylor and Cristianini (2004)—which covers turning generative models into kernels—are followed below.

Let us assume that one has two strings s and t of fixed length n that are composed of symbols from an alphabet Σ . Furthermore it is assumed that they have been generated by a hidden model M , whose elements are represented by strings h of n states each from a set A , and that each symbol is generated independently, so that

$$P(s, t|h) = \prod_{i=1}^n P(s_i|h_i)P(t_i|h_i).$$

Consider the hidden Markov model

$$P_M(h) = P_M(h_1)P_M(h_2|h_1) \dots P_M(h_n|h_{n-1}).$$

Define the states of the model to be

$$\{a_I\} \cup A \times \{1, \dots, n\} \cup a_F,$$

with the transition probabilities given by

$$\begin{aligned} P_M((a, i)|a_I) &= \begin{cases} P_M(a) & \text{if } i = 1; \\ 0 & \text{otherwise,} \end{cases} \\ P_M((a, i)|(b, j)) &= \begin{cases} P_M(a|b) & \text{if } i = j + 1; \\ 0 & \text{otherwise,} \end{cases} \\ P_M(a_F|(b, j)) &= \begin{cases} 1 & \text{if } i = n; \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

This means that in order to marginalise, one needs to sum over a more complex probability distribution for the hidden states to obtain the corresponding marginalisation kernel

$$\begin{aligned} \kappa(s, t) &= \sum_{h \in A^n} P(s|h)P(t|h)P_M(h) \\ &= \sum_{h \in A^n} \prod_{i=1}^n P(s_i|h_i)P(t_i|h_i)P_M(h_i|h_{i-1}), \end{aligned} \quad (1)$$

where the convention that $P_M(h_1|h_0) = P_M(h_1)$ has been used.

Each hidden sequence h is considered as a template for the sequences s , t in the sense that if it is in state h_i at position i , the probability that the observable sequence has a symbol s_i in that position is a function of h_i . In

the generative model, sequences are generated independently from the hidden template with probabilities $P(s_i|h_i)$ that can be specified by a matrix of size $|\Sigma| \times |A|$. So given this matrix and a fixed h , one can compute $P(s|h)$ and $P(t|h)$. The problem is that there are $|A|^n$ different possible models for generating the sequences s, t , that is the feature space is spanned by a basis of $|A|^n$ dimensions. Furthermore, a special generating process for h of Markov type, the probability of a state depends only on the preceding state, is considered. The consequent marginalisation step will therefore be prohibitively expensive, if performed in a direct way. Dynamic programming techniques will be exploited to speed it up.

Consider the set of states A_a^k of length k that end with a given by

$$A_a^k = \{h \in A^k : h_k = a\}.$$

A series of subkernels $\kappa_{k,a}$ for $k = 1, \dots, n$ and $a \in A$ are introduced as follows

$$\begin{aligned} \kappa_{k,a}(s, t) &= \sum_{h \in A_a^k} P(s|h)P(t|h)P_M(m) \\ &= \sum_{h \in A_a^k} \prod_{i=1}^k P(s_i|h_i)P(t_i|h_i)P_M(h_i|h_{i-1}), \end{aligned}$$

where the definitions of $P(s|h)$ and $P(h)$ have been implicitly extended to cover the case when h has fewer than n symbols by ignoring the rest of the string s .

Clearly, the HMM kernel can be expressed simply by

$$\kappa(s, t) = \sum_{a \in A} \kappa_{n,a}(s, t).$$

For $k = 1$ one has

$$\kappa_{1,a}(s, t) = P(s_1|a)P(t_1|a)P_M(a).$$

Recursive equations for computing $\kappa_{k+1,a}(s, t)$ in terms of $\kappa_{k,b}(s, t)$ for $b \in A$ are now obtained, as the following derivation shows

$$\begin{aligned} \kappa_{k+1,a}(s, t) &= \sum_{h \in A_a^{k+1}} \prod_{i=1}^{k+1} P(s_i|h_i)P(t_i|h_i)P_M(h_i|h_{i-1}) \\ &= \sum_{b \in A} P(s_{k+1}|a)P(t_{k+1}|a)P_M(a|b) \sum_{h \in A_b^k} \prod_{i=1}^k P(s_i|h_i)P(t_i|h_i)P_M(h_i|h_{i-1}) \\ &= \sum_{b \in A} P(s_{k+1}|a)P(t_{k+1}|a)P_M(a|b)\kappa_{k,b}(s, t). \end{aligned}$$

When computing these kernels the usual dynamic programming tables, one for each $\kappa_{k,b}(s, t)$, need to be used, though of course those obtained for $k - 1$ can be overwritten when computing $k + 1$. The result is summarised in the following algorithm.

