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 leafs
- Model prior, starting point
  - baseline from a simple predictor
- Regularization parameters
  - number of iterations, learning rate
- Weights at leafs

Oops!

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

H-space

- Weights at leafs

space

Hyperparameter Optimization

F. Hutter, H. Hoos, K. Leyton-Brown, J. Bergstra, J. Snoek, H.Larochelle, R.P. Adams, Y. Bengio, M. Feurer, J.T. Springenberg:

- 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

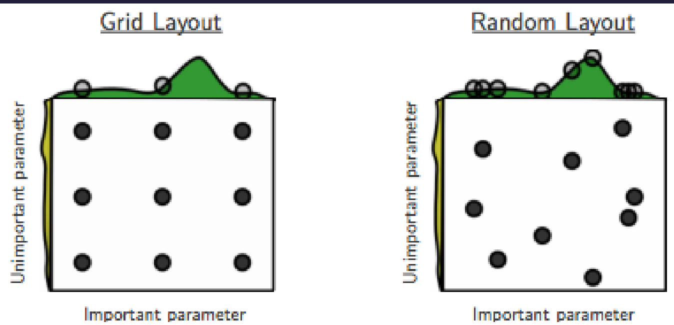


Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct 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.

Random search is better!

Hyperparameter Optimization

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

Random search is better!

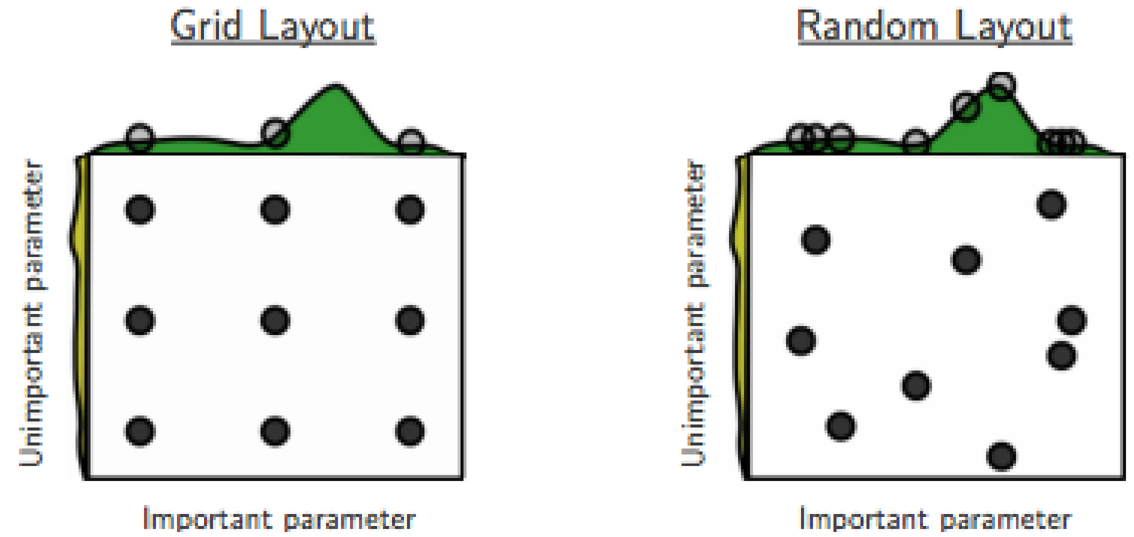


Figure 1: Grid and random search of nine trials for optimizing a function $f(x,y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct 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
  - 2012, Practical Bayesian Optimization of Machine Learning Algorithms, Jasper Snoek, Hugo Larochelle, and Ryan Prescott Adams
    - 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

$$P(D \mid model) \text{ and } P(model \mid D)$$

for small datasets.

Exhaustive hyperparameters optimization may introduce some overfitting.

But, for big datasets it is not the case:

$$L = P(D| ..) = P(d\_1|..) \times P(d\_2|..) \times \ldots \times P(d\_N| ..)$$

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

$$\log(L \times P(model)) / N \approx \operatorname{Log}(L) / N$$

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

[Czo-05] I. Czogiel, K. Luebke, and C. Weihs. Response surface methodology for optimizing hyper parameters. Technical report, Universit¨at Dortmund Fachbereich Statistik, September 2005.

[Hut-09] Frank Hutter. Automated Configuration of Algorithms for Solving Hard Computational Problems. PhD thesis, University of British Columbia, 2009

[Hal-09] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update, ACM SIGKDD explorations newsletter, 11(1):10-18, 2009.

Hyperparameter Optimization

[Ber-11] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl. Algorithms for hyper-parameter optimization, NIPS, 24:2546–2554, 2011.

[Hut11] F. Hutter, H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration, LION-5, 2011. Extended version as UBC Tech report TR-2010-10.

[Ber-13a] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures, In Proc. ICML, 2013.

[Ber-13b] J. Bergstra, D. Yamins, and D. D. Cox. Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms, SciPy'13, 2013.

Hyperparameter Optimization

[Cir-12] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column Deep Neural Networks for Image Classification, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3642-3649. 2012.

[Dom14] T. Domhan, T. Springenberg, F. Hutter. Extrapolating Learning Curves of Deep Neural Networks, ICML AutoML Workshop, 2014.

[Egg13] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown. Towards an empirical foundation for assessing bayesian optimization of hyperparameters, NIPS workshop on Bayesian Optimization in Theory and Practice, 2013.

