Calibration curves as features for tuning hyperparameters

Feb 2016

Artem Vorozhtsov, Yandex
avorozhtsov@yandex-team.ru
Hyperparameters, aka H-space

- Model
  - Architecture
  - Method
    - Model prior
    - Data preprocessing
    - Regularization
    - Prediction post processing (isotonic regression)
Hyperparameters, aka H-space

• Model families:
  – SVM
  – Bayes
  – Logistic
  – Random Forest
  – Gradient Boosted Trees
  – Neural Networks
  – ...

Hyperparameters, aka H-space

• Model
  – Gradient Boosted Trees (GBT)
• Architecture
  – trees depth
• Method
  – feature discretization algorithm
  – Newton method for calculating values at leafs
• Model prior, starting point
  – baseline from a simple predictor
• Regularization parameters
  – number of iterations, learning rate
Hyperparameters, aka H-space

• Model
  – Gradient Boosted Trees (GBT)

• Architecture
  – trees depth

• Method
  – feature discretization algorithm
    – Newton method for calculating values at leaves

• Model prior, starting point
  – baseline from a simple predictor

• Regularization parameters
  – number of iterations, learning rate

• Weights at leaves
Hyperparameters, aka H-space

- Model
  - Gradient Boosted Trees
- Architecture
  - trees depth
- Method
  - Newton for greed steps
  - feature discretization algorithm
- Model prior
  - baseline from simple predictor
- Regularization parameters
  - number of iterations, learn rate
- Weights at leafs

H-space
Hyperparameter Optimization


- Grid search & random search
- Bayesian model selection (GP, TPE)
- Search in parallel
- Meta-learning
  - Initialization of prior dist. of model over H-space
Hyperparameter Optimization

- grid search and random search
  - 2011, “Random search for hyper-parameter optimization”, J. Bergstra and Y. Bengio

Random search is better!

Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of $g$. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.
Hyperparameter Optimization

J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization”

Random search is better!

Figure 1: Grid and random search of nine trials for optimizing a function $f(x, y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of $g$. This failure of grid search is the rule rather than the exception in high dimensional hyper-parameter optimization.
Hyperparameter Optimization

- Bayesian approach, Gaussian Processes
  - Spearmint, https://github.com/HIPS/Spearmint
    - Gaussian Processes (GP)
Hyperparameter Optimization

Meta-learning

- transfer learning for H-space.
- 2015, Initializing Bayesian Hyperparameter Optimization via Meta-Learning, Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter

Meta-features are properties of a dataset.

- 57 datasets
- 46 meta-features
- Combined Algorithm Selection and Hyperparameter optimization (CASH) setting by Thornton, namely CASH(SVM_rbf + SVM_linear + RF)

The recipe: Spearmint + CASH(SVM_rbf + SVM_linear + RF)
Hyperparameter Optimization

Some software packages

- Spearmint (Python, Gaussian Processes)
- BayesOpt (C++ with Python and Matlab/Octave interfaces)
- hyperopt (Python, TPE)
- SMAC (Java, Random Forests)
- REMBO (Matlab)
- MOE (C++/Python)
Hyperparameter Optimization

Goals & Results

• Save CPU time
• Improve Prediction
Hyperparameter Optimization

Why is prediction improved (meta-learning VS exhaustive random search) ?

1. Difference between optimizing of

\[ P(D \mid \text{model}) \text{ and } P(\text{model} \mid D) \]

for small datasets.

Exhaustive hyperparameters optimization may introduce some overfitting.

But, for big datasets it is not the case:

\[ L = P(D \mid \text{..}) = P(d_1 \mid \text{..}) \times P(d_2 \mid \text{..}) \times \ldots \times P(d_N \mid \text{..}) \]

Multiplier \( P(\text{model}) \), i.e. the model prior, does not influence much LogLoss:

\[ \log(L \times P(\text{model})) / N \approx \log(L) / N \]
Hyperparameter Optimization

Why is prediction improved
(meta-learning VS extensive random search) ?

