Machine Learning and Data Mining

Meta-learning, lecture

Maxim Borisyak

2016/09/22

National Research University
Higher School of Economics

HIGHER SCHOOL OF ECONOMICS
NATIONAL RESEARCH UNIVERSITY

Meta-Learning

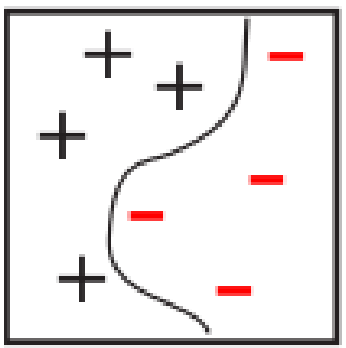
Meta-Learning

As we have seen a Machine Learning is about selecting well-suited algorithm for given problem (data).

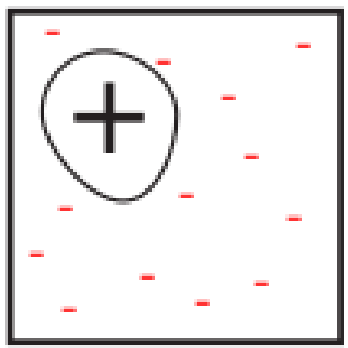
- Inventing a very specific learning procedure that reflects all prior assumptions and understanding of problem can be hard.
- On the other hand, the is a lot of general purpose algorithms like:
  - SVM;
  - Decision Trees;
  - Logistic Regression.

Question

How can we make a specific learning procedure out of general ones?



Weak learner



Strong learner

Bias-Variance trade-off [1]

- underlying function ('true' function) $h(x) = \mathbb{E}[t \mid x]$;
- regressor $y(x)$;
- mean squared loss: $L = \frac{1}{n}\sum_{i=1}^{n}(h(x) - t)^2$

$$\mathbb{E}[L] = \int (y(x) - h(x))^2 p(x) dx + \iint (h(x) - t)^2 p(x, t) dx dt$$

- second term is irreducible, intrinsic noise of data.

[1] Following Pattern Recognition and Machine Learning by C. M. Bishop

- dataset D -second term:

$$(y(x; D) - h(x))^2$$

$$\begin{aligned}
& (y(x; D) - h(x))^2 = \\
& (y(x; D) - \mathbb{E}_D[y(x; D)] + \mathbb{E}_D[y(x; D)] - h(x))^2 = \\
& (y(x; D) - \mathbb{E}_D[y(x; D)])^2 + (\mathbb{E}_D[y(x; D)] - h(x))^2 + \\
+ \quad & 2(y(x; D) - \mathbb{E}_D[y(x; D)])(\mathbb{E}_D[y(x; D)] - h(x))
\end{aligned}$$

Bias-Variance trade-off

$$\mathbb{E}_D[(y(x;D) - h(x))^2] = \underbrace{\{\mathbb{E}_D[y(x;D)] - h(x)\}^2}_{\text{bias}^2} + \underbrace{\mathbb{E}_D\left[(y(x;D) - \mathbb{E}_D[y(x;D)])^2\right]}_{\text{variance}}$$

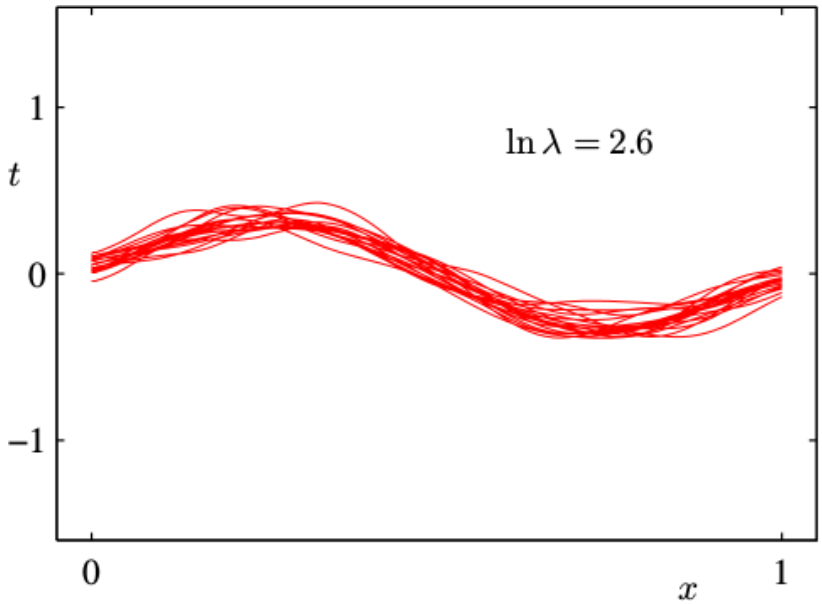
$$\text{expected loss} = \text{bias}^2 + \text{variance} + \text{noise}$$