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,

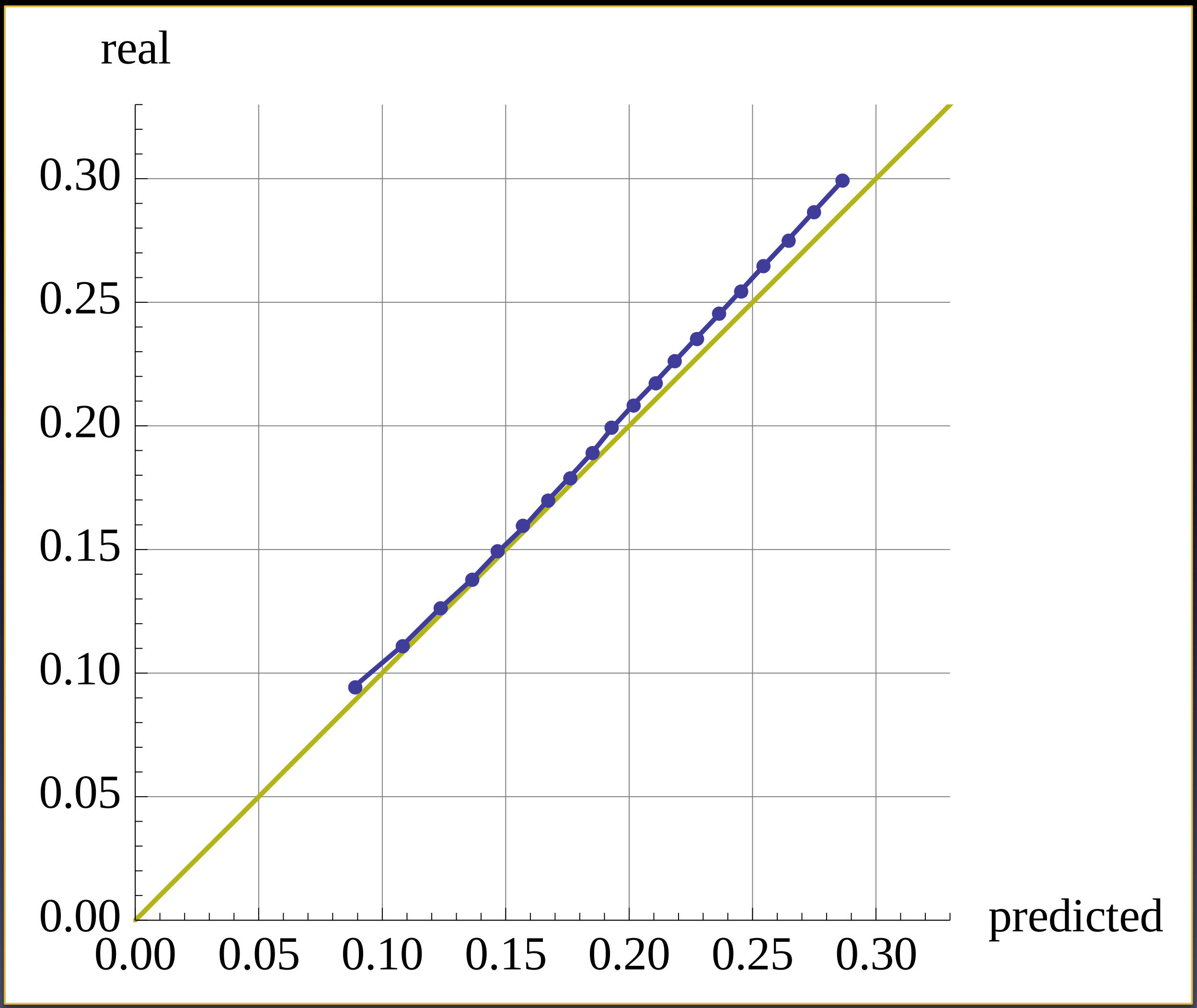
Calibration curve,

…

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



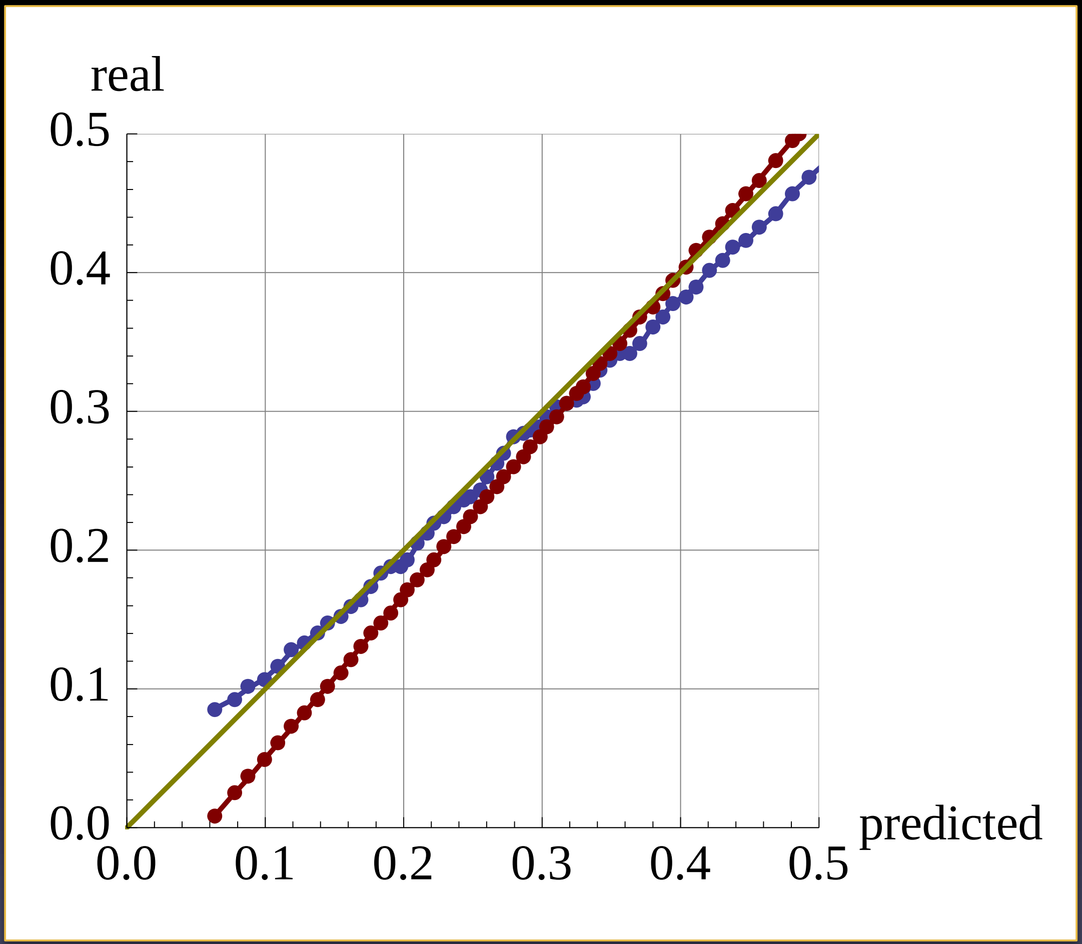
mean predicted value

VS

fraction of positives

Its not a big deal

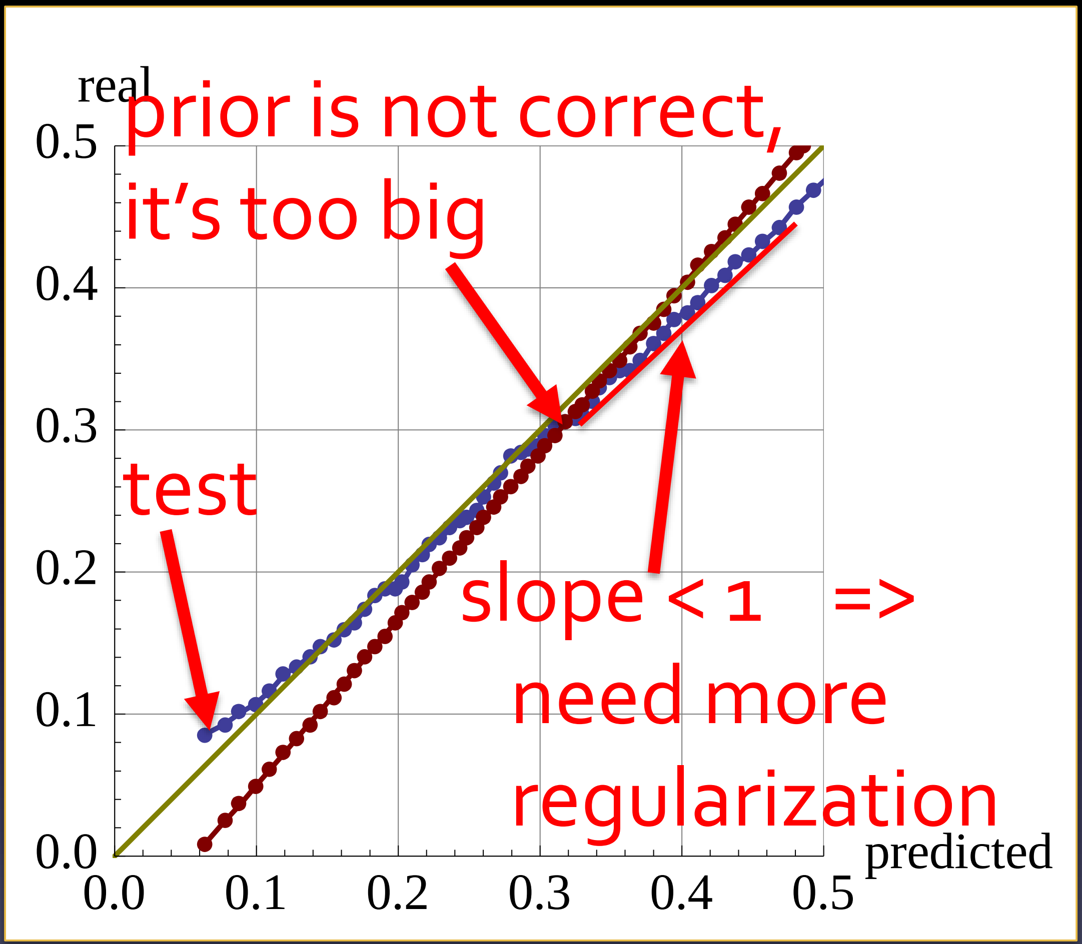
Interview questions



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?

