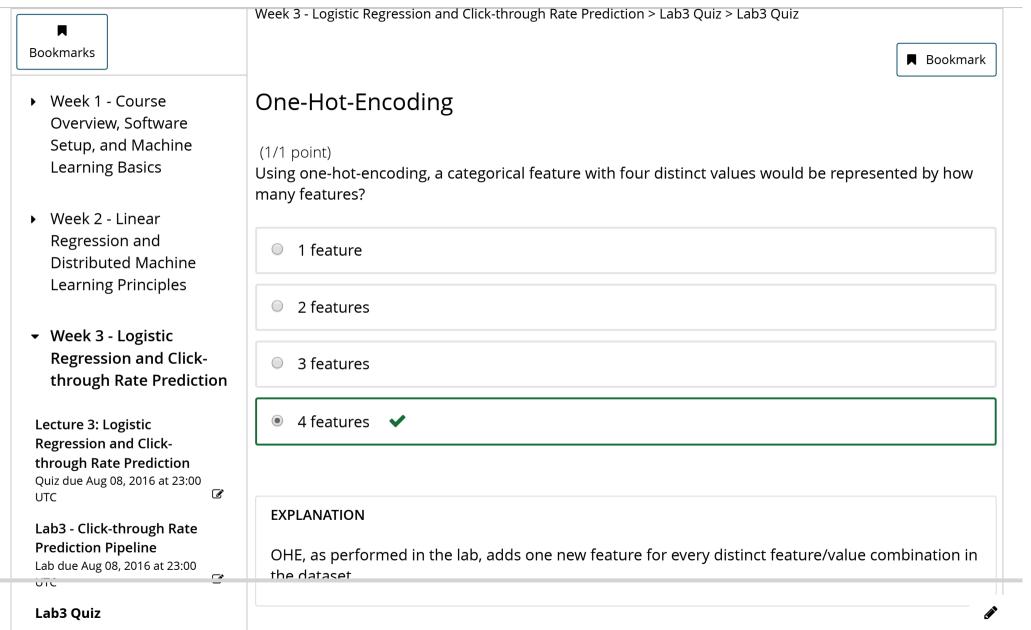


BerkeleyX: CS120x Distributed Machine Learning with Apache Spark



Quiz due Aug 08, 2016 at 23:00 UTC

 Week 4 - Principal Component Analysis and Neuroimaging

Rare Events

(1/1 point)

For rare events it is often a good idea to predict probabilities instead of classes.

🏿 True 🗸

False

EXPLANATION

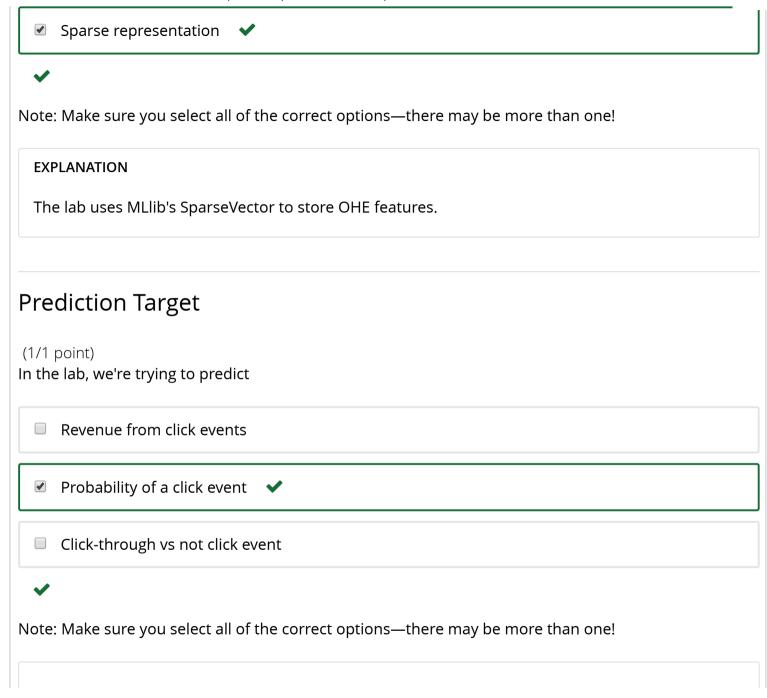
True. When dealing with rare events (like click events), we often want to use probabilities and log-loss rather than class predictions and 0 / 1 loss. For example, if the probability of a click event is relatively high, say 10%, we would still classify that observation as not a click event given the typical threshold of 50%, so using probabilities provides us with much more granular information.

Feature Representation

(1/1 point)

The OHE features in the lab are stored in a:

Dense representation



EXPLANATION

We're trying to predict the probability of a click event so that we can minimize the log-loss.

OHE Features

(1/1 point)

In the lab, using the OHE method on the training data in Part (3d) creates a dictionary with:

- 23,394 features
- 36,177 features
- 233,941 features
- 361,772 features

EXPLANATION

The OHE method generates over 200,000 different features, and we are only working with a small sample of the full dataset. Note that the model you created in Part (4a) also has over 200,000 weights, which are used to convert these features into predictions.