Input	Symbol strings s and t , state transition probability matrix $P_M(a b)$, initial state probabilities $P_M(a)$ and conditional probabilities $P(\sigma a)$ of symbols given states.
Process	<pre> Assume p states, $1, \dots, p$. for $a = 1 : p$ DPr(a) = $P(S_1 a)P(t_1 a)P_M(a)$; end for $i = 1 : n$ Kern = 0; for $a = 1 : p$ DP(a) = 0; for $b = 1 : p$ DP(a) = DP(a) + $P(s_i a)P(t_i a)P_M(a b)DPr(b)$; end Kern = Kern + DP(a); end DPr = DP; end </pre>
Output	$\kappa(s, t) = \text{Kern}$

The complexity of the kernel can be bounded from the structure of the algorithm by

$$O(n|A|^2).$$

6 Fisher Kernel

The *log-likelihood* of a data item x with respect to the model $m(\boldsymbol{\theta}^0)$ for a given setting of the parameters $\boldsymbol{\theta}^0$ is defined to be

$$\log \mathcal{L}_{\boldsymbol{\theta}^0}(x).$$

Consider the vector gradient of the log-likelihood

$$\mathbf{g}(\boldsymbol{\theta}, x) = \left(\frac{\partial \log \mathcal{L}_{\boldsymbol{\theta}}(x)}{\partial \theta_i} \right)_{i=1}^N.$$

The *Fisher score* of a data item x with respect to the model $m(\boldsymbol{\theta}^0)$ for a given setting of the parameters $\boldsymbol{\theta}^0$ is

$$\mathbf{g}(\boldsymbol{\theta}^0, x).$$

The *Fisher information matrix* with respect to the model $m(\boldsymbol{\theta}^0)$ for a given setting of the parameters $\boldsymbol{\theta}^0$ is given by

$$\mathbf{I}_M = \mathbb{E} \left[\mathbf{g}(\boldsymbol{\theta}^0, x) \mathbf{g}(\boldsymbol{\theta}^0, x)' \right],$$

where the expectation is over the generation of the data point x according to the data generating distribution.

The Fisher score gives us an embedding into the feature space \mathbb{R}^N and hence immediately suggests a possible kernel. The matrix \mathbf{I}_M can be used to define a non-standard inner product in that feature space.

Definition 1 *The invariant Fisher kernel with respect to the model $m(\boldsymbol{\theta}^0)$ for a given setting of the parameters $\boldsymbol{\theta}^0$ is defined as*

$$\kappa(x, z) = \mathbf{g}(\boldsymbol{\theta}^0, x)' \mathbf{I}_M^{-1} \mathbf{g}(\boldsymbol{\theta}^0, z).$$

The practical Fisher kernel is defined as

$$\kappa(x, z) = \mathbf{g}(\boldsymbol{\theta}^0, x)' \mathbf{g}(\boldsymbol{\theta}^0, z).$$

The Fisher kernel gives a ‘natural’ similarity measure that takes into account an underlying probability distribution. It seems natural to compare two data points through the directions in which they ‘stretch’ the parameters of the model, that is by viewing the score function at the two points as a function of the parameters and comparing the two gradients. If the gradient vectors are similar it means that the two data items would adapt the model in the same way, that is from the point of view of the given parametric model at the current parameter setting they are similar in the sense that they would require similar adaptations to the parameters.

7 Fisher Kernels for Hidden Markov Models

The model can now be viewed as the sum over all of the state paths or individual models with the parameters the various transition and emission probabilities, so that for a particular parameter setting the probability of a sequence s is given by

$$P_M(s) = \sum_{m \in A^n} P(s|m) P_M(m) = \sum_{m \in A^n} P_M(s, m),$$

where

$$P_M(m) = P_M(m_1) P_M(m_2|m_1) \dots P_M(m_n|m_{n-1}),$$

and

$$P(s|m) = \prod_{i=1}^n P(s_i|m_i)$$

so that

$$P_M(s, m) = \prod_{i=1}^n P(s_i|m_i) P(m_i|m_{i-1}).$$

The parameters of the model are the emission probabilities $P(s_i|m_i)$ and the transition probabilities $P_M(m_i|m_{i-1})$. For convenience parameters are introduced

$$\theta_{s_i|m_i} = P(s_i|m_i) \text{ and } \tau_{m_i|m_{i-1}} = P_M(m_i|m_{i-1}),$$

where the convention that $P_M(m_1) = P_M(m_1|m_0)$ with $m_0 = a_0$ for a special fixed state $a_0 \notin A$ is used. The difficulty is that these parameters are not independent. The unconstrained parameters are introduced

$$\theta_{\sigma,a} \text{ and } \tau_{a,b}$$

with

$$\theta_{\sigma|a} = \frac{\theta_{\sigma,a}}{\sum_{\sigma' \in \Sigma} \theta_{\sigma',a}} \text{ and } \tau_{a|b} = \frac{\tau_{a,b}}{\sum_{a' \in A} \tau_{a',b}}$$

These values are assembled into a parameter vector $\boldsymbol{\theta}$. Furthermore it is assumed that the parameter setting at which the derivatives are computed satisfies

$$\sum_{\sigma \in \Sigma} \theta_{\sigma,a} = \sum_{a \in A} \tau_{a,b} = 1, \quad (2)$$

for all $a, b \in A$ in order to simplify the calculations.