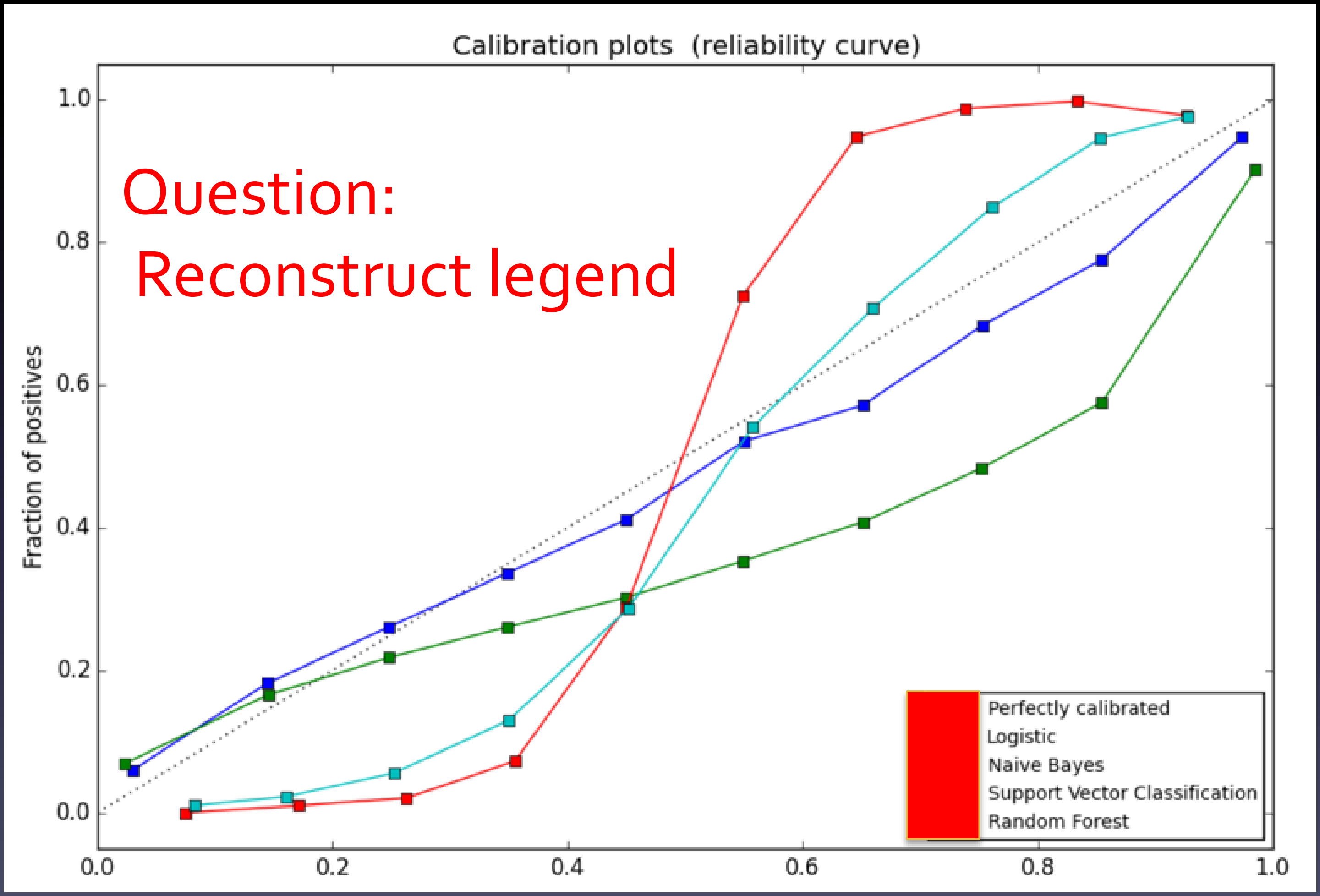
Interview questions



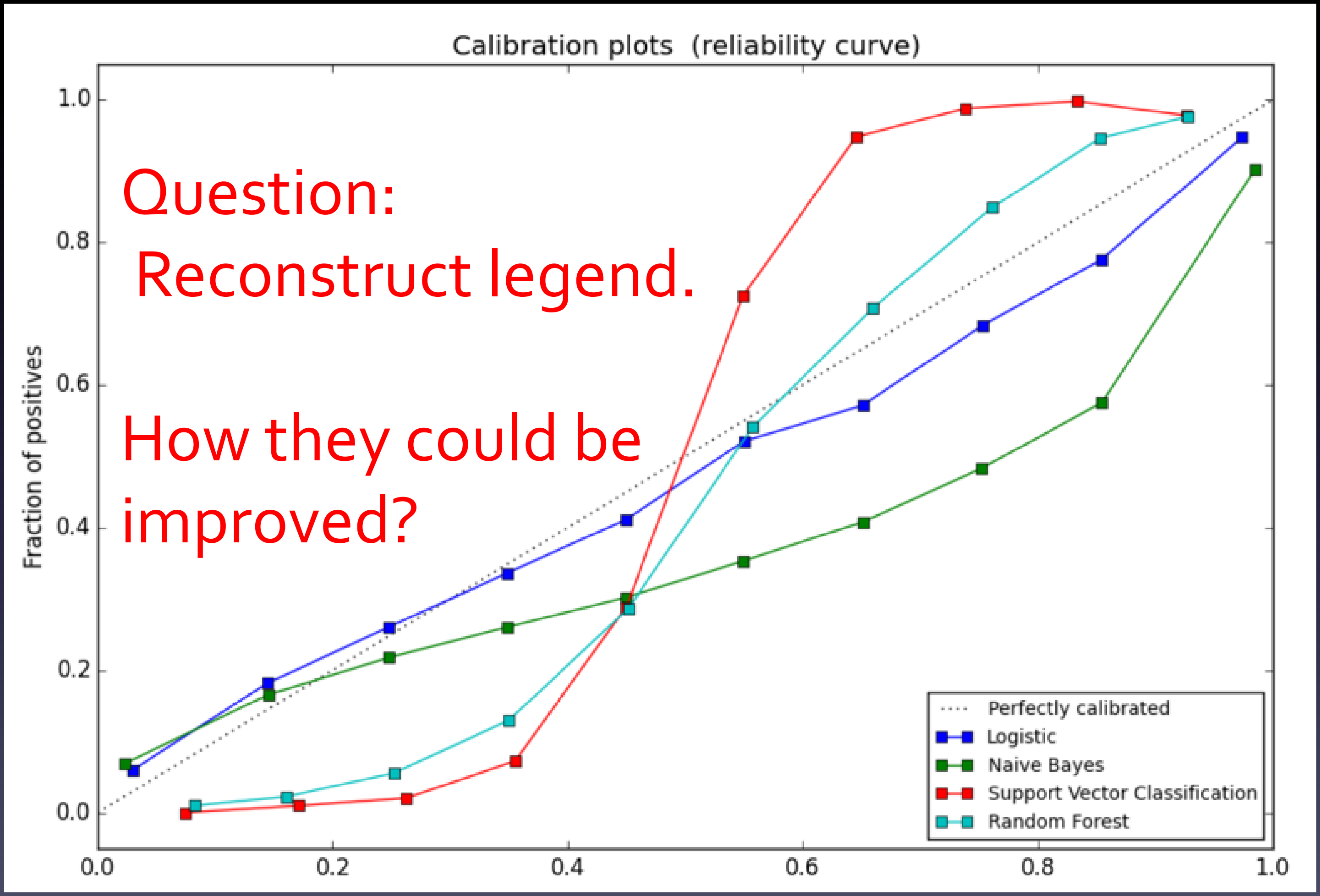
These are test and train calibration curves.