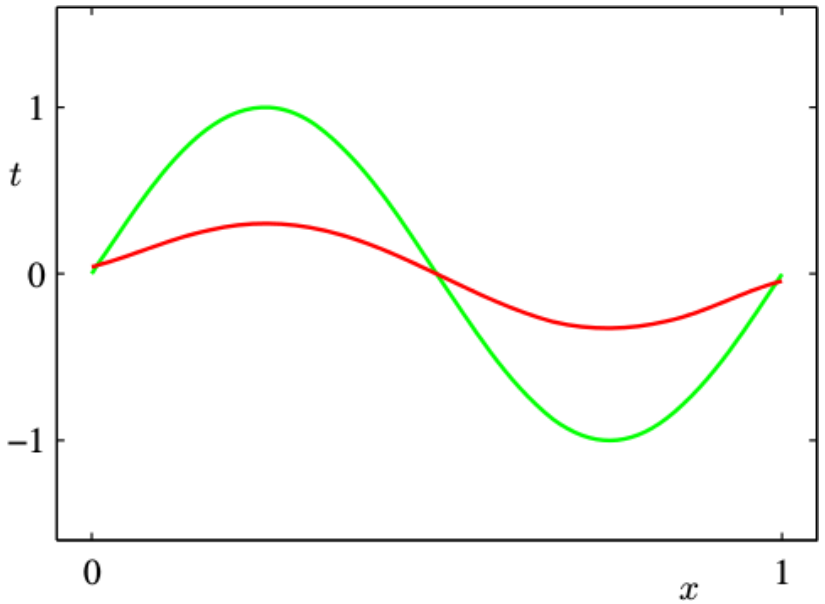
where:

- $\text{bias}^2 = \int \{\mathbb{E}_D[y(x;D)] - h(x)\}^2 \, p(x)dx$
- $\text{variance} = \int \mathbb{E}_D\left[(y(x;D) - \mathbb{E}_D[y(x;D)])^2\right] p(x)dx$
- $\text{noise} = \iint (h(x) - t)^2 p(x,t)dxdt$

Bias-Variance trade-off

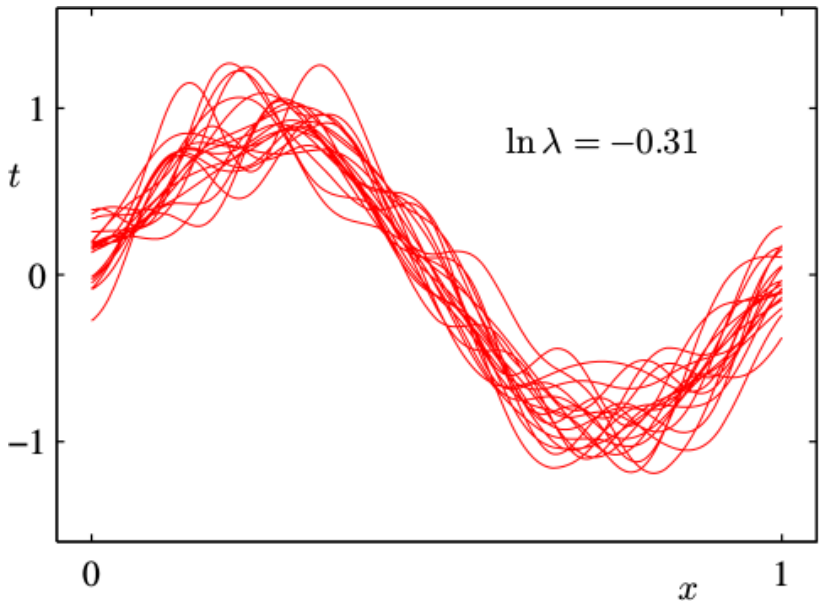
High regularization.