The derivatives of the log-likelihood with respect to the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\tau}$ must be computed. The computations for both sets of parameters follow an identical pattern, so to simplify the presentation first a template that assumes both cases is derived. Let

$$\bar{\psi}(b, a) = \frac{\psi(b, a)}{\sum_{b' \in B} \psi(b', a)}, \text{ for } a \in A \text{ and } b \in B.$$

Let

$$Q(\mathbf{a}, \mathbf{b}) = \prod_{i=1}^n \bar{\psi}(b_i, a_i) c_i,$$

for some constants c_i . Consider the derivative of $Q(\mathbf{a}, \mathbf{b})$ with respect to the parameter $\psi(b, a)$ at point $(\mathbf{a}^0, \mathbf{b}^0)$ where

$$\sum_{b \in B} \psi(b, a_i^0) = 1 \text{ for all } i. \quad (3)$$

One has

$$\begin{aligned} \frac{\partial Q(\mathbf{a}, \mathbf{b})}{\partial \psi(b, a)} &= \sum_{k=1}^n c_k \prod_{i \neq k} \bar{\psi}(b_i^0, a_i^0) c_i \frac{\partial}{\partial \psi(b, a)} \frac{\psi(b_k, a_k)}{\sum_{b' \in B} \psi(b', a_k)} \\ &= \sum_{k=1}^n \left(\frac{[b_k^0 = b][a_k^0 = a]}{\sum_{b' \in B} \psi(b', a_k^0)} - \frac{\psi(b_k^0, a_k^0)[a_k^0 = a]}{(\sum_{b' \in B} \psi(b', a_k^0))^2} \right) c_k \prod_{i \neq k} \bar{\psi}(b_i^0, a_i^0) c_i \\ &= \sum_{k=1}^n \left(\frac{[b_k^0 = b][a_k^0 = a]}{\bar{\psi}(b, a)} - [a_k^0 = a] \right) \prod_{i=k} \bar{\psi}(b_i^0, a_i^0) c_i \\ &= \sum_{k=1}^n \left(\frac{[b_k^0 = b][a_k^0 = a]}{\bar{\psi}(b, a)} - [a_k^0 = a] \right) Q(\mathbf{a}^0, \mathbf{b}^0), \end{aligned}$$

where use of (3) has been made to obtain the third line from the second. Now return to considering the derivatives of the log-likelihood, first with respect to the parameter $\theta_{\sigma,a}$

$$\begin{aligned}\frac{\partial \log P_M(s|\boldsymbol{\theta})}{\partial \theta_{\sigma,a}} &= \frac{1}{P_M(s|\boldsymbol{\theta})} \frac{\partial}{\partial \theta_{\sigma,a}} \sum_{m \in A^n} \prod_{i=1}^n P(s_i|m_i) P_M(m_i|m_{i-1}) \\ &= \frac{1}{P_M(s|\boldsymbol{\theta})} \sum_{m \in A^n} \frac{\partial}{\partial \theta_{\sigma,a}} \prod_{i=1}^n \frac{\theta_{s_i,m_i}}{\sum_{\sigma \in \Sigma} \theta_{\sigma,m_i}} \tau_{m_i|m_{i-1}}.\end{aligned}$$

Letting \mathbf{a} be the sequence of states m and \mathbf{b} the string s , with $\psi(a,b) = \theta_{b,a}$ and $c_i = \tau_{m_i|m_{i-1}}$ one has

$$Q(\mathbf{a}, \mathbf{b}) = \prod_{i=1}^n \frac{\theta_{s_i,m_i}}{\sum_{\sigma \in \Sigma} \theta_{\sigma,m_i}} \tau_{m_i|m_{i-1}} = P_M(s, m|\boldsymbol{\theta}).$$