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

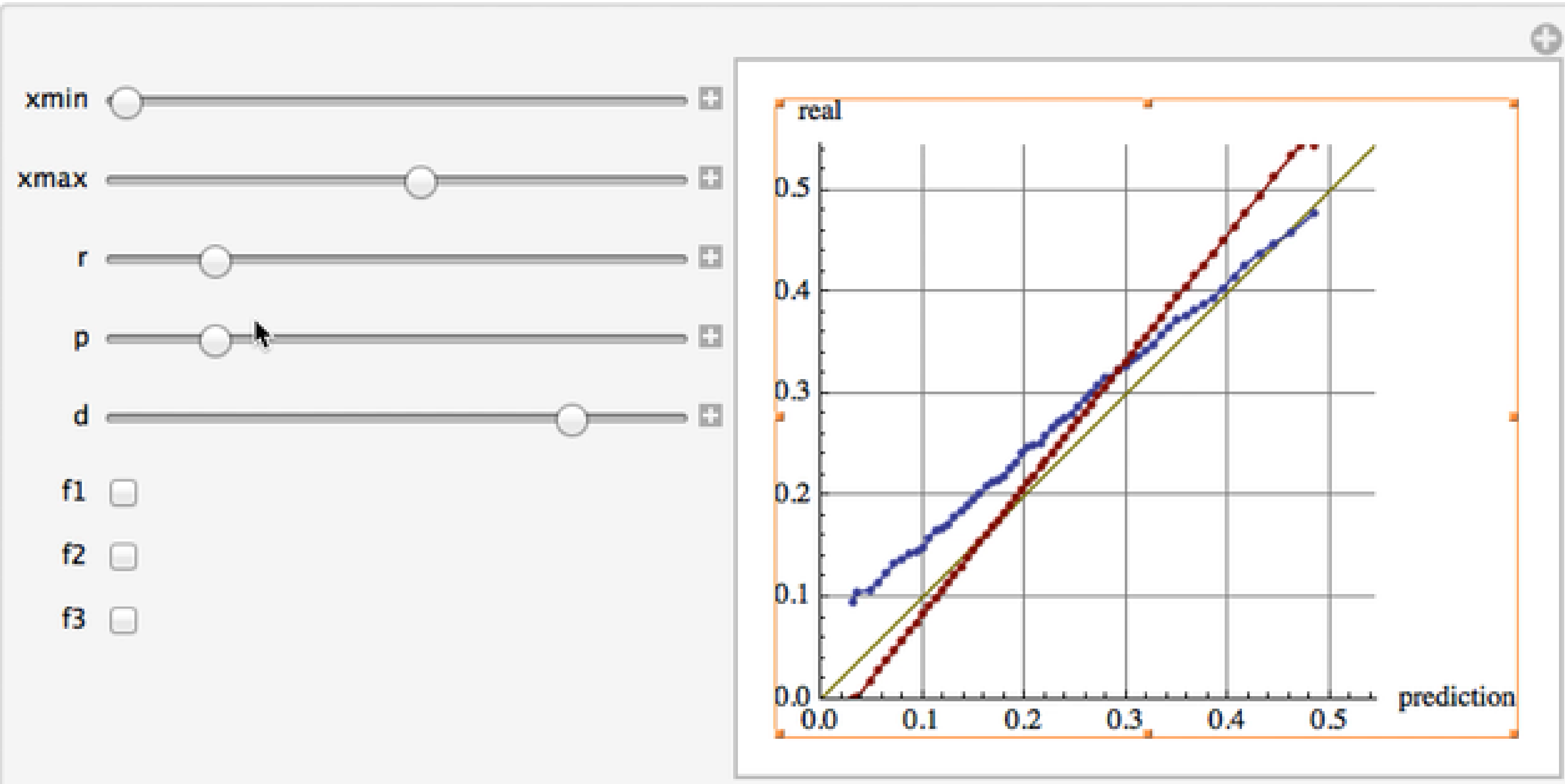
Interview questions



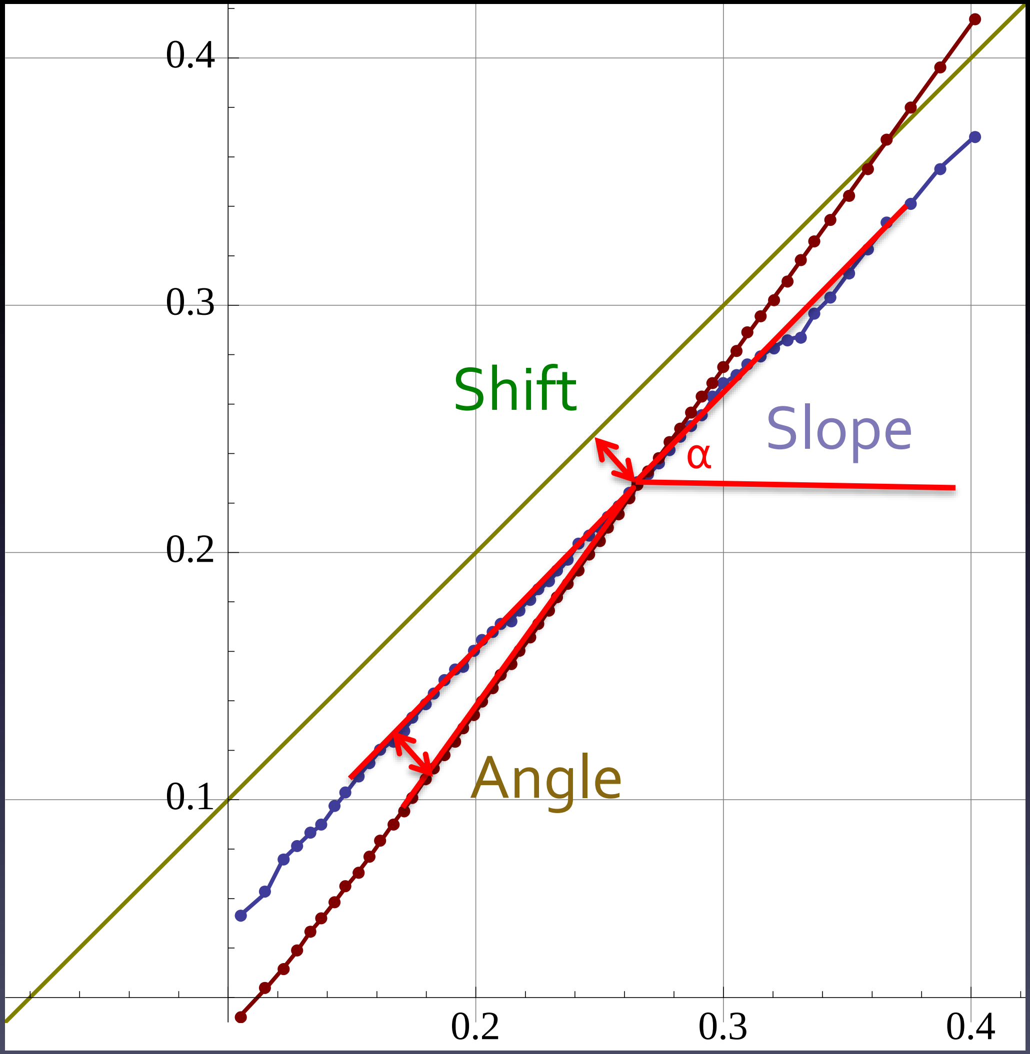
Interview questions



Mathematica Demo 1



Canonical parameters in H-space



