

EdX and its Members use cookies and other tracking technologies for performance, analytics, and marketing purposes. By using this website, you accept this use. Learn more about these technologies in the [Privacy Policy](#).



MITx: 6.86x

Machine Learning with Python-From Linear Models to Deep Learning

[Help](#)[sandipan_dey](#) ▼

[Unit 2 Nonlinear Classification,](#)
[Linear regression, Collaborative](#)

[Course](#) > [Filtering \(2 weeks\)](#)

> [Project 2: Digit recognition \(Part 1\)](#) > 9. Cubic Features

9. Cubic Features

In this section, we will work with a **cubic feature** mapping which maps an input vector $x = [x_1, \dots, x_d]$ 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

3/3 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., $\phi(x) = [f_1(x_1, x_2), \dots, f_N(x_1, x_2)]$

[STANDARD NOTATION](#)

Hint

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

[Hide](#)

$\phi(x) = [x_1^3, x_2^3, \sqrt{3}x_1^2x_2, \sqrt{3}x_1x_2^2, \sqrt{3}]$ ✓

Submit

You have used 2 of 20 attempts

✓ Correct (3/3 points)

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

1/1 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 = ✓

Submit

You have used 1 of 20 attempts

✓ Correct (1/1 point)

Discussion

Hide Discussion

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

Add a Post

Show all posts ▼

by recent activity ▼

<p>? <u>Order of operations for applying cubic features to MNIST</u></p> <p><u>First, I assume the reference to "the quadratic feature mapping" should instead be "cubic." Second, I get an error of about .21, which is much higher than the expected 0.08, s...</u></p>	<p>9</p>
<p>? <u>Typo? Instead, we will apply the quadratic feature mapping to the 10-dimensional PCA representation of</u></p> <p><u>"Instead, we will apply the **quadratic** feature mapping to the 10-dimensional PCA representation of" Should this be cubic instead of quadratic?</u></p>	<p>2</p>

Learn About Verified Certificates