MLWave / **extremely-simple-one-shot-learning**

Extremely simple one-shot learning in Python

MLWave added note — Latest commit c606f67 on 17 Mar 2016

.gitattributes	Added .gitattributes & .gitignore files	a year ago
.gitignore	Added .gitattributes & .gitignore files	a year ago
esmf.py	first commit	a year ago
readme.md	added note	a year ago

readme.md

An embarrassingly simple approach to one-shot learning

This is an experiment in Python using approaches from the ICML '15 paper “An embarrassingly simple approach to zero-shot learning” by Bernardino Romera-Paredes, Philip H. S. Torr.

http://jmlr.org/proceedings/papers/v37/romera-paredes15.pdf

An embarrassingly simple approach to zero-shot learning

With matrix factorization you can decompose a `n*m` matrix into a `n*a` matrix and a `a*m` matrix, where `a` is the number of latent features.

An embarrassingly simple approach to zero-shot learning uses this to do zero-shot learning.

During the training stage an `n*m` weight/coefficient matrix is trained, where `n` is the number of features and `m` is the number of classes. Such that `np.argmax( np.dot( x.T, weight_matrix ) )` = predicted class.

They also train an `a*m` Signature matrix in an unsupervised manner. `a` is a number of (binary or soft) class attributes which can be found in the dataset, from external data, or in an unsupervised manner.

For instance, when the classes are: `bear` and `horse` and the attributes are `[brown, can_ride, domesticated]` the Signature matrix may look like:

```
bear horse
[ 1, 1 ] brown
[ 0, 1 ] can_ride
[ 0, 1 ] domesticated
```

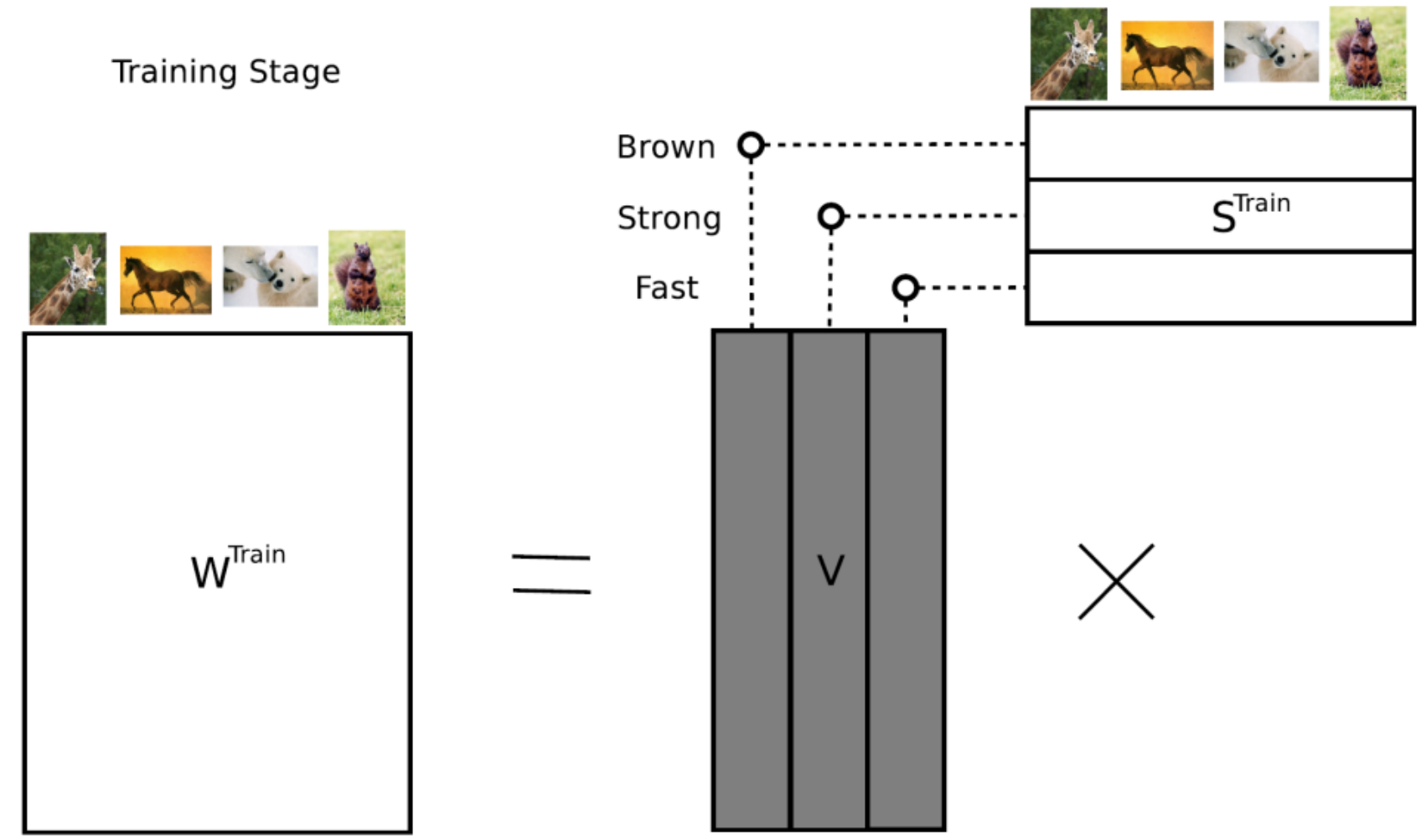
Using the Signature matrix and the Weight matrix we calculate an `n*a` matrix `V`, such that `np.dot(V, S) = ~W`.