2. Random search is not so exhaustive.

H-space is a magic. Especially because of NN.

It has many secret places and meta-learning magic finds them.

Random walk does not work.
Hyperparameter Optimization

Random walk does not work.
Hyperparameter Optimization


Hyperparameter Optimization


Hyperparameter Optimization


Hyperparameter Optimization

Sometimes

- It’s just black box optimization.
- They don’t make use of any H-space properties
- It’s just final metrics pursuit (log_loss, roc_auc, ...)

Hyperparameter Optimization

Shot(hyper-point) ➔

LossFunction,

ROC curve,

Calibration curve,

...
Calibration curve

• Predicted Value vs Real Value

• Ideal calibration curve is $y = x$

• For classification problem:
  – 100 bins for predicted probability
  – calculate mean predicted probability and fraction of positives for each bin
Calibration curve

mean predicted value

VS

fraction of positives
Its not a big deal
Interview questions

These are test and train calibration curves.

- Which one is a test curve?
- Do I need to increase the regularization parameter?
- Is the prior distribution correct?
Interview questions

These are test and train calibration curves.

- Which one is a test curve?
- Do I need to increase the regularization parameter?
- Is the prior distribution correct?

prior is not correct, it’s too big

slope < 1 => need more regularization

test
Question: Reconstruct legend
Interview questions

Question: Reconstruct legend.
How they could be improved?
A simple Bayesian model with independent features is a source of canonical parameters of calibration curves “topology” and they could be transferred to H-space.
Canonical parameters in H-space

- **Slope – regularization**
  \[ \tan(\alpha), \text{ should be } 1 \]

- **Shift – prior**
  \[ \text{distance, should be } 0 \]

- **Angle – fit metrics**
  \[ \text{angle in radians, should be } 0 \]
Canonical parameters in H-space
Canonical parameters in H-space
Canonical parameters in H-space

Three points is enough to find point in H-space with Slope=1 and Shift=0
Canonically parameters in H-space

• Slope
  – canonical regularization parameter
  – change the slopes of train and test curves in the same way
  – test and train calibration curves intersect at the same Y.
Canonical parameters in H-space

• Shift
  – shift of prior from true prior
  – change the position of the intersection of test and train curves
  – does not change the slopes (and the angle between curves) in the intersection point
Canonical parameters in H-space

- **Angle**
  - data quantity metrics
  - can be treated as one of final metrics
  - direction of decreasing *Angle* in H-space is a real finding
    (it’s like open your eyes and see more information in the train dataset)
Canonical parameters in H-space

- Bayesian model
  - features are categorical and independent
  - prior is beta-distribution with \((\alpha, \beta)\)
  - parameters I’ve chosen:
    - \(p = \frac{\alpha}{\alpha + \beta}\)
    - \(r = \sqrt{\alpha^2 + \beta^2}\)
    - \(L = \text{size of train set}\)
The proposal

• Use a simple Bayesian model as a source of canonical hyperparameters.
• Map them to the properties of the calibration and ROC curves for test and train sets.
• Find canonical parameterization of any H-space for any classification problem.
• Make use of this parameterization in meta-learning algorithms
The proposal

Hints from a teacher

It’s like advices from chef:

• more milk
• more sugar
• less pepper

It’s more informative than just score for you dish.
The proposal

This is exactly what V. Vapnik proposed in 2009:

During the learning process a teacher supplies training example with additional information which can include comments, comparison, explanation and so on.

This information is available only for the training examples. It will not be available (hidden) for the test examples.

Hidden information can play an important role in the learning process.

