

58147 Machine Learning (Spring 2005)

Exercise 4 (Wednesday 23 February)

1. Consider the degree q polynomial kernel k_q , given in Example 2.32. Write out explicitly the features corresponding to this kernel in the case $n = 2$, $q = 2$, $c = 1$.

Explanation: We already know that the features consist of monomials of two variables, up to degree 2. Thus, you only need to figure out the constants c_i in

$$\boldsymbol{\psi}((x_1, x_2)) = (c_1, c_2x_1, c_3x_2, c_4x_1^2, c_5x_1x_2, c_6x_2^2)$$

so that $\boldsymbol{\psi}(\mathbf{x}) \cdot \boldsymbol{\psi}(\mathbf{z}) = (\mathbf{x} \cdot \mathbf{z} + 1)^2$. The purpose of the exercise is to show more concretely the actual computation hidden behind the kernel trick.

2. As with the all subsets kernel (Example 2.33), define for $A \subseteq \{1, \dots, n\}$ the feature

$$\psi_A(\mathbf{x}) = \prod_{i \in A} x_i.$$

The degree q ANOVA feature map has the $\binom{n}{q}$ features ψ_A where $|A| = q$. (Thus the all subsets feature map combines the ANOVA features for $q = 0, \dots, n$.)

Let k_q be the kernel of this feature map. There is no nice closed form for this kernel, but given $\mathbf{x}, \mathbf{z} \in \mathbf{R}^n$ we can still compute the value

$$k_q(\mathbf{x}, \mathbf{z}) = \sum_{|A|=q} \psi_A(\mathbf{x})\psi_A(\mathbf{z})$$

much more efficiently than the naive $O(n^q)$. Give an algorithm to do this.

Hint: Express $k_q((x_1, \dots, x_n), (z_1, \dots, z_n))$ in terms of $k_{q-1}((x_1, \dots, x_{n-1}), (z_1, \dots, z_{n-1}))$ and $k_q((x_1, \dots, x_{n-1}), (z_1, \dots, z_{n-1}))$. You can save computation effort by dynamic programming.

3. *Online linear regression* is like online linear classification, except that \hat{y}_t and y_t can both be arbitrary real numbers. Consider the classical Least Mean Squares algorithm (LMS, also known as Widrow-Hoff):

- Initialise $\mathbf{w}_1 = \mathbf{0}$.
- Repeat for $t = 1, \dots, T$:
 1. Get $\mathbf{x}_t \in \mathbf{R}^n$.
 2. Predict $\hat{y}_t = \mathbf{w}_t \cdot \mathbf{x}_t$.
 3. Receive the correct answer y_t .
 4. Update $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\hat{y}_t - y_t)\mathbf{x}_t$.

Here $\eta > 0$ is a learning rate parameter.

Assume that there are some $\mathbf{u} \in \mathbf{R}^n$ and $X > 0$ such that $y_t = \mathbf{u} \cdot \mathbf{x}_t$ and $\|\mathbf{x}_t\|_2 \leq X$ for all t . Show that the square loss of the LMS algorithm can be bounded as

$$\sum_{t=1}^T (y_t - \hat{y}_t)^2 \leq \|\mathbf{u}\|_2^2 X^2.$$

Hint: show that

$$\frac{1}{2}\|\mathbf{u} - \mathbf{w}_t\|_2^2 - \frac{1}{2}\|\mathbf{u} - \mathbf{w}_{t+1}\|_2^2 = \left(\eta - \frac{1}{2}\eta X^2\right) (y_t - \hat{y}_t)^2.$$

Optimise η and sum over t .

- 4.-5. We continue last week's experiment on large-dimensional data. Create again a 200×50 matrix of 200 instance vectors drawn uniformly at random from $[-1, 1]^{50}$, and label the instances using a linear classifier $(\mathbf{u}, b) = ((0.5, 0.5, 0, \dots, 0), 0)$. Split the data into a training set and a test set, each having 100 examples.

Implement the Winnow and Marginalised Perceptron algorithms. Compare the performance of these two algorithms to the basic Perceptron. How many iterations are needed until the training set is correctly classified? How accurate is the final hypothesis on the test set?

Do not spend too much time in trying to find the optimal parameter settings. Since in this experiment you know the "correct" \mathbf{u} , in advance, you can compute the actual margin of \mathbf{u} on your data. Plug it into the theorems given in lectures to get some feasible value for η (and ρ for the Marginalised Perceptron). Then try to increase or decrease the parameters and see whether the performance improves noticeably. Compare the mistake bounds from the theorems to the actual performance of the algorithms.

Define $f(t) = (|w_{t,1}| + |w_{t,2}|) / \|\mathbf{w}_t\|_1$ to be the proportion of weight given to the relevant variables (x_1 and x_2) in the t th hypothesis of the algorithm. Ideally $f(t)$ should converge to 1 quickly as t increases, meaning that the relative weight of the 48 irrelevant variables goes to zero. Plot $f(t)$ as a function of t for the three algorithms. Can you see any obvious differences between them?