It follows from the derivative of Q that

$$\begin{aligned}\frac{\partial \log P_M(s|\boldsymbol{\theta})}{\partial \theta_{\sigma,a}} &= \sum_{m \in A^n} \sum_{k=1}^n \left(\frac{[s_k = \sigma][m_k = a]}{\theta_{\sigma|a}} - [m_k = a] \right) \frac{P_M(s, m|\boldsymbol{\theta})}{P_M(s|\boldsymbol{\theta})} \\ &= \sum_{k=1}^n \sum_{m \in A^n} \left(\frac{[s_k = \sigma][m_k = a]}{\theta_{\sigma|a}} - [m_k = a] \right) P_M(m|s, \boldsymbol{\theta}) \\ &= \sum_{k=1}^n \mathbb{E} \left[\frac{[s_k = \sigma][m_k = a]}{\theta_{\sigma|a}} - [m_k = a] \middle| s, \boldsymbol{\theta} \right] \\ &= \frac{1}{\theta_{\sigma|a}} \sum_{k=1}^n \mathbb{E}[s_k = \sigma][m_k = a|s, \boldsymbol{\theta}] - \sum_{k=1}^n \mathbb{E}[m_k = a|s, \boldsymbol{\theta}],\end{aligned}$$

where the expectations are over the hidden states that generate s . Now consider the derivatives with respect to the parameter $\tau_{a,b}$

$$\begin{aligned}\frac{\partial \log P_M(s|\boldsymbol{\theta})}{\partial \tau_{a,b}} &= \frac{1}{P_M(s|\boldsymbol{\theta})} \frac{\partial}{\partial \tau_{a,b}} \sum_{m \in A^n} \prod_{i=1}^n P(s_i|m_i) P_M(m_i|m_{i-1}) \\ &= \frac{1}{P_M(s|\boldsymbol{\theta})} \sum_{m \in A^n} \frac{\partial}{\partial \tau_{a,b}} \prod_{i=1}^n \frac{\theta_{s_i,m_i}}{\sum_{\sigma \in \Sigma} \theta_{\sigma,m_i}} \tau_{m_i|m_{i-1}}.\end{aligned}$$

Letting \mathbf{a} and \mathbf{b} be the sequence of states m and \mathbf{b} be the same sequence of states shifted one position, $\psi(a,b) = \tau_{a,b}$ and $c_i = \theta_{s_i|m_i}$, one has

$$Q(\mathbf{a}, \mathbf{b}) = \prod_{i=1}^n \theta_{s_i|m_i} \frac{\tau_{m_i,m_{i-1}}}{\sum_{a' \in A} \tau_{a',m_{i-1}}} \tau_{m_i|m_{i-1}} = P_M(s, m|\boldsymbol{\theta}).$$

It follows from the derivative of Q that

$$\begin{aligned} \frac{\partial \log P_M(s|\boldsymbol{\theta})}{\partial \tau_{a,b}} &= \sum_{k=1}^n \sum_{m \in A^n} \left(\frac{[m_{k-1} = b][m_k = a]}{\tau_{a|b}} - [m_k = a] \right) P_M(m|s, \boldsymbol{\theta}) \\ &= \frac{1}{\tau_{a|b}} \sum_{k=1}^n \mathbb{E}[[m_{k-1} = b][m_k = a]|s, \boldsymbol{\theta}] - \sum_{k=1}^n \mathbb{E}[[m_k = a]|s, \boldsymbol{\theta}], \end{aligned}$$

It remains to compute the expectations in each of the sums. These are the expectations that the particular emissions and transitions occurred in the generation of the string s .

The computation of these quantities will rely on an algorithm known as the forwards-backwards algorithm. As the name suggests this is a two-stage algorithm that computes the quantities

$$f_a(i) = P(s_1 \dots s_i, m_i = a),$$

in other words the probability that the i th hidden state is a with the prefix of the string s together with the probability $P(s)$ of the sequence. Following this the backwards algorithm computes

$$b_a(i) = P(s_{i+1} \dots s_n | m_i = a).$$

Once these values have been computed it is possible to evaluate the expectation

$$\begin{aligned} \mathbb{E}[[s_k = \sigma][m_k = a]|s] &= P(s_k = \sigma, m_k = a|s) \\ &= [s_k = \sigma] \frac{P(s_{k+1} \dots s_n | m_k = a) P(s_1 \dots s_k, m_k = a)}{P(s)} \\ &= [s_k = \sigma] \frac{f_a(k) b_a(k)}{P(s)}. \end{aligned}$$

Similarly

$$\begin{aligned} \mathbb{E}[[m_k = a]|s] &= P(m_k = a|s) \\ &= \frac{P(s_{k+1} \dots s_n | m_k = a) P(s_1 \dots s_k, m_k = a)}{P(s)} \\ &= \frac{f_a(k) b_a(k)}{P(s)}. \end{aligned}$$

