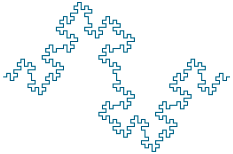
HEP, HEP, Hooray

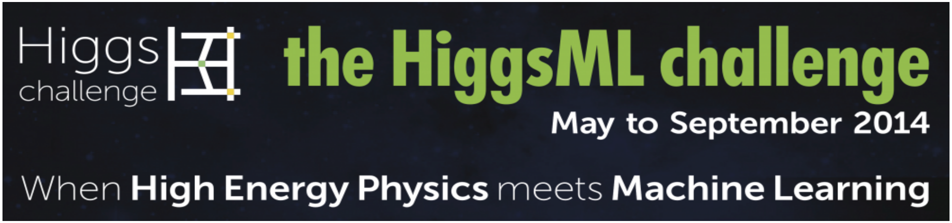
Gábor Melis

@GaborMelis | http://quotenil.com | mega@retes.hu

Franz Inc., Fixnum Services

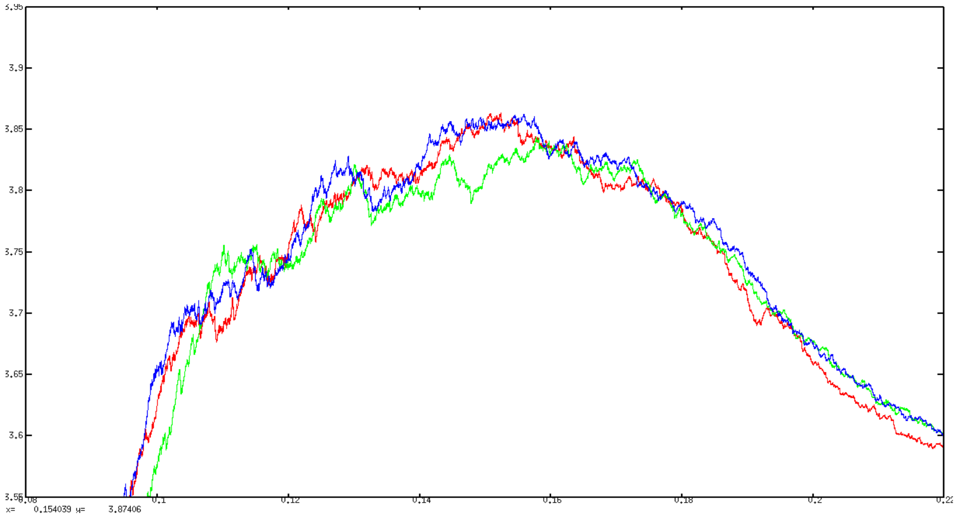


2014 December 13

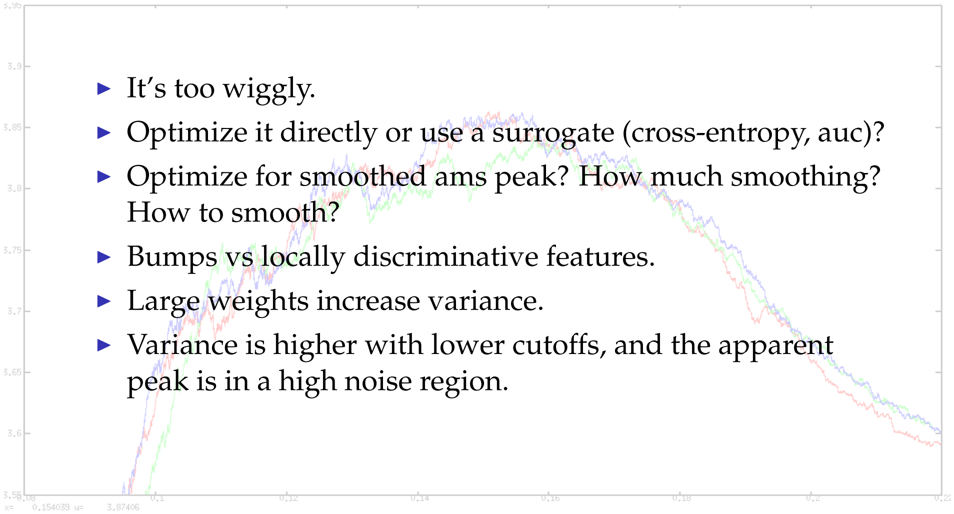


- Largest Kaggle competition to date.
- 4 months. Ouch.
- Classification task: separate tau-tau signal from background on simulated data (4-momenta of particles, some derived features).
- Unusual evaluation metric: AMS ($\sim s/\sqrt{b}$)
- Training set: 250000 labeled, weighted collision events
- Test set: 550000 events with unknown labels and weights.