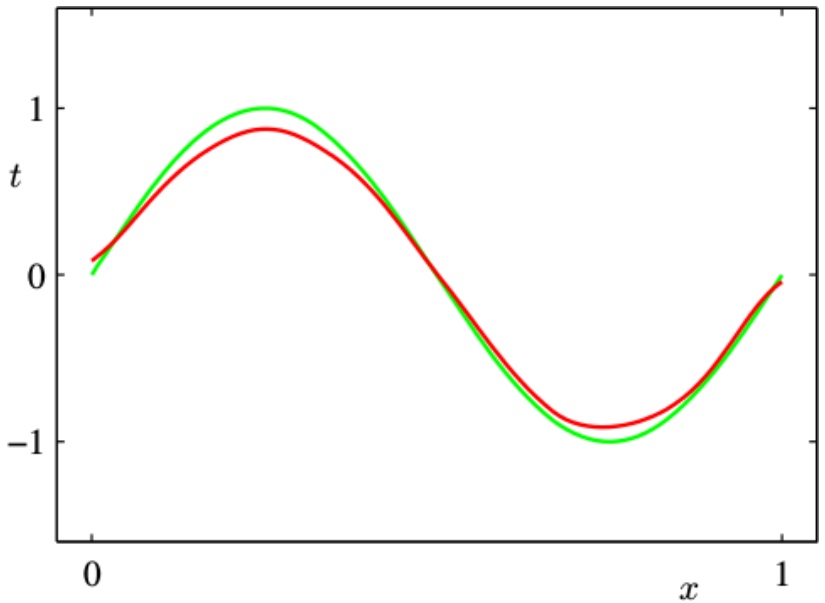




Bias-Variance trade-off

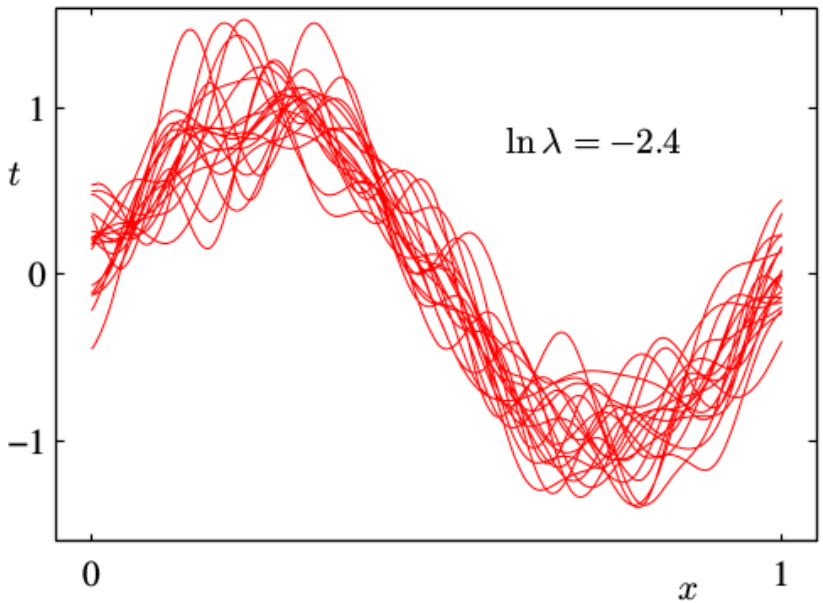
Something optimal.