Feature Hashing

(1/1 point)

The feature hashing performed in the lab:

- Discards rare features
- Increases the number of features
- Requires calculating the OHE dictionary
- Causes feature collisions for certain observations



Note: Make sure you select all of the correct options—there may be more than one!

EXPLANATION

No features are discarded. Instead, several features map to the same buckets. The total number of features is decreased substantially. Feature hashing does not require computing an OHE dictionary. Several of the training observations (almost 5,000) have hash collisions, which can seen by running: hashTrainData.filter(lambda lp: np.any(lp.features.values > 1)).count().

Sparse Vectors

(3/3 points)

In Part (1b) we use a sparse vector representation to efficiently store a one-hot-encoded (OHE) feature vector. Imagine that we have 1000 OHE features, and that for a particular data point, we have *s* non-zero OHE features.

If s = 10, how much smaller is the storage footprint of the sparse vector representation versus the dense representation (assume that all indices and values are stored as floats)?

100x

50x 🗸

10x

they are the same size

If s = 500, how much smaller is the storage footprint of the sparse vector representation versus the dense representation (assume that all indices and values are stored as floats)?

100x

50x

Lab3 Quiz Lab3 Quiz CS120x Courseware edX				
○ 10x				
● they are the same size ✔				
Suppose we would like to compute a dot product between this feature vector and a dense vector, and assume $s = 10$. How many fewer scalar multiplications must we perform if we use a sparse vector representation versus a dense representation of the feature vector (assume we have random access the entries of the dense vector)?				
● 100x ✔				
● 50x				
○ 10x				
they are the same size				

EXPLANATION

The dense representation requires 1000 floats, while the sparse representation requires 2s floats, since for each non-zero we must store the index and the value. When computing a dot product, we only need to consider the non-zero entries of the feature vector, so we only need to perform s

scalar multiples when using a sparse representation, versus 1000 scalar multiplies when using a dense representation.

Hashing

(3/3 points)

In Part (5a) of the coding assignment we hashed the three sample points using numBuckets=4 and numBuckets=100. Complete the three statements below about these hashed features summarized in the following table using each answer once.

Name	Raw Features	4 Buckets	100 Buckets
sample_one	[(0, 'mouse'), (1, 'black')]	{3: 2.0}	{99: 1.0, 51: 1.0}
sample_two	[(0, 'cat'), (1, 'tabby'), (2, 'mouse')]	{0: 1.0, 1: 2.0}	{72: 1.0, 9: 1.0, 21: 1.0}
sample_thre		{0: 1.0, 2: 1.0, 3:	{80: 1.0, 82: 1.0, 51:
e	'salmon')	1.0}	1.0}

With 100 buckets, sample one and sample three both contain index 51 due to _____.

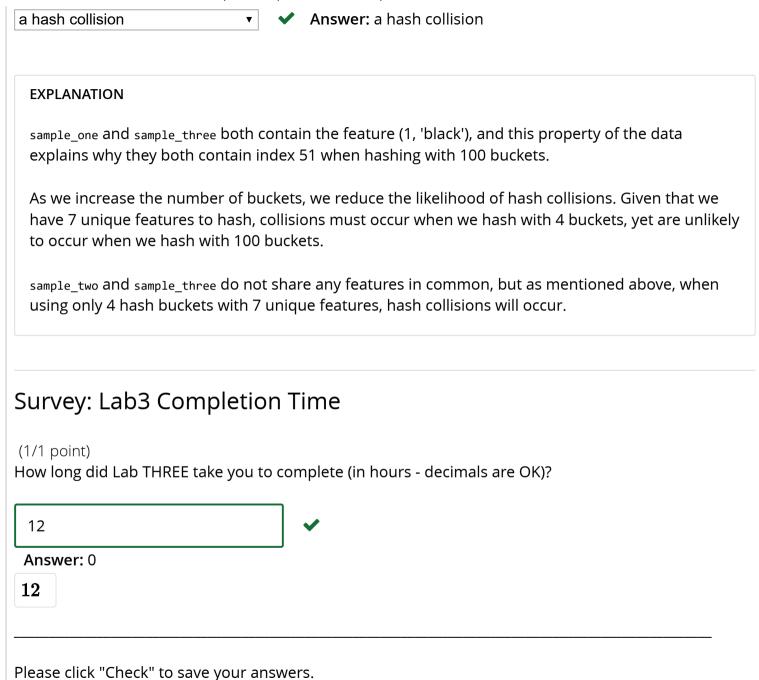
underlying properties of the data ▼ ✓ Answer: underlying properties of the data

It is likely that sample_two has two indices with 4 buckets, but three indices with 100 buckets due to

the fact we go from 4 to 100 buckets

Answer: the fact we go from 4 to 100 buckets

With 4 buckets, sample_two and sample_three both contain index 0 due to _____.



⊚ ③ ⑤ ⑤ Some Rights Reserved



© edX Inc. All rights reserved except where noted. EdX, Open edX and the edX and Open EdX logos are registered trademarks or trademarks of edX Inc.















