Lecture 8

Hyperparameter optimization and model selection

Machine Learning

Andrey Filchenkov

05.07.2019

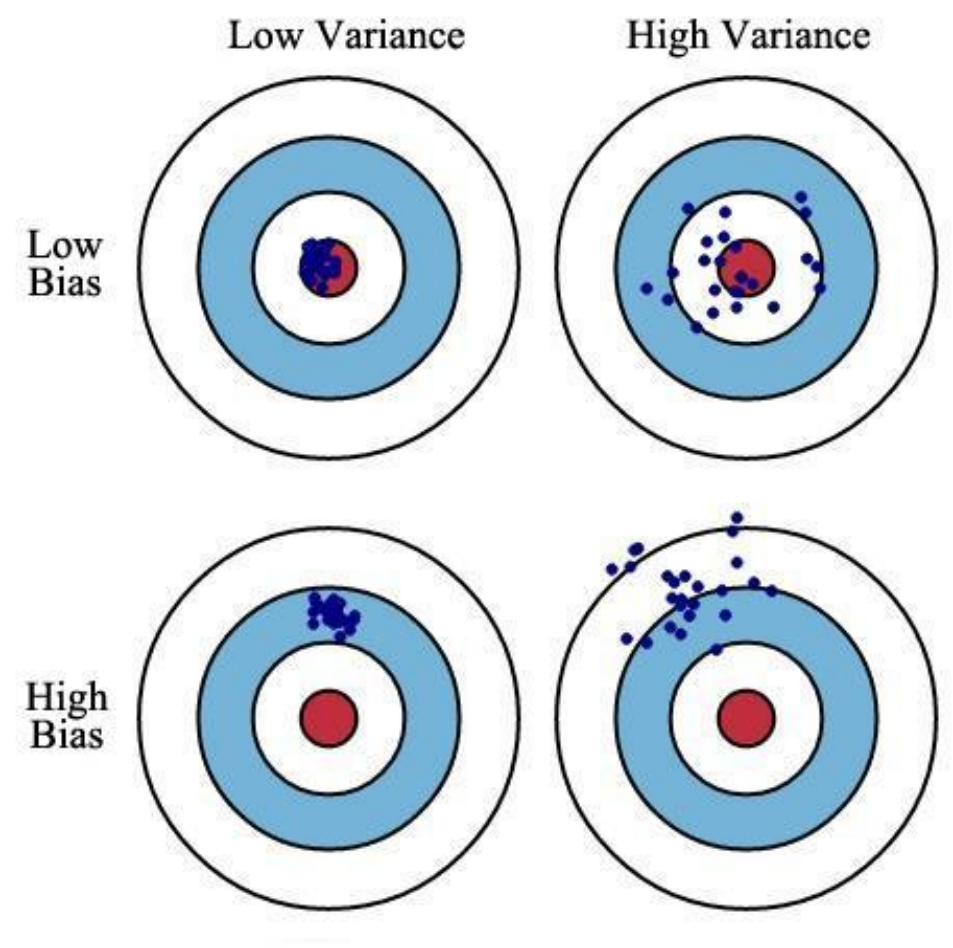
Lecture plan

- Bias vs variance
- Learning curve
- Hyperparameter optimization

Lecture plan

- Bias vs variance
- Learning curve
- Hyperparameter optimization

Bias vs Variance



High bias

What to do?

- Get more features
- Decrease penalty coefficient
- Build more complex model

High variance

What to do?

- Select features
- Get more data
- Increase penalty coefficient
- Build less complex models