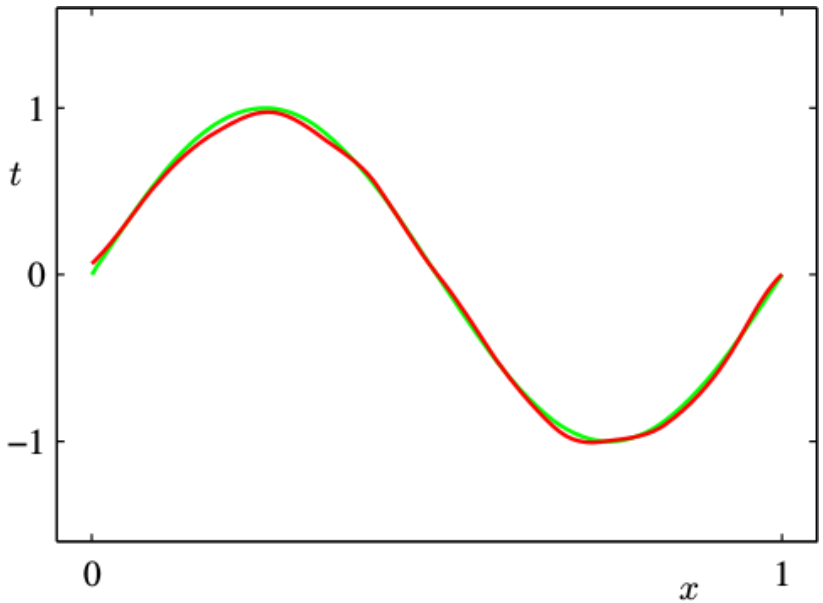




Bias-Variance trade-off

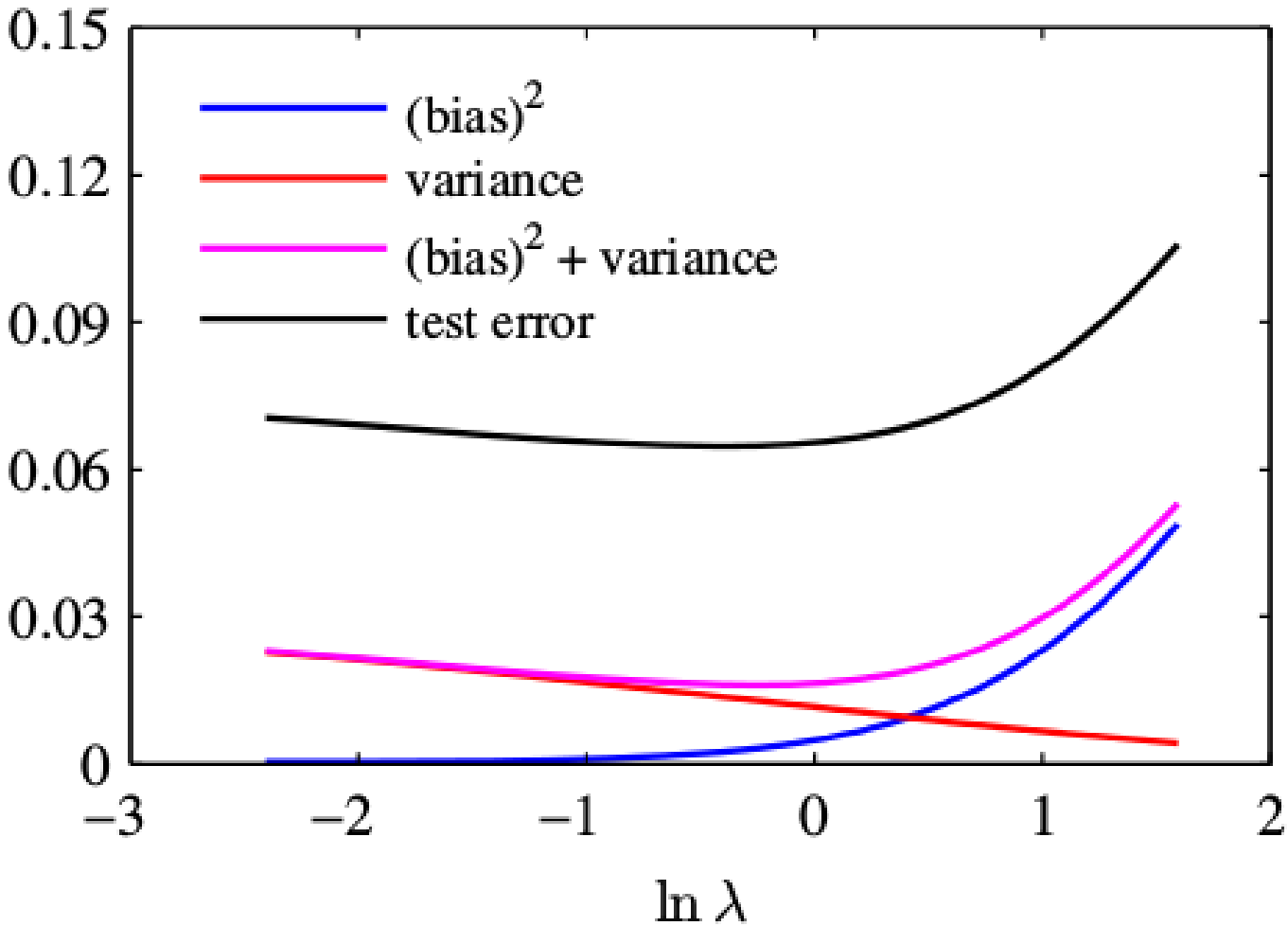
Weak regularization.