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(α) , should be 1

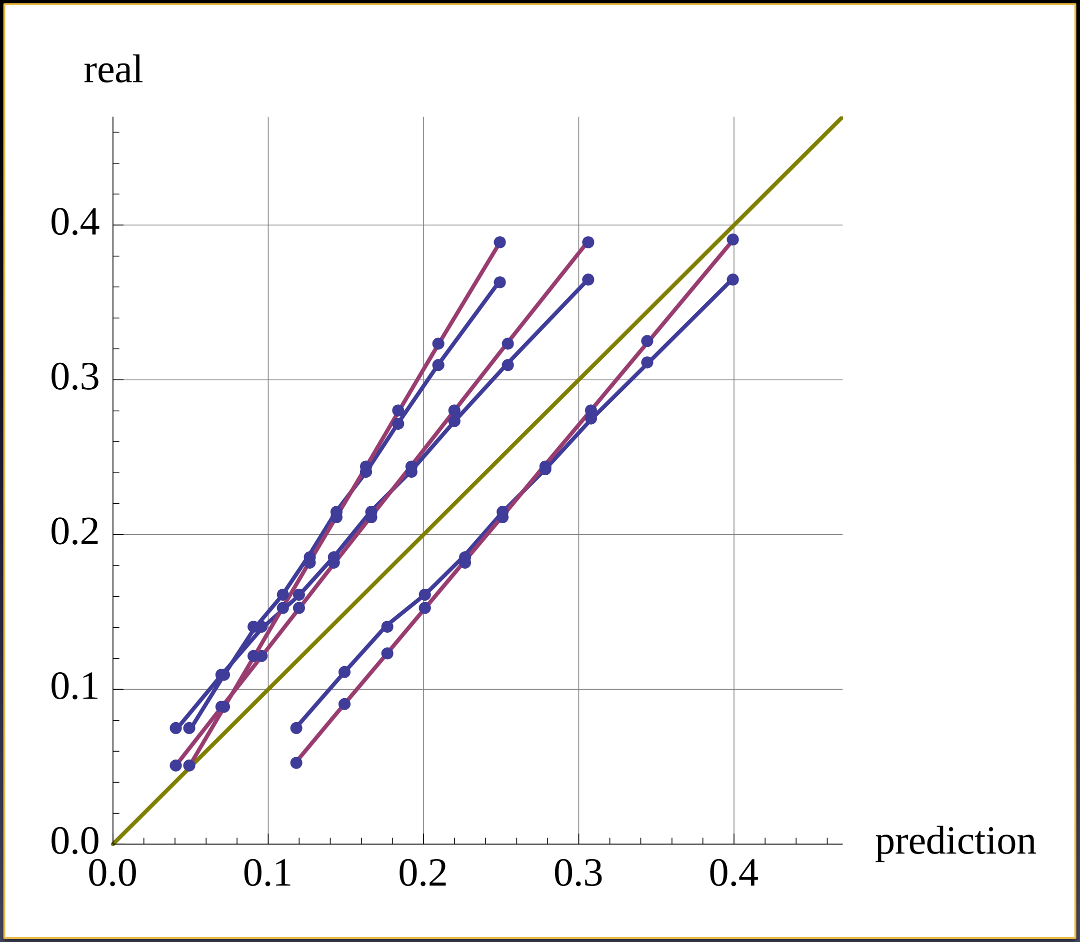
- Shift – prior

  = distance, should be 0

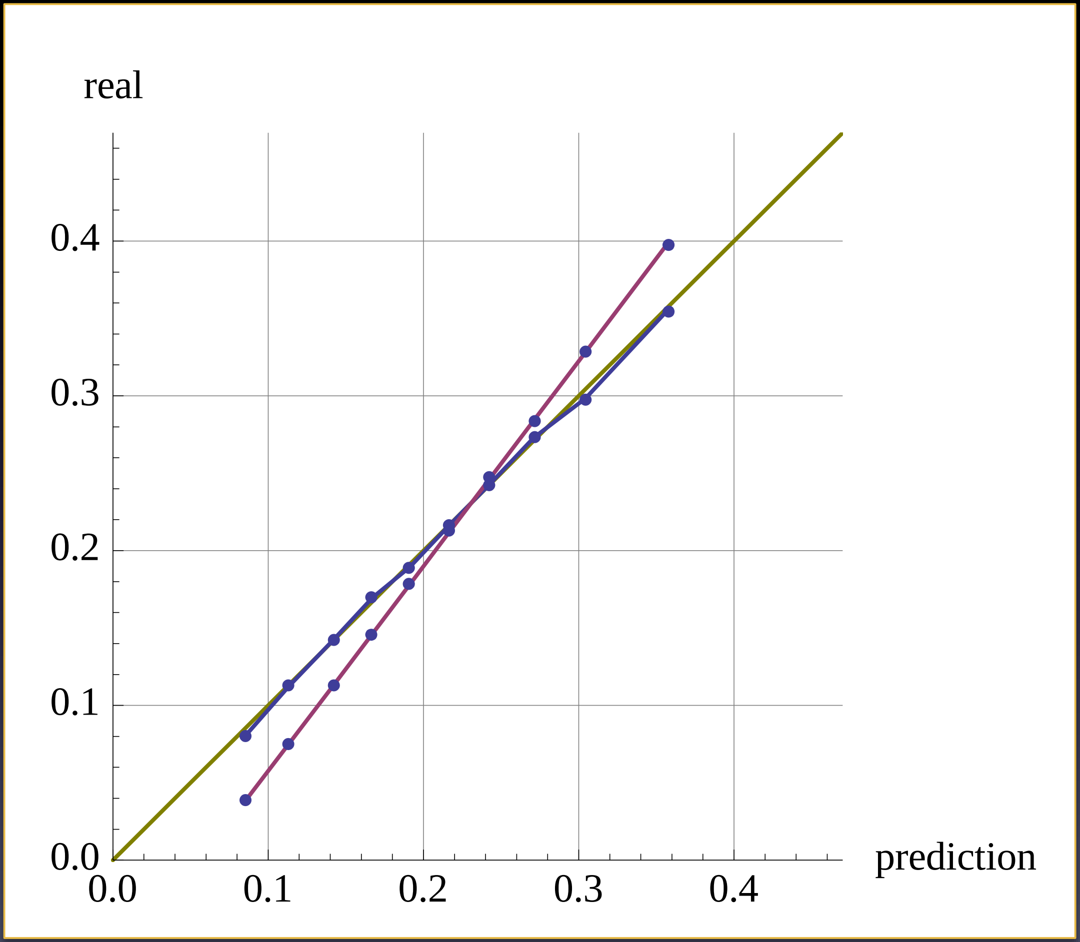
- Angle – fit metrics

