Contextual Bandit Example

Edit New Page

arshak edited this page on 28 Aug 2014 · 3 revisions

Contextual Bandit functionalities in VW

VW contains a contextual bandit module which allows you to optimize a predictor based on already collected contextual bandit data. In other words, the module does not handle the exploration issue, it assumes it can only use the currently available data previously collected from some "exploration" policy.

The data is specified as a set of tuples (x,a,c,p) where x are the current features/context for the decision, a is the chosen action by the exploration policy for the context x, c is the observed cost for action a in context x, and p is the probability the exploration policy choose this action in context x.

Based on this data, the module tries to optimize a policy that chooses actions with minimum cost for the observed contexts. The module implements approaches described in the paper:

"Doubly Robust Policy Evaluation and Learning", by Miroslav Dudik, John Langford and Lihong Li. In ICML 2011.

Simple Example

Here is a simple example that illustrates the input format and how to use vw on this data.

We consider a problem with 4 actions and we observed the following 5 datapoints in VW format:

```
1:2:0.4 | a c
3:0.5:0.2 | b d
4:1.2:0.5 | a b c
2:1:0.3 | b c
3:1.5:0.7 | a d
```

Here each line is a separate example and each takes the form:

```
action:cost:probability | features
```

Where

- action is the id of the action taken where we observed the cost
- cost is the cost observed for this action
- probability is the probability the exploration policy choose this action when collecting the data
- features are the list of all features for this example specified as usual for classification/regression problem with vw

So the first line above indicates we observed action 1 has cost 2 on an example with features a and c, and this action was chosen with probability 0.4 by the exploration policy in this context when collecting the data.

If this data is in a file train.dat, then we can train a predictor with vw on this data as follows:

```
vw -d train.dat --cb 4
```

The flag `--cb` specifies that we want to use the contextual bandit module. When used, it must be followed by the number of actions. Hence here we specified `--cb 4` since there are 4 actions.

Efficient Second Order Online Learning

Error Correcting Tournament (ect) multi class example

Examples

Show 28 more pages…

Clone this wiki locally

https://github.com/JohnLa

Specifying the contextual bandit approach

In addition, one can specify the particular policy evaluation approach when optimizing the policy. 3 approaches can be used: inverse propensity score (`ips`), direct method (`dm`), and doubly robust (`dr`). These are all detailed in the paper above. Doubly robust is the approach used by default. To use a different one you can use the `--cb_type` flag:

```
vw -d train.dat --cb 4 --cb_type ips
```

This will use the ips method. `--cb_type dm` will use the direct method, and `--cb_type dr` uses the default doubly robust method.

Detailed input format

As shown above the simplest way to specify each training example is through a line

```
Action:cost:probability | features
```

When such examples are specified it is assume all actions could be taken in this context. However sometimes the actions that can be taken might be dependent on the context. In this case, one can specify examples by listing the available actions, e.g. as follows:

```
1 3:1:0.5 4 | a b e
```

In the scenario above where there are 4 actions, this line indicates an example with features a b e where only actions 1,3 and 4 can be taken, action 3 was chosen by the exploration policy, with probability 0.5, and observed cost 1. The action specified with a cost and a probability is the one that is considered observed for this training example.

Additionally one can specify the costs of all actions if they are known for proper evaluation of the learned predictor. When specifying a cost for only one action, it is hard to know exactly the cost incured by a policy that chooses different actions. This is estimated using the methods described in the paper above and this is the evaluated cost shown when running vw. However these are only estimates. If the cost of all actions on each examples are known, for example in a test set, one can evaluate exactly the cost of the learned policy for the actions it chooses. That input format allows to specify a cost for all actions in each example, e.g. as follows:

```
1:2 2:1:0.3 3:0.5 4:0.1 | a b
```

This line indicates an example with features a b, where action 1 has cost 2, action 2 has cost 1 and was chosen by the exploration policy, with probability 0.3, action 3 has cost 0.5, and action 4 has cost 0.1. Again the action where both a cost and a probability is specified is considered as the observed action. So it this case if we train on this example only the cost 1 for action 2 is assumed observed and used by the training algorithm. The other costs for the other actions are not used for training, but are used only for evaluating the cost of the predictions made by the predictor, instead of the cost estimates from the ips, dm or dr methods. This can also be used with examples where only a subset of actions can be chosen, e.g.:

```
1:2 3:4:0.2 | a d
```

This indicates an example where only action 1 or 3 can be chosen, action 1 has cost 2 and action 3 has cost 4 and was chosen by the exploration policy, with probability 0.2.

Example training and testing on separate datasets

Consider the same example as above with 4 actions with the above training data and we have another test set test.dat:

```
1:2 3:5 4:1:0.6 | a c d
1:0.5 2:1:0.4 3:2 4:1.5 | c d
```

Then we can train a predictor-model on training data and test it on this test data as follows:

```
vw -d train.dat --cb 4 -f cb.model
vw -t -d test.dat -i cb.model
```

The first command trains a model and saves it in file cb.model using the usual `-f` flag. The second command specifies we are only testing using the usual `-t` flag and that we are loading the model cb.model using the usual -i flag. Note that it is not necessary to respecify the `--cb` flag, since this is saved in the model file.