Bias-Variance trade-off

Varying regularization parameter.



Bias-Variance trade-off

- High Bias = underfit:
  - too narrow model: few model parameters;
  - your model does not capture all relations between features and target:
    - too strong regularization;
    - shallow trees;
    - not enough units or layers in NN.
- High Variance = overfit:
  - inappropriately wide model: too much model parameters;;
  - your model starts capturing noise as feature-target relation:
    - weak regularization;
    - too deep trees
    - high number of units in NN.

Heuristic

Sometimes a heuristics like:

$$\text{model parameters} = C \cdot \text{size of dataset}$$

works quite well.

- regularization changes model parameters prior distribution, thus allowing to effectively increase C;
- C depends heavily on Loss and model;
- from my experience, to train a well-founded model C should usually be $\sim \frac{1}{10}$.

Hyper-parameters

Hyper-parameters

Usually, a machine learning algorithm actually contains parametrized family of models:

- linear regression + regularization = LASSO, ridge regression;
- Decision Trees can be parametrized in a number of ways:
  - maximal depth;
  - minimal number of samples to make split;
  - minimal number of samples in the node.
- logistic regression has regularization coefficient;
- Neural Networks accept a number of regularization terms.

Train validation test split

To estimate how well an algorithm suits given data you can select them using validation set.

1. training set: to train algorithms;
2. validation set: to estimate performance of chosen algorithm;
3. test set: to estimate performance of chosen algorithm.

Tip

You shold **never** use test dataset to select models!

Cross-validation

Cross validation procedure generalizes train-validation-test split by generating a number of train-validation splits:

- leave-one-out: leaves one sample for validation;
- k-folds: splits dataset into k subsets (folds), trains on $k-1$ folds, validates on the left one;
- Monte-Carlo cross validation: randomly splits into train and validation sets.

Cross-validation

Cross-validation tends to underestimate algorithm performance since uses less samples for training.

Metrics to estimate:

- accuracy, precision, recall:
  - the best procedure is leave-one-out since uses the largest training set;
  - in case of k-folds k should be as large as possible;
- integral metrics (e.g. ROC AUC):
  - quality of metric estimation depends on a number of samples;
  - trade-off between size of training sample and correct estimation of metric.