  = 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

Canonical 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.

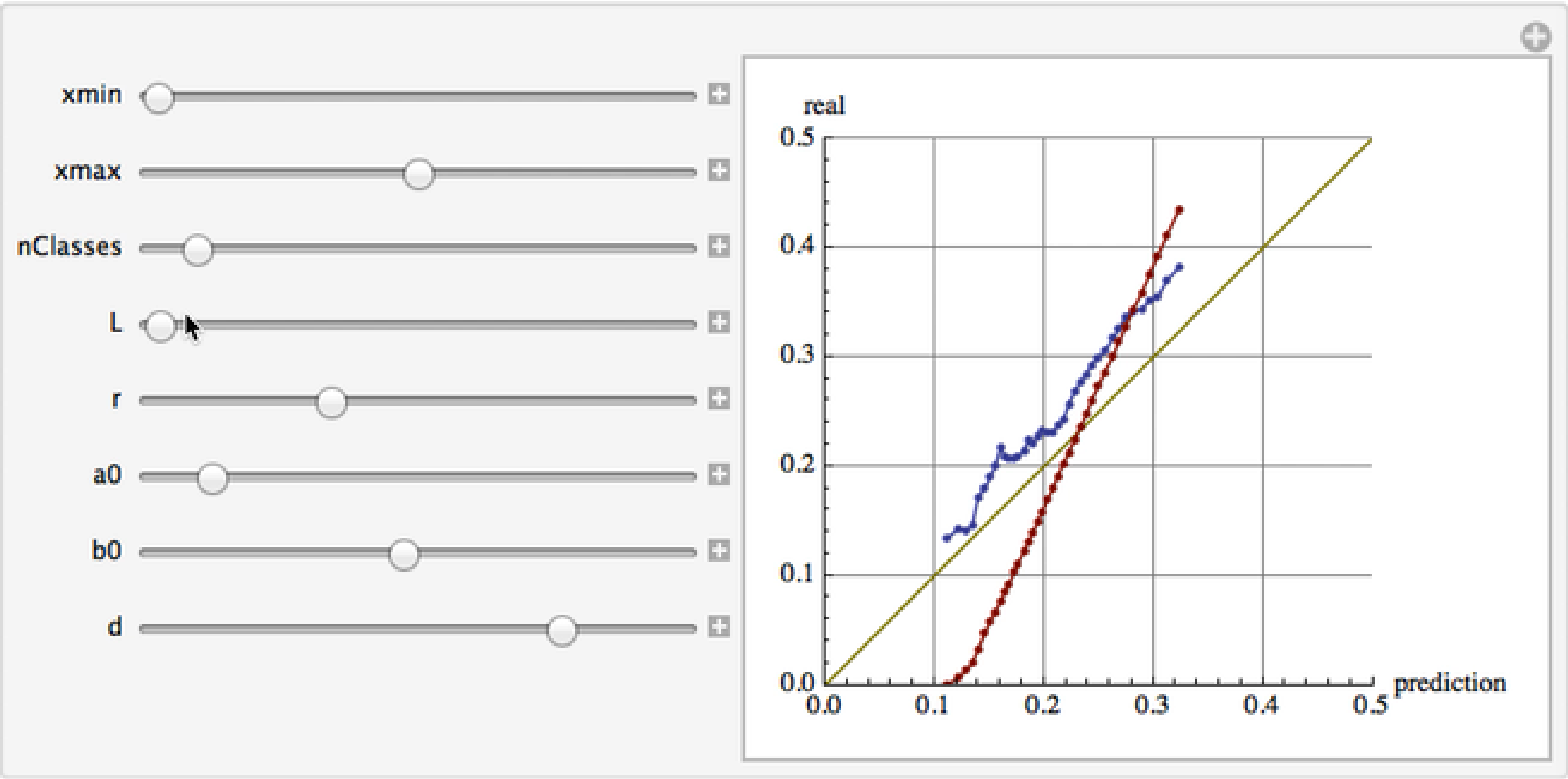
Canonical parameters in H-space

- 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)