Finally, for the second pair of expectations the only tricky evaluation is $\mathbb{E}[[m_{k-1} = b][m_k = a]|s]$, which equals

$$\frac{P(s_{k+1} \dots s_n | m_k = a) P(s_1 \dots s_{k-1}, m_{k-1} = b) P(a|b) P(s_k | m_k = a)}{P(s)} = \frac{f_b(k-1) b_a(k) \tau_{a|b} \theta_{s_k|a}}{P(s)}.$$

Hence, the Fisher scores can be evaluated based on the results of the forwards-backwards algorithm. The forwards-backwards algorithm again uses a dynamic programming approach based on the recursion

$$f_b(i+1) = \theta_{s_{i+1}|b} \sum_{a \in A} f_a(i) \tau_{b|a},$$

with $f_{a_0}(0) = 1$ and $f_a(0) = 0$, for $a = a_0$. Once the forward recursion is complete one has

$$P(s) = \sum_{a \in A} f_a(n).$$

The initialisation for the backward algorithm is

$$b_a(n) = 1$$

with the recursion

$$b_b(i) = \sum_{a \in A} \tau_{a|b} \theta_{\sigma_{i+1}|a} b_a(i+1).$$

Putting all of these observations together the following code is obtained, the calculation of the Fisher scores for the transmission probabilities is my own contribution.

```
//line numbers refer to Code Fragment 12.4 (page 435)
//in Shawe-Taylor and Cristianini (2004)
//use symbols 1, 2, 3, etc.

#include <iostream>
#include <fstream>
#include <sstream>
#include <math.h>
#include <string>

using namespace std;

int main()
{
    int string_length = 10;
    int number_of_states = 5;
    int number_of_symbols = 5;
    int p = number_of_states;
    int n = string_length;
    int a,b;
    double Prob = 0;
    string stringstring;

    ifstream hmmstream("globalhmm.txt"); //INPUT: HMM, contains one line of params
    ifstream stringfile("strings.txt"); //INPUT: symbol strings, one per line
    ofstream fisherfile("fisher.txt"); //OUTPUT: Fisher scores, one data item/line

    int s[n+1]; //symbol string, uses s[1] to s[n] (s[0] is never used)

    double PM[p+1][p+1]; //state transition probability matrix
    double P[number_of_symbols+1][p+1]; //conditional probs of symbols given states

    double scoree[p+1][number_of_symbols+1]; //Fisher scores for the emission probs
```

```

double scoret[p+1][p+1]; //Fisher scores for the transmission probabilities

double forw[p+1][n+1];
double back[p+1][n+1];

//initialize to zero
for (int i=0; i<=p; i++)
    for (int j=0; j<=p; j++)
        PM[i][j] = 0;
for (int i=0; i<=number_of_symbols; i++)
    for (int j=0; j<=p; j++)
        P[i][j] = 0;

PM[1][0] = 1.0; //because it is a left-to-right hidden Markov model
for (int i=2; i<=p; i++)
    PM[i][0] = 0;
for (int i=1; i<=p; i++)
    for (int j=1; j<=p; j++)
        hmmstream >> PM[i][j];
for (int i=1; i<=number_of_symbols; i++)
    for (int j=1; j<=p; j++)
        hmmstream >> P[i][j];

while (getline(stringfile, stringstring)) {
    istringstream stringstream (stringstring);

    //initialize to zero
    for (int i=0; i<=p; i++)
        for (int j=0; j<=n; j++)
            forw[i][j] = 0;
    for (int i=0; i<=p; i++)
        for (int j=0; j<=n; j++)
            back[i][j] = 0;
    for (int i=0; i<=p; i++)
        for (int j=0; j<=number_of_symbols; j++)
            scoree[i][j] = 0;
    for (int i=0; i<=p; i++)
        for (int j=0; j<=p; j++)
            scoret[i][j] = 0;
    for (int i=0; i<=n; i++)
        s[i] = 0;

    for (int i=1; i<=n; i++)
        stringstream >> s[i];

    for (int i=0; i<=p; i++)
        for (int j=1; j<=number_of_symbols; j++)
            scoree[i][j] = 0; //line 2

    for (int i=0; i<=p; i++)