Tip

For estimation of ROC AUC, precision-recall, etc you can firstly compute predictions, then estimate metric on the entire vector of predicitions. Usually, better metric estimation pays off, especially, if you have low number of samples for same class.

Tip

Practically, you can refit your eastimator on the entire train-validation set. Doing this you may slightly overfit, but increase is usually lower than gain in performance.

Meta-algorithms and no-free-lunch

Train-validation-test split is a machine learning algorithm itself (meta-algorithm). Thus it is affected by no-free-lunch theorem.
You pay in data samples to infer right prior assumptions on given data.

Conclusion

Using a meta-learning algorithms in general you can't achieve the best possible performance.

But in practice they often achieve the best practically possible performance.

Ensembles

Bootstrap aggregating

Bootstraping (or bagging) is a method to stabilize (decrease variance) by averaging models.

- dataset D of size n;
- sample with replacement n' samples to form D_i;
- train model on D_i to obtain y_i;
- the final classifier y_b:

$$y_b(x) = \frac{1}{m} \sum_{i=1}^{m} y_i(x)$$

or

$$y_b(x) = \text{median}\{y_i(x) \mid i = 1 \dots m\}$$

Random forest

Random forest is ensemble of Decision Trees.

Decision Trees tend to overfit. Bagging reduces the variance of model, thus can be used for preventing Decision Tree overfitting.

1. sample with replacement dataset D_i;
2. fit a Decision Tree to D_i;
3. combine predictions using average.

Random Forest

> *Having good features in dataset makes predicitons of Decision Trees to be heavily correlated. For datasets with high feature-samples ration it causes possibly useful features to be unused.*

To decelerate predictions idea similar to bagging is applied to features:

1. from D sample with replacement dataset D_i;
2. from D' sample with replacement features D'_i;
3. fit a Decision Tree to D'_i;
4. combine predictions using average.

Tip

It is usually recommended to sample $\sqrt{k}$ features from dataset with k features.

Tip

Since each tree in Random Forest is independent, training can be done in parallel. Due to this fact, it is often a primarily choice when having large datasets.

Extra Trees

Extra Trees stands for Extremely-Randomized Trees. In contrast to plain Decision Trees, instead of choosing feature by maximization of impurity gain, Extra Trees simply randomly select next split.

Tip

Usign an Extra Tree outside of ensembling methods is usually not a good idea since they have extremely high variance.

Tip

Extra Trees can be succesfully used having a lot of equally bad features.

Extra Trees

Tip

Since Extra Trees selects splits randomly they are computationally more efficient than Decision Trees. The speed up can be up to the number of features (in Random Forest up to $\sqrt{k}$).

Tip

Practically, they usually used to obtained less biased estimate of feature importance.

Stacking

Stacking

The basic idea of stacking is to consider scores of different models as new features.

- a set of base estimators: $f_i(x)$;
- a meta-estimator $G(x)$:

$$G(x) = g(f_1(x), f_2(x), \ldots, f_n(x))$$

- Stacking, in contrast to bagging, decreases bias.
- **Meta-estimator should be trained on independent dataset**.
- For better performance base estimators should be uncorrelated as possible.

Tip

Cross-validation can be used to save samples.

A simple methods are usually used for meta estimator, since:

- usually, the new dataset forms an easier problem;
- usually, number of base estimators (and so the number of features in new dataset) is lower than in the original one.

Calibration

An interesting case of stacking is calibration: restoring posterior probability distribution from classifiers scores.

- usually, calibration estimators use cross-entropy loss:

$$\mathcal{L}(x, y, p) = \frac{1}{n} \sum_{i=1}^{n} (1 - y) \log(1 - p(x)) + y \log p(x)$$

- solution

$$p(x \mid y = 1) = \arg\min_{p} : \mathcal{L}(x, y, p)$$

preserves the most information about the true distribution.

Calibration

Methods that have loss for which posterior distribution does not deliver solution require calibration.

Tip

Usually, methods like SVM, or Decision Tree based require calibration.

Tip

Usual choice of 'calibrator' is logistic regression.