9. Cubic Features

Extension Note: Project 2 due date has been extended by 2 days to **July 18 23:59UTC** (Note the UTC time zone).

In this section, we will work with a **cubic feature** mapping which maps an input vector $x = [x_1, \ldots, x_d]$ into a new feature vector $\phi(x)$, defined so that for any $x, x' \in \mathbb{R}^d$:

$$\phi(x)^T \phi(x') = (x^T x' + 1)^3$$

You will be working in the files `part1/main.py` and `part1/features.py` in this problem

Computing Cubic Features

3.0/3.0 points (graded)
In 2-D, let $x = [x_1, x_2]$. Write down the explicit cubic feature mapping $\phi(x)$ as a vector; i.e., $\phi(x) = [f_1(x_1, x_2), \cdots, f_N(x_1, x_2)]$

STANDARD NOTATION

Hint

Hint: $\phi(x)$ should be a 10-dimensional vector.

Hide

$\phi(x) =$ [x_1^3, x_2^3, sqrt(3)*x_1^2*x_2, sqrt(3)*x_1*x_2^2, sqrt(✔

Answer: [x_1^3, sqrt(3)*x_1^2*x_2, sqrt(3)*x_1^2, sqrt(3)*x_1*x_2^2, sqrt(6)*x_1*x_2, sqrt(3)*x_1, x_2^3, sqrt(3)*x_2^2, sqrt(3)*x_2, 1]

Submit You have used 2 of 20 attempts

Answers are displayed within the problem

The `cubic_features` function in `features.py` is already implemented for you. That function can handle input with an arbitrary dimension and compute the corresponding features for the cubic Kernel. Note that here we don't leverage the kernel properties that allow us to do a more efficient computation with the kernel function (without computing the features themselves). Instead, here we do compute the cubic features explicitly and apply the PCA on the output features.

Applying to MNIST

1.0/1.0 point (graded)
If we explicitly apply the cubic feature mapping to the original 784-dimensional raw pixel features, the resulting representation would be of massive dimensionality. Instead, we will apply the quadratic feature mapping to the 10-dimensional PCA representation of our training data which we will have to calculate just as we calculated the 18-dimensional representation in the previous problem. After applying the cubic feature mapping to the PCA representations for both the train and test datasets, retrain the softmax regression model using these new features and report the resulting test set error below.

Important: You will probably get a runtime warning for getting the log of 0, ignore. Your code should still run and perform correctly.

Note: Use the same training parameters as the first softmax model given in `main.py` file and temperature 1.

If you have done everything correctly, softmax regression should perform better (on the test set) using these features than either the 18-dimensional principal components or raw pixels. The error on the test set using cubic features should only be around 0.08, demonstrating the power of nonlinear classification models.

Error rate for 10-dimensional cubic PCA features = 0.0864 ✔ **Answer:** 0.0867

Submit You have used 1 of 20 attempts

Answers are displayed within the problem

Discussion

Show Discussion

Topic: Unit 2 Nonlinear Classification, Linear regression, Collaborative Filtering (2 weeks):Project 2: Digit recognition (Part 1) / 9. Cubic Features