```

```

    for (int j=1; j<=p; j++)
        scoret[i][j] = 0; //mvs

for (int i=0; i<=p; i++)
    forw[i][0] = 0; //line 3

for (int i=0; i<=p; i++) //line 4
    back[i][n] = 1;

forw[0][0] = 1; //line 4 (corrected)
Prob = 0;

for (int i=1; i<=n; i++) { //line 5
    for (a=1; a<=p; a++) { //line 7
        forw[a][i] = 0; //line 8
        for (b=0; b<=p; b++) //line 9 (corrected)
            forw[a][i] = forw[a][i] + PM[a][b]*forw[b][i-1]; //line 10
        forw[a][i] = forw[a][i]*P[s[i]][a]; //line 12
    }
}

for (a=1; a<=p; a++) //line 15
    Prob = Prob + forw[a][n];
for (int i=n-1; i>=1; i--) { //line 18
    for (a=1; a<=p; a++) { //line 19
        back[a][i] = 0; //line 20
        for (b=1; b<=p; b++)
            back[a][i] = back[a][i] +
                PM[b][a]*P[s[i+1]][b]*back[b][i+1]; //line 22
    }
}

//Fisher scores for the emission probabilities
for (int i=n-1; i>=1; i--) { //line 18
    for (a=1; a<=p; a++) { //line 19
        scoree[a][s[i]] = scoree[a][s[i]] +
            back[a][i]*forw[a][i] / (P[s[i]][a]*Prob); //line 24
        for (int sigma = 1; sigma<=number_of_symbols; sigma++)
            scoree[a][sigma] = scoree[a][sigma] -
                back[a][i]*forw[a][i]/Prob; //line 26 (corrected)
    }
}

//Fisher scores for the transmission probabilities
for (int i=n-1; i>=1; i--)
    for (b=1; b<=p; b++)
        for (a=1; a<=p; a++) {
            scoret[b][a] = scoret[b][a] +
                (back[a][i]*forw[b][i-1]*P[s[i]][a]/Prob - back[b][i]*forw[b][i]/Prob);
        }
}

```

```

//transform Fisher scores to the interval (0,1) using the logistic function
//scoree[i][j] = 1/(1 + exp(-scoree[i][j]));

for (int i=1; i<=p; i++)
  for (int j=1; j<=p; j++)
    fisherfile << scoree[i][j] << " ";

for (int j=1; j<=number_of_symbols; j++)
  for (int i=1; i<=p; i++)
    fisherfile << scoree[i][j] << " ";

fisherfile << endl;
}
hmmstream.close();
fisherfile.close();
system("PAUSE");
}

```

8 Test

This section concerns the prediction of synthetic data, generated by a very simple 5-symbol, 5-state HMM, in order to test the Fisher kernel. The hidden Markov model used in this thesis is based on a C++ implementation of a basic left-to-right HMM which uses the Baum-Welch (maximum likelihood) training algorithm written by Richard Myers. The hidden Markov model used to generate the synthetic data is shown below. Following the header are a series of ordered blocks, each of which is two lines long. Each of the 5 blocks corresponds to a state in the model. Within each block, the first line gives the probability of the model recurring (the first number) followed by the probability of generating each of the possible output symbols when it recurs (the following five numbers). The second line gives the probability of the model transitioning to the next state (the first number) followed by the probability of generating each of the possible output symbols when it transitions (the following five numbers).

```

states: 5
symbols: 5
0.5 0.96 0.01 0.01 0.01 0.01
0.5 0.96 0.01 0.01 0.01 0.01

0.5 0.01 0.96 0.01 0.01 0.01
0.5 0.01 0.96 0.01 0.01 0.01

0.5 0.01 0.01 0.96 0.01 0.01
0.5 0.01 0.01 0.96 0.01 0.01

0.5 0.01 0.01 0.01 0.96 0.01
0.5 0.01 0.01 0.01 0.96 0.01

```

```
1.0 0.01 0.01 0.01 0.01 0.96
0.0 0.0 0.0 0.0 0.0 0.0
```

1. Create a HMM with 5 states and 5 symbols, as above. Save as `hmm.txt`.
2. Use the HMM to generate 10,000 sequences, each 11 symbols long, $\{0, 1, 2, 3\}$, `'generate_seq hmm.txt 10000 11'`. Output will be `hmm.txt.seq`.
3. Save the output, `hmm.txt.seq`, in `Fisher.xls`, Sheet 1. Split the data into 5000 sequences for training, 2500 sequences for validation and 2500 sequences for testing. Separate the 11th column, this will be the target and is not used until later.
4. Copy the training data (without the 11th column) into `strings.txt`.
5. Run `'train_hmm strings.txt 1234 5 5 .01'`.
6. Copy the output, `hmmnew.txt`, into `Fisher.xls`, Sheet 2.
7. Take the average of each column, and save this as a one-row file, `globalhmm.txt`.
8. From `Fisher.xls`, Sheet 1, copy all of the data except the target column into `strings.txt` (delete previous contents).
9. Replace symbols thus: $4 \rightarrow 5, 3 \rightarrow 4, 2 \rightarrow 3, 1 \rightarrow 2, 0 \rightarrow 1$ (this is simply an artefact of the software). Save.
10. Run `Fisher.exe`, inputs are `globalhmm.txt` and `strings.txt`, output will be `fisher.txt`.
11. Use `oldformat.exe` to convert `fisher.txt` to LIBSVM format: `'oldformat fisher.txt fisher2.txt'`.
12. Copy and paste `fisher2.txt` into `Fisher.xls`, Sheet 3 (cells need to be formatted for text).
13. Copy target data from Sheet 1 `Fisher.xls` into a temporary file and replace symbols thus: $4 \rightarrow 5, 3 \rightarrow 4, 2 \rightarrow 3, 1 \rightarrow 2, 0 \rightarrow 1$.
14. Insert the target data into column A in `Fisher.xls`, Sheet 3, then split the data into training set, validation set and test set.
15. Copy and paste into `training.txt`, `validation.txt` and `test.txt`.
16. Scale the data.
17. Use LIBSVM for regression with default Gaussian (rbf) kernel and use the validation set to determine C and ϵ . `'svmtrain.exe -s 3 -t 2 [...]'`