PROBLEMS WITH AMS

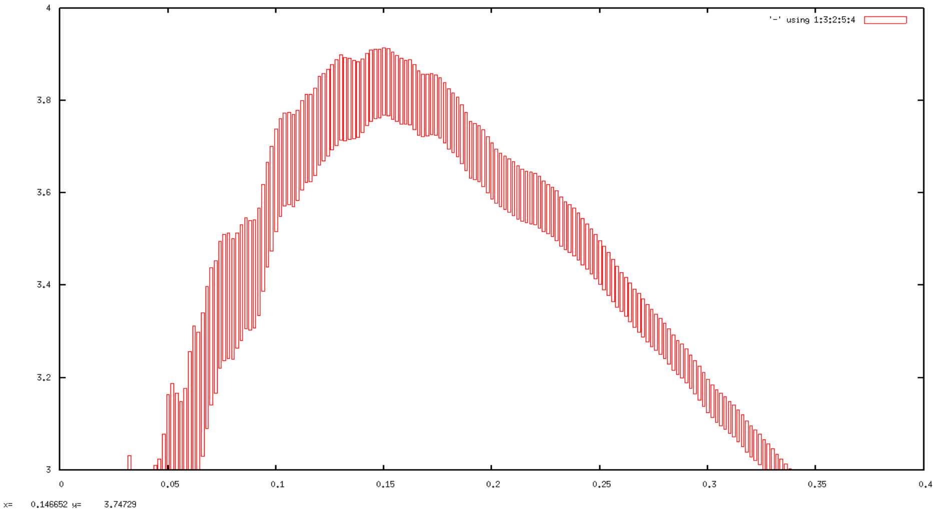


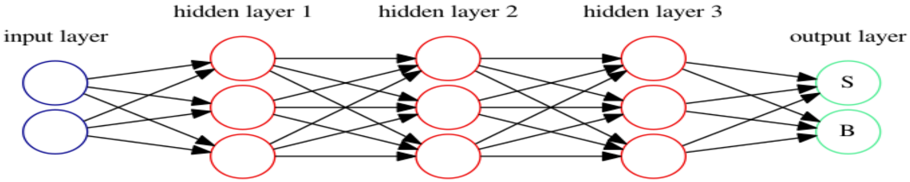
PROBLEMS WITH AMS



- It's too wiggly.
- Optimize it directly or use a surrogate (cross-entropy, auc)?
- Optimize for smoothed ams peak? How much smoothing? How to smooth?
- Bumps vs locally discriminative features.
- Large weights increase variance.
- Variance is higher with lower cutoffs, and the apparent peak is in a high noise region.

PROBLEMS WITH AMS





- inputs: normalized features (˜30), some log transformed
- 3 hidden layers of 600 neurons each
- output layer: 2 softmax units (one for signal, one for background)
- activation function: “max channel” in groups of 3
- trained to minimize cross entropy
- regularization: dropout on hidden layers, L1 + L2 penalty and a mild sparsity constraint input weights

NEURAL NETWORK PROS AND CONS

+ Powerful: better by ˜0.05 than Boosted Decision Trees (BDT, xgboost).
+ Your best bet to find interactions you don't know about.
- Somewhat sensitive to input encoding, normalization and also to correlated inputs (mitigated by bagging).
- Can take a long time to train.
- Lots of choices, parameters to tune (learning rate, momentum, number of layers, activation function, weight decay, weight constraints, etc).
- Trained model is difficult to interpret.

FEATURE ENGINEERING

- Log transforms, normalization.
- Deep NNs benefit less from feature engineering.
- Mass MMC and other derived features still useful with NNs.
- Additional CAKE features seemed to contribute ˜0.01 to BDTs, less clear with NNs.
- Evolutionary algorithms didn't find good features.

CROSS-VALIDATION

- Public leaderboard dataset too small, single cutoff, very unreliable.
- CV showed huge fluctuations across folds (± 0.15) which made comparisons almost impossible and slowed progress down.
- Since the AMS is non-linear, it seemed better to merge predictions from different folds and then calculate the AMS instead of averaging the AMSs of different folds.
- Even better: repeat CV with different splits and average the CV scores.

BAGGING

- Crowded decision boundary: AMS is very sensitive to minor differences in predicted probabilities.
- Model averaging to the rescue.
- Bagging: average models of the same kind trained on random subsets of the training set.
- Improved scores by about 0.1.

CV BAGGING

Repeated cross-validated bagging requires very many models to be trained (repetions x folds x bags). But there is a shortcut: one can get a more reliable CV score, a more accurate model *at the same time*, faster than with standard bagging by using the CV splits instead of random subsets of the training data.

```
for rep in [0..9]:
  shuffled = data.shuffle()
  for fold in [0..4]:
    [training, test] = shuffled.split_fold(fold, 5)
    train(newModel(), training, test)
    # Stash away the trained model (or only its
    # predictions). Merge and average predictions
    # of all models trained so far, calculate AMS.
```

ENSEMBLING

- CV bagged NNs: 3.83
- CV bagged xgboost: 3.79
- Weighted average of the above two: 3.84
- Anything more complicated overfitted.
- Evolutionary algorithms brought grief once again.

WHERE IS THE BOTTLENECK?

AMS is noisy, new features seem to help only marginally, feature generation overfits, optimizing AMS directly overfits, even ensembling can overfit. Data is extremely expensive. Simulated data is simply is just very, very expensive.

Where is the Bottleneck?

AMS is noisy, new features seem to help only marginally, feature generation overfits, optimizing AMS directly overfits, even ensembling can overfit. Data is extremely expensive. Simulated data is simply is just very, very expensive. We need better regularization or much more data.

WHERE IS THE BOTTLENECK?

AMS is noisy, new features seem to help only marginally, feature generation overfits, optimizing AMS directly overfits, even ensembling can overfit. Data is extremely expensive. Simulated data is simply is just very, very expensive.
We need better regularization or much more data.

- Incorporate prior knowledge on what kind of interactions are expected (for example, by restricting connections in the neural network topology).
- Train a generative model that mimicks the simulator (i.e produces data, label pairs) and is quick.

FUTURE DIRECTIONS

A generative model could:

- act as a generative classifier itself
- or generate more training data cheaply for a discriminative classifier. Ideally, the generative model is good enough so that its samples are indistinguishable from real data. See Generative Adversarial Networks for an example of how to train a Generative and a Discriminative neural network jointly.