Mathematica Demo 2



Canonical parameters in H-space

- Bayesian model
  - features are categorical and independent
  - prior is beta-distribution with (α, β)
  - parameters I've chosen:
    - **p** = $\alpha \; / (\alpha + \beta)$
    - **r** = sqrt(α^2 + β^2)
    - **L** = 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

http://www.cs.princeton.edu/courses/archive/sprin
g13/cos511/handouts/vapnik-slides.pdf

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

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

Simple metafeatures:
number of patterns
log number of patterns
number of classes
number of features
log number of features
number of patterns with missing values
percentage of patterns with missing values
number of features with missing values
percentage of features with missing values
number of missing values
percentage of missing values
number of numeric features
number of categorical features
ratio numerical to categorical
ratio categorical to numerical
dataset dimensionality
log dataset dimensionality
inverse dataset dimensionality
log inverse dataset dimensionality
class probability min
class probability max
class probability mean
class probability std

Information-theoretic metafeature:
class entropy

Statistical metafeatures:
min # categorical values
max # categorical values
mean # categorical values
std # categorical values
total # categorical values
kurtosis min
kurtosis max
kurtosis mean
kurtosis std
skewness min
skewness max
skewness mean
skewness std

PCA metafeatures:
pca 95%
pca skewness first pc
pca kurtosis first pc

Landmarking metafeatures:
One Nearest Neighbor
Linear Discriminant Analysis
Naive Bayes
Decision Tree
Decision Node Learner
Random Node Learner

Initializing Bayesian Hyperparameter Optimization via Meta-Learning,

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter, 2015

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**

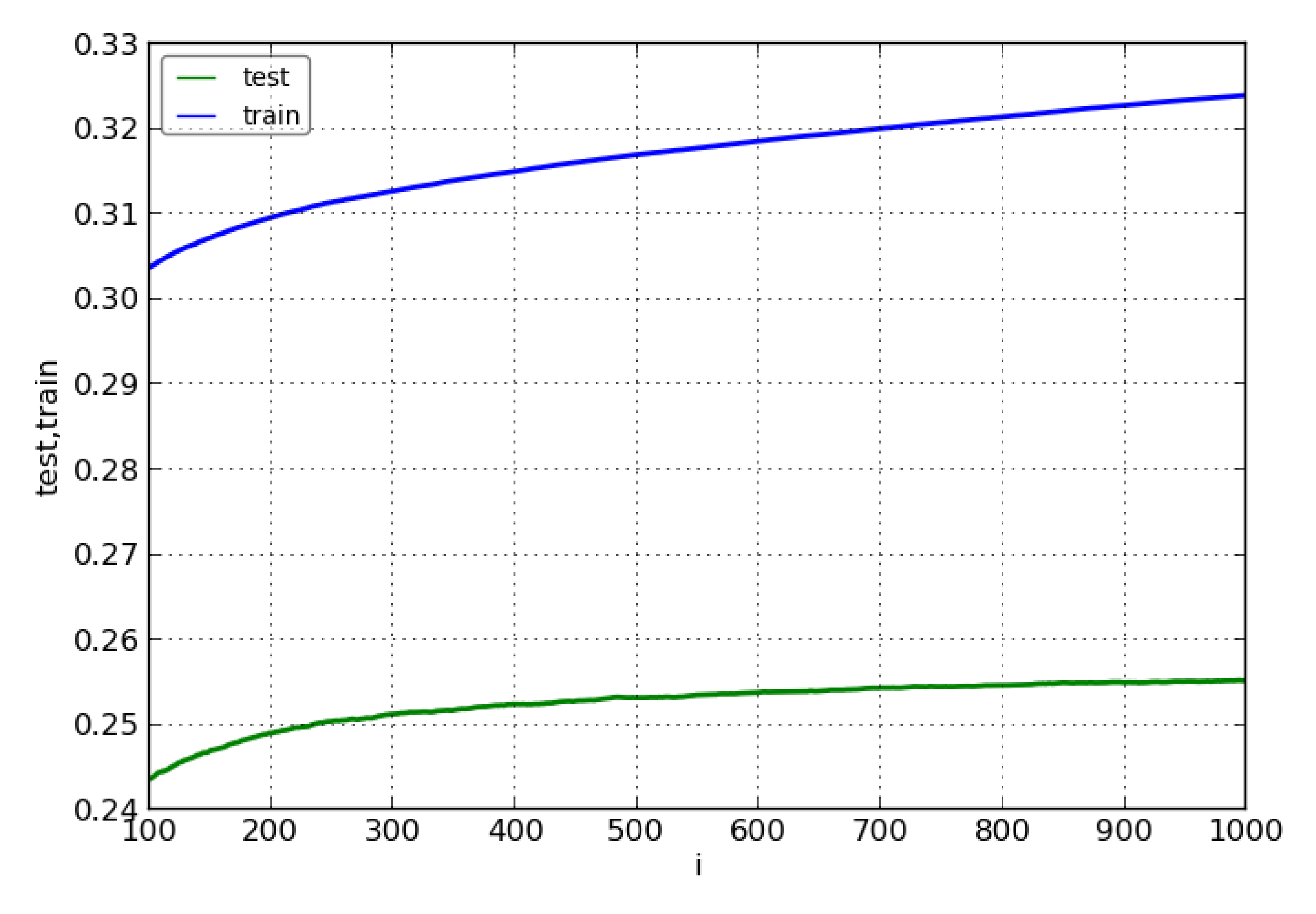
Hidden Meta-features

Learning Using Hidden Information (LUHI)