Zero-shot

When we want to predict new classes we create a new signature matrix `S'`.

```
moose donkey tiger
[ 1, 1, 0 ] brown
[ 0, 1, 0 ] can_ride
[ 0, 1, 0 ] domesticated
```

We use `np.dot( V, S')` to obtain `~W'` . For new test samples we do `np.argmax( np.dot( x.T, ~W' )` to get our class prediction.



One-shot

We create the Weight matrix using logistic regression.

We create attributes with unsupervised learning.

We take the first 2 components of:

- PCA
- Local linear embedding

to create 4 class attributes. For every train sample belonging to a class we average the 4-dimensional PCA_LLE filter to get our Signature matrix. For instance, with 2 classes `digit1` and `digit3` :

```
digit1 digit3
[ 0.05, 0.06 ] PCA1
[ 0.11, 0.96 ] PCA2
[ 0.45, 0.11 ] LLE1
[ 0.95, 0.13 ] LLE2
```

When we want to predict new classes we take at least 1 sample and use the fitted PCA and LLE models to get a 4-dimensional vector. Taking the average of more samples per class improves performance.

```
digit7 digit9
[ 0.04, 0.19 ] PCA1
[ 0.12, 0.76 ] PCA2
[ 0.49, 0.14 ] LLE1
[ 0.85, 0.11 ] LLE2
```

Experiment

We use a 10 class digit dataset with 64 features. We will use digits 0,1,2,7,8,9 for the seen classes. For the unseen digits we use 3,4,5,6.

After fitting logistic regression (`0.911` accuracy) on the 6 seen classes our Weight coefficient matrix is `64*6` . Our Signature matrix is `4*6` . We calculate a `64*4` matrix V.

To create predictions for the unseen classes we take 1 sample per new class and transform them with PCA and LLE to get

our Signature matrix `S'`.

We use `V` and `S'` to calculate `~W'` with size `64*4`. Now for all test samples we calculate `np.argmax( np.dot( x.T, ~W' )` to get a class prediction.

We obtain a multi-class accuracy of `0.846` with 50 labeled samples per class, `0.759` with 10 labeled samples per class, and a variant accuracy of `0.609` with 1 labeled sample per class. By comparison: random guessing is `0.25` accuracy.

Note

We used a very simple toy dataset and non-rigorous method of evaluation. The goal was to replicate the basic idea with an extremely simple baseline, not to obtain (or claim) state-of-the-art performance.

One-shot learning has less constraints than zero-shot learning approach (we need at least one labeled sample, or another model communicating this as a vector). But we do get to use this approach when no class attributes are available.

We completely gloss over one of the main contributions in the paper: Regularization of the V matrix. We calculate V from W and S with least-squares. The paper includes a regularizer with more favourable properties.

See the original Matlab code here: https://dl.dropboxusercontent.com/u/5961057/ESZSL_v0.1.zip and a repository with the code for the real data experiments in the paper here: https://github.com/bernard24/Embarrassingly-simple-ZSL