Support vector regression with a radial basis function kernel ($e^{-\gamma\|\vec{u}-\vec{v}\|^2}$) was used. A validation set was used to select $C \in \{0.1, 1, 10, 100, 1000, 10000, 100000\}$ and $\epsilon \in \{0.00001, 0.0001, 0.001, 0.01, 0.1\}$. Three parameter combinations were joint best, namely $\{C\ 100, \epsilon\ 0.00001\}$, $\{C\ 100, \epsilon\ 0.0001\}$ and $\{C\ 100, \epsilon\ 0.001\}$, so the middle one was chosen $C = 100$ and $\epsilon = 0.0001$.

	Training set	Validation set	Test set
Correct classification (%)	84.08	83.52	82.80

Table 1: Fisher kernel test results

Results are given in Table 1 (page 15). There are five symbols, so if the algorithm was no better than random, one would expect a correct classification rate of 20.00%. The results are impressive, and evidence the fact that my implementation of the Fisher kernel works.

References

- BAUM, Leonard E., 1972. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *In: Oved SHISHA, ed. Inequalities III: Proceedings of the Third Symposium on Inequalities*. New York: Academic Press, pp. 1–8.
- BAUM, L. E., and J. A. EAGON, 1967. An Inequality with Applications to Statistical Estimation for Probabilistic Functions of a Markov Process and to a Model for Ecology. *Bulletin of the American Mathematical Society*, **73**(3), 360–363.
- BAUM, Leonard E., and Ted PETRIE, 1966. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, **37**(6), 1554–1563.
- BAUM, Leonard E., *et al.*, 1970. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, **41**(1), 164–171.
- BERNSTEIN, Serge, 1927. Sur l’Extension du Théorème Limite du Calcul des Probabilités aux Sommes de Quantités Dépendantes. *Mathematische Annalen*, **97**(1), 1–59.
- BHAR, Ramaprasad, and Shigeyuki HAMORI, 2004. *Hidden Markov Models: Applications to Financial Economics*. Volume 40 of *Advanced Studies in Theoretical and Applied Econometrics*. Dordrecht: Kluwer Academic Publishers.
- CAPPÉ, Olivier, Eric MOULINES, and Tobias RYDÉN, 2005. *Inference in Hidden Markov Models*. Springer Series in Statistics. New York: Springer Science+Business Media.
- DICK, Uwe, and Kristian KERSTING, 2006. Fisher Kernels for Relational Data. *In: Johannes FÜRNKRANZ, Tobias SCHEFFER, and Myra SPILIOPOULOU, eds. Machine Learning: ECML 2006: 17th European Conference on Machine Learning, Berlin, Germany, September 2006, Proceedings*, Volume 4212 of *Lecture Notes in Computer Science*. Berlin: Springer-Verlag, pp. 114–125.
- DURBIN, R., *et al.*, 1999. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge: Cambridge University Press.
- EDDY, Sean R., 2004. What is a Hidden Markov Model? *Nature Biotechnology*, **22**(10), 1315–1316.
- ELLIOT, Robert J., Lakhdar AGGOUN, and John B. MOORE, 2004. *Hidden Markov Models: Estimation and Control*. Volume 29 of *Applications of Mathematics*. New York: Springer-Verlag.