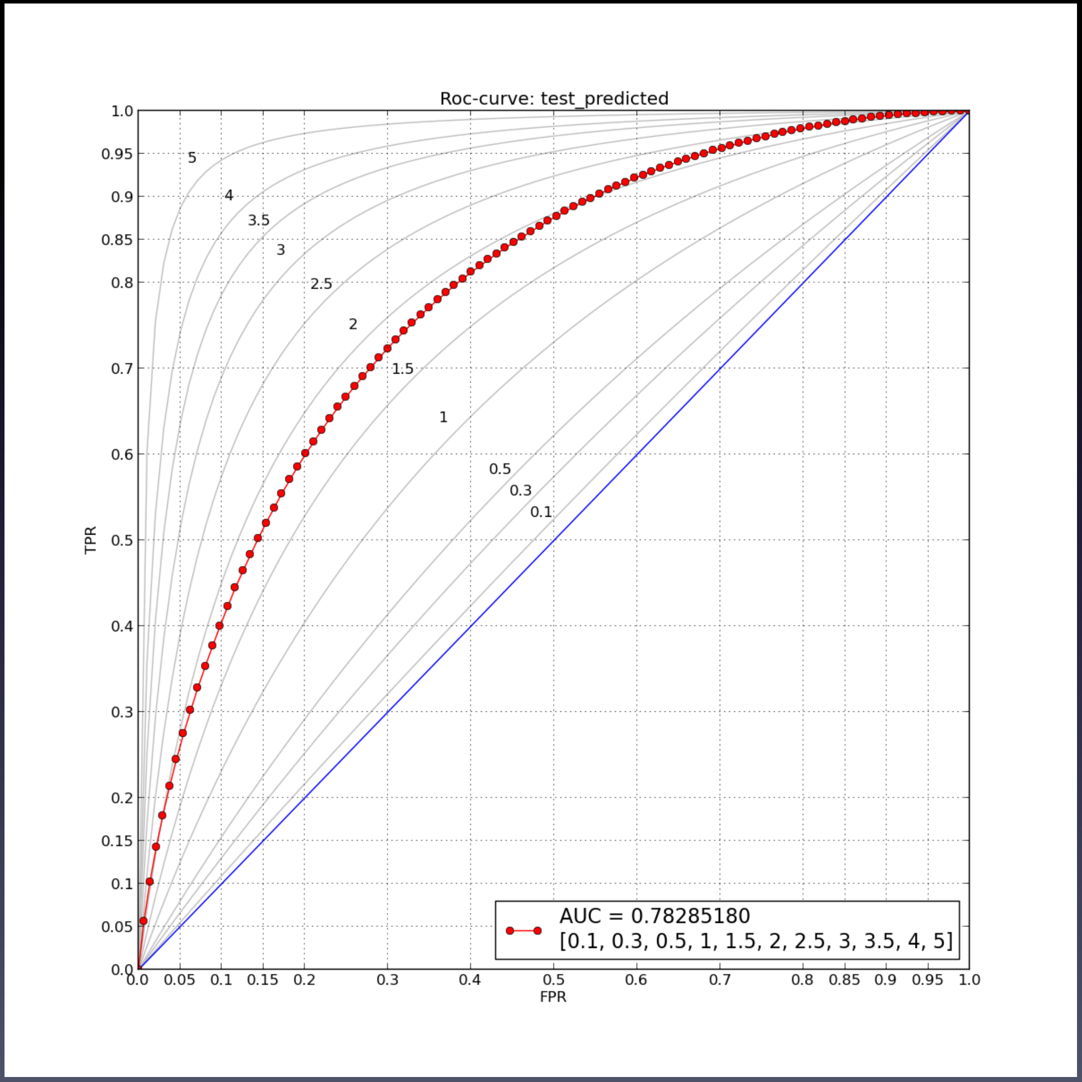
for meta-learning

– 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

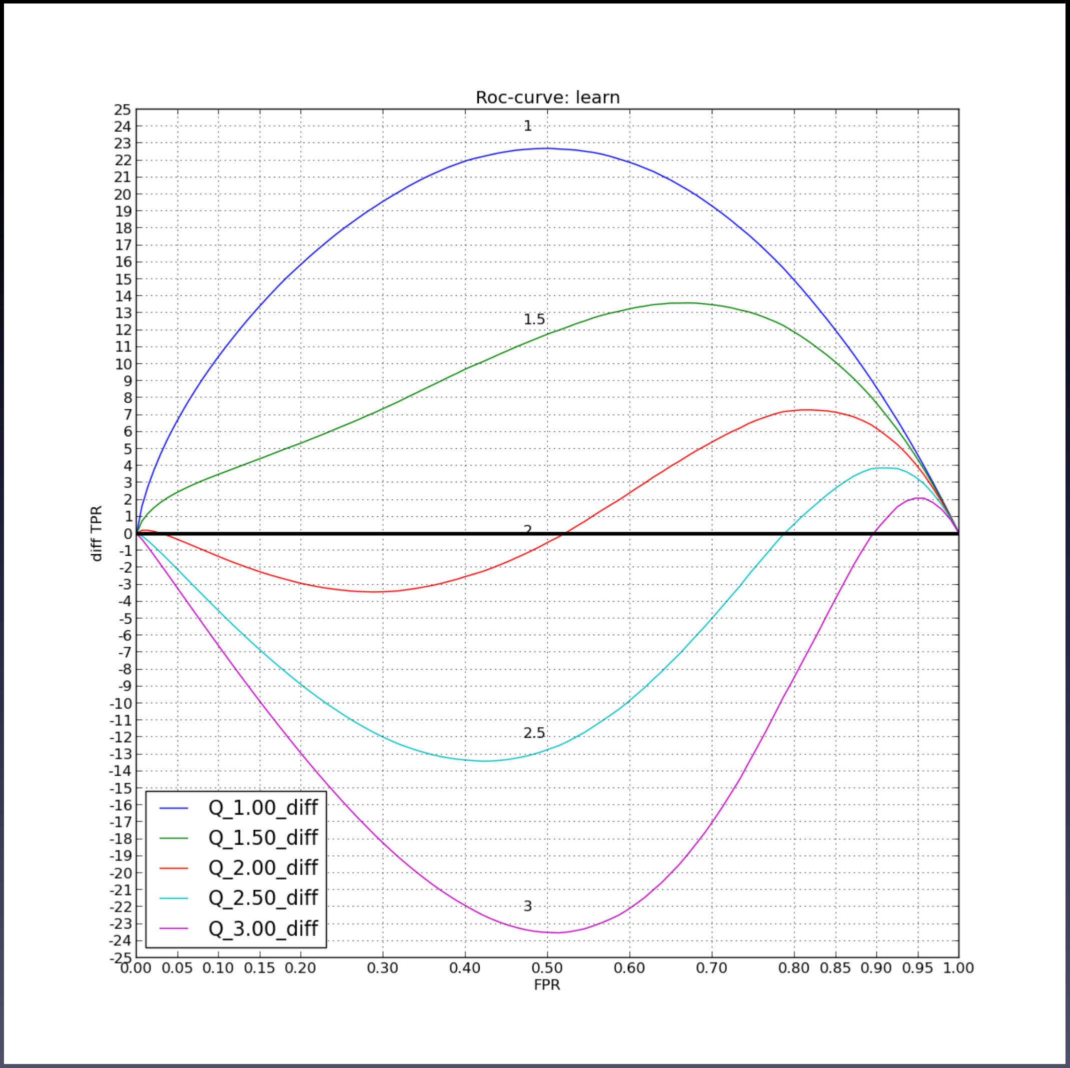
Hidden Meta-features



Hidden Meta-features



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