COMP4702/COMP7703/DATA7703 - Machine Learning
Homework 8 - Kernel Machines
Solutions

Marcus Gallagher

Core Questions

1. (From Marsland's text) Suppose that the following are a set of points in two classes:

$$\text{class 1}: \begin{pmatrix}1\\1\end{pmatrix} \begin{pmatrix}1\\2\end{pmatrix} \begin{pmatrix}2\\1\end{pmatrix}$$

$$\text{class 2}: \begin{pmatrix}0\\0\end{pmatrix} \begin{pmatrix}1\\0\end{pmatrix} \begin{pmatrix}0\\1\end{pmatrix}$$

Plot them and find the optimal separating line. What are the support vectors, and what is the margin[1]?

Answer:

Optimal separating line: $x_2 = -x_1 + 1.5$

Support vectors are:

$$\begin{pmatrix}1\\1\end{pmatrix} \begin{pmatrix}1\\0\end{pmatrix} \begin{pmatrix}0\\1\end{pmatrix}$$

The margin (one-side) is $\frac{\sqrt{2}}{4} \approx 0.3536$.

Students may use different methods to get these answers. The support vectors are clear "by inspection" and the line can be found using simple algebra. The margin can be found using trigonometry or the formula from Alpyadin ($\frac{g(\mathbf{x})}{||\mathbf{w}||}$).

2. Using the data and your answer from the previous question, calculate the (Euclidean) distance between the optimal separating line (i.e. the discriminant) and the point $(7.14 \;\; 5.23)^T$.

Answer: Using the formula above (with $g(\mathbf{x}) = x_1 + x_2 - 1.5$), the answer is ≈ 7.6863

3. Lodhi et al. [1] describe the string subsequence kernel and illustrate its calculation with an example (p.422-423). If this example was expanded to include the word "rat", how many dimensions would the feature space have?

Solutions: The example already has 8 dimensions. We would need to add `r-a, r-t`, so we would have 10 dimensions.

[1] That is, the distance from the discriminant to the nearest data point on one side. Some books may define the margin as twice this distance (i.e. both sides).

Extension Questions

4. From the lecture material, we know that there is an equivalance between the kernel function and the basis functions in support vector machines:

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}).\phi(\mathbf{x}')$$

for two data points $\mathbf{x}$ and $\mathbf{x}'$. One interesting example of a kernel function is the Arc-cosine kernel:

$$k_{ac}(\mathbf{x}, \mathbf{x}') = \frac{||\mathbf{x}||||\mathbf{x}'||}{2\pi}(\sin\theta + (\pi - \theta)\cos\theta)$$

which is known to have an equivalence

$$k_{approx}(\mathbf{x}, \mathbf{x}') = \frac{1}{n}\sum_{i=1}^{n} \max(0, \mathbf{w_i}.\mathbf{x}) \max(0, \mathbf{w_i}.\mathbf{x}') \approx k_{ac},$$

with the approximation becoming exact as $n \to \infty$. Here, $\mathbf{w_i}$ is a weight vector of dimensionality equal to the dimensionality of $\mathbf{x}$, with each entry in $\mathbf{w_i}$ drawn independently from $\mathcal{N}(0, 1)$.

Examine this approximation by writing a program to compute and plot the value of k_{ac} and k_{approx} as a function of θ.

Hints:

- Use the dataset supplied (hw8.csv) as your input data. Use (e.g) the first 500 columns of the dataset as examples of $\mathbf{x}$, the other half as examples of $\mathbf{x}'$. The angles between these vectors (θ) have been calculated for you in the final column. Note also that in this dataset, $||\mathbf{x}|| = ||\mathbf{x}'|| = 1$ for all data points.
- Plot k_{ac} as a curve, and overlay k_{approx} as points that should roughly follow the curve.
- State the value of n you have used to create your plot.

See Figures 1 and 2. This plot is for $n = 1000$.

5. Consider k_{approx} from the previous question. What machine learning model (recently covered in the course!) does this equation describe? Be specific about the model (e.g. its structure and any functions used within the model).

This looks very much like a neural network with a large number (n) of hidden units in a single hidden layer, with Relu activation functions.

References

[1] Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C. (2002). Text classification using string kernels. Journal of Machine Learning Research, 2(Feb), 419-444.

```
close all; clc; clear all;
data = csvread('input_data.csv');

d = 500;
num_data_points = 500;
num_random_features = 1000;

X = data(:,1:d);
Y = data(:,d+1:2*d);
theta_list = data(:,2*d+1:2*d+1);
ker_list = zeros(length(theta_list), 1);

% Plot the analytical result
plot(theta_list, ...
    1/(2*pi)*(sin(theta_list) + (pi-theta_list).*cos(theta_list)), ...
    'linewidth', 8); hold on

% Plot the experimental results
W = randn(num_random_features, d);
for i = 1:num_data_points
    x = X(i,:)';
    y = Y(i,:)';

    % The obvious way
    ker = 0;
    for j = 1:num_random_features
        w = W(j,:);
        ker = ker + max(0, dot(w, x)) * max(0, dot(w,y));
    end
    ker = ker/num_random_features;

    % The somewhat faster, somewhat less obvious way
    %ker = 1/num_random_features*dot(max(0, W*x), max(0, W*y));

    ker_list(i,1) = ker;
end

plot(theta_list, ker_list, 'k*', 'markersize', 4);

xlabel('$$\theta$$', 'Interpreter','latex')
ylabel('$$\frac{1}{n} \psi(x) \cdot \psi(x'')$$', 'Interpreter','latex')
legend('Arc-cosine Kernel', 'Experimental data')
```

Figure 1: Matlab code.

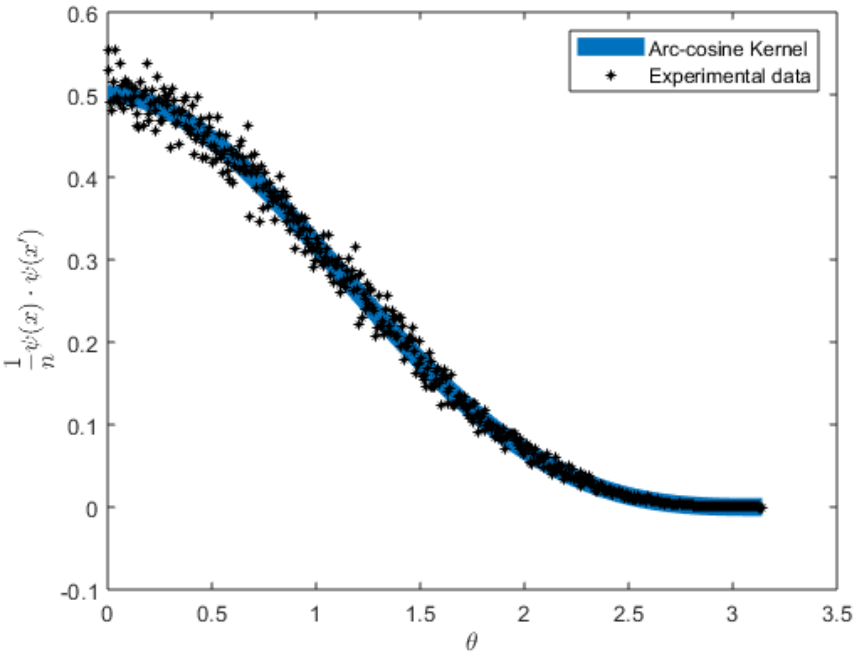

Figure 2: Comparison of k_{ac} as a curve, and k_{approx} as points.