V. Vapnik: Learning Using Hidden Information
The proposal

Learning Using Hidden Information (LUHI)

“The situation with existence of hidden information is very common. In fact, for almost all machine learning problems there exists some sort of hidden information.”
The proposal

Learning Using Hidden Information (LUHI) for meta-learning

Meta-learning algorithm

Learning Using Hidden Information (LUHI)

- online setting
- Meta-Learning provides ranking scores
  - Bayesian model generates candidates with estimates of acquisition values $\alpha_i$
  - ML-model provide score for candidates $\text{score}_i$
  - Provide these scores to Bayesian model and recalculate $\alpha_i$
Features for meta-learning

Learning Using Hidden Information (LUHI) for meta-learning

- Meta-features
- Hidden Meta-features (hint features from teacher)
**Meta-features**

Table 1. List of implemented metafeatures

<table>
<thead>
<tr>
<th>Simple metafeatures:</th>
<th>Statistical metafeatures:</th>
<th>PCA metafeatures:</th>
<th>Landmarking metafeatures:</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of patterns</td>
<td>min # categorical values</td>
<td>pca 95%</td>
<td>One Nearest Neighbor</td>
</tr>
<tr>
<td>log number of patterns</td>
<td>max # categorical values</td>
<td>pca skewness first pc</td>
<td>Linear Discriminant Analysis</td>
</tr>
<tr>
<td>number of classes</td>
<td>mean # categorical values</td>
<td>pca kurtosis first pc</td>
<td>Naive Bayes</td>
</tr>
<tr>
<td>number of features</td>
<td>std # categorical values</td>
<td></td>
<td>Decision Tree</td>
</tr>
<tr>
<td>log number of features</td>
<td>total # categorical values</td>
<td></td>
<td>Decision Node Learner</td>
</tr>
<tr>
<td>number of patterns with missing values</td>
<td>kurtosis min</td>
<td></td>
<td>Random Node Learner</td>
</tr>
<tr>
<td>percentage of patterns with missing values</td>
<td>kurtosis max</td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of features with missing values</td>
<td>kurtosis mean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>percentage of features with missing values</td>
<td>kurtosis std</td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of missing values</td>
<td>skewness min</td>
<td></td>
<td></td>
</tr>
<tr>
<td>percentage of missing values</td>
<td>skewness max</td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of numeric features</td>
<td>skewness mean</td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of categorical features</td>
<td>skewness std</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ratio numerical to categorical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ratio categorical to numerical</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dataset dimensionality</td>
<td>log inverse dataset dimensionality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>log dataset dimensionality</td>
<td>inverse dataset dimensionality</td>
<td></td>
<td></td>
</tr>
<tr>
<td>log inverse dataset dimensionality</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class probability min</td>
<td></td>
<td>pca kurtosis first pc</td>
<td></td>
</tr>
<tr>
<td>class probability max</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class probability mean</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class probability std</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information-theoretic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>metafeature:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>class entropy</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Meta-features

Learning Using Hidden Information (LUHI) for meta-learning

- Meta-features (features about dataset)
  - statistics, skewness, kurtosis
  - entropies
  - PCA metafeatures
  - Landmarking metafeatures
    » Properties of predictions of fast predictors on the part of datasets
      • performance
      • calibration curves
Hidden Meta-features

Learning Using Hidden Information (LUHI) for meta-learning

- Hidden meta-features
  - All the metrics (LogLoss, AUC, F1, ...)
  - Properties of calibration curves (parameters of bet fits of calibration test and train curves)
  - Properties of ROC
  - Properties of learn-curves
Hidden Meta-features
Hidden Meta-features

![ROC Curve Diagram]

AUC = 0.78285180

[0.1, 0.3, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 5]
Hidden Meta-features
Meta-learning algorithm

Learning Using Hidden Information (LUHI) for meta-learning

- online setting

- Meta-Learning provides ranking scores
  - Bayesian model generates candidates with estimates of acquisition values $\alpha_i$
  - ML-model provide score for candidates $score_i$
  - Provide these scores to Bayesian model and recalculate $\alpha_i$
Questions?
Titles for this presentation

- PCA for hyperparameters
- Intuition from calibration curves and other curves
- Calibration curve as a feature source for meta-learning (optimizing hyperparameters)
- Model-selection based on calibration curves
- Canonical parameterization of H-space
- LUHI for meta-learning