- HOLUB, Alex D., Max WELLING, and Pietro PERONA, 2005. Combining Generative Models and Fisher Kernels for Object Recognition. *In: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05), Volume 1*. Washington, DC: IEEE, pp. 136–143.
- JAAKKOLA, Tommi, Mark DIEKHANS, and David HAUSSLER, 1999. Using the Fisher Kernel Method to Detect Remote Protein Homologies. *In: Thomas LENGAUER, et al., eds. Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*. Menlo Park, CA: AAAI Press, pp. 149–158.
- JAAKKOLA, Tommi, Mark DIEKHANS, and David HAUSSLER, 2000. A Discriminative Framework for Detecting Remote Protein Homologies. *Journal of Computational Biology*, **7**(1–2), 95–114.
- JAAKKOLA, Tommi S., and David HAUSSLER, 1999a. Exploiting Generative Models in Discriminative Classifiers. *In: Michael S. KEARNS, Sara A. SOLLA, and David A. COHN, eds. Advances in Neural Information Processing Systems 11*, Bradford Books. Cambridge, MA: The MIT Press, pp. 487–493.
- JAAKKOLA, Tommi S., and David HAUSSLER, 1999b. Probabilistic Kernel Regression Models. *In: David HECKERMAN and Joe WHITTAKER, eds. Proceedings of the 1999 Conference on AI and Statistics*. San Mateo, CA: Morgan Kaufmann.
- JUANG, B. H., 1985. Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chains. *AT&T Technical Journal*, **64**(6), 1235–1249.
- KEMENY, John G., J. Laurie SNELL, and Anthony W. KNAPP, 1976. *Denumerable Markov Chains*. Second ed. Volume 40 of *Graduate Texts in Mathematics*. New York: Springer-Verlag.
- KERSTING, Kristian, and Thomas GÄRTNER, 2004. Fisher Kernels for Logical Sequences. *In: Jean-François BOULICAUT, et al., eds. Machine Learning: ECML 2004: 15th European Conference on Machine Learning, Pisa, Italy, September 2004. Proceedings*, Volume 3201 of *Lecture Notes in Computer Science*. Berlin: Springer, pp. 205–216.
- MACDONALD, Iain L., and Walter ZUCCHINI, 1997. *Hidden Markov and Other Models for Discrete-Valued Time Series*. Volume 70 of *Monographs on Statistics and Applied Probability*. Boca Raton: Chapman & Hall/CRC.
- MAMON, Rogemar S., and Robert J. ELLIOTT, eds., 2007. *Hidden Markov Models in Finance*. Volume 104 of *International Series in Operations Research & Management Science*. New York: Springer.

- MARKOV, A. A., 1906. Rasprostranenie zakona bol'shikh chisel na velichiny, zavisyaschie drug ot druga. *Izvestiya Fiziko-Matematicheskogo Obschestva pri Kazanskom Universitete, 2-ya seriya*, **15**, 135–156. In Russian. English translation: ‘Extension of the law of large numbers to dependent quantities’.
- MORENO, Pedro J., and Ryan RIFKIN, 2000. Using the Fisher Kernel Method for Web Audio Classification. *In: 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing: Proceedings, Volume IV*. IEEE, pp. 2417–2420.
- NICOTRA, Luca, Alessio MICHELI, and Antonina STARITA, 2004. Fisher Kernel for Tree Structured Data. *In: Jose C. PRINCIPE, et al., eds. Proceedings. 2004 IEEE International Joint Conference on Neural Networks. Volume 3*. IEEE, pp. 1917–1922.
- PORITZ, Alan B., 1988. Hidden Markov Models: A Guided Tour. *In: 1988 International Conference on Acoustics, Speech, and Signal Processing, 1988. ICASSP-88. Volume 1*. New York: IEEE Press, pp. 7–13.
- RABINER, Lawrence R., 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, **77**(2), 257–286.
- RABINER, L. R., and B. H. JUANG, 1986. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine*, **3**(1, Part 1), 4–16.
- SAUNDERS, Craig, John SHAWE-TAYLOR, and Alexei VINOKOUROV, 2003. String Kernels, Fisher Kernels and Finite State Automata. *In: Suzanna BECKER, Sebastian THRUN, and Klaus OBERMAYER, eds. Advances in Neural Information Processing Systems 15*, Bradford Books. Cambridge, MA: The MIT Press, pp. 633–640.
- SEWELL, Martin, 2006. Stochastic Processes. Lecture given to Department of Economics, Royal Holloway, University of London, Egham.
- SHAWE-TAYLOR, John, and Nello CRISTIANINI, 2004. *Kernel Methods for Pattern Analysis*. Cambridge: Cambridge University Press.
- TSUDA, Koji, *et al.*, 2004. Asymptotic Properties of the Fisher Kernel. *Neural Computation*, **16**(1), 115–137.
- VINOKOUROV, Alexei, and Mark GIROLAMI, 2001. Document Classification Employing the Fisher Kernel Derived from Probabilistic Hierarchic Corpus Representations. *In: Proceedings of ECIR-01, 23rd European Colloquium on Information Retrieval Research*. British Computer Society Information Retrieval Specialist Group. Berlin: Springer-Verlag, pp. 24–40.