Lecture plan

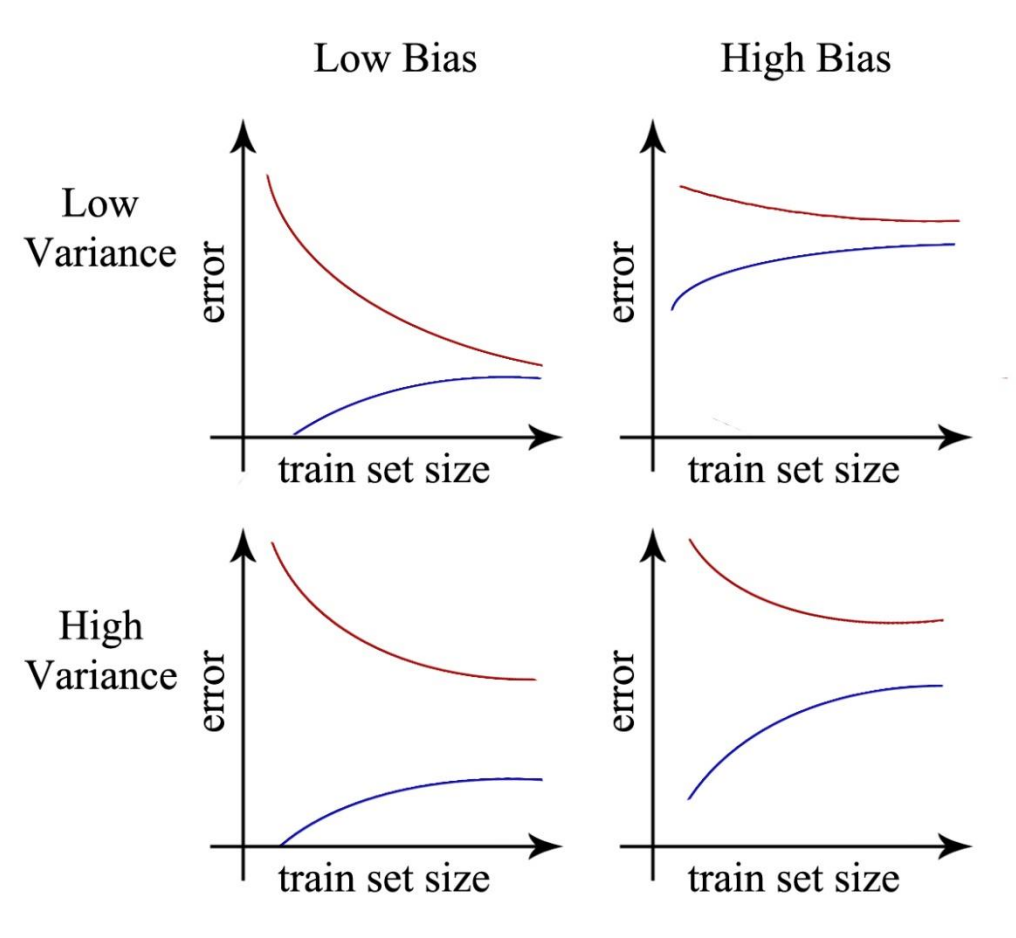
- Bias vs variance
- Learning curve
- Hyperparameter optimization

Learning Curves

A learning curve is a graphical representation of the increase of learning (vertical axis) with experience (horizontal axis).

A learning curve is often useful to plot for algorithmic sanity checking or improving performance.

Learning Curve analysis



Learning Curves

What if you have high bias?

Q_{train}

- Training error is small at first and grows
- Training error becomes close to cross validation
- So the performance of the cross validation and training set end up being similar (but very poor)

Q_{train}

- Straight line fit is similar for a few vs. a lot of data
- So it doesn't generalize any better with lots of data because the function just doesn't fit the data

Learning Curves

What if you have high bias?

If a learning algorithm as high bias as we get more examples the cross validation error doesn't decrease. **So if an algorithm is already suffering from high bias, more data does not help!**

- So knowing if you're suffering from high bias is good!
- In other words, high bias is a problem with the underlying way you're modeling your data
  - So more data won't improve that model
  - It's too simplistic

Learning Curves

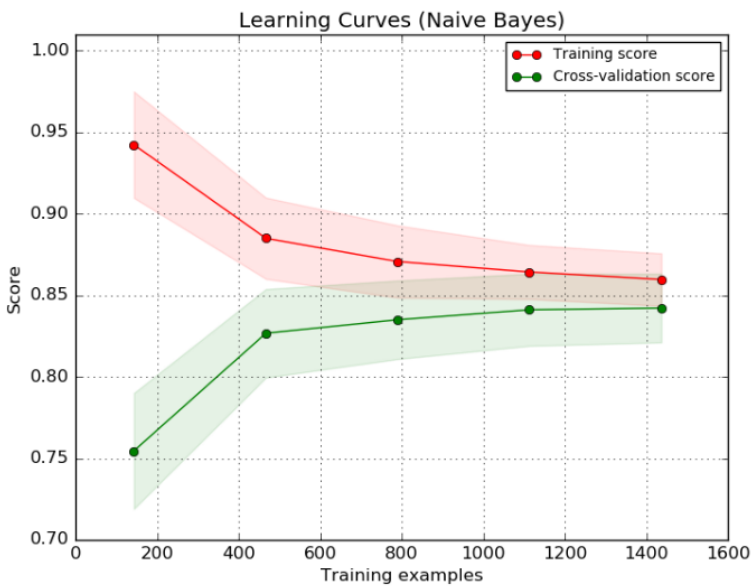
What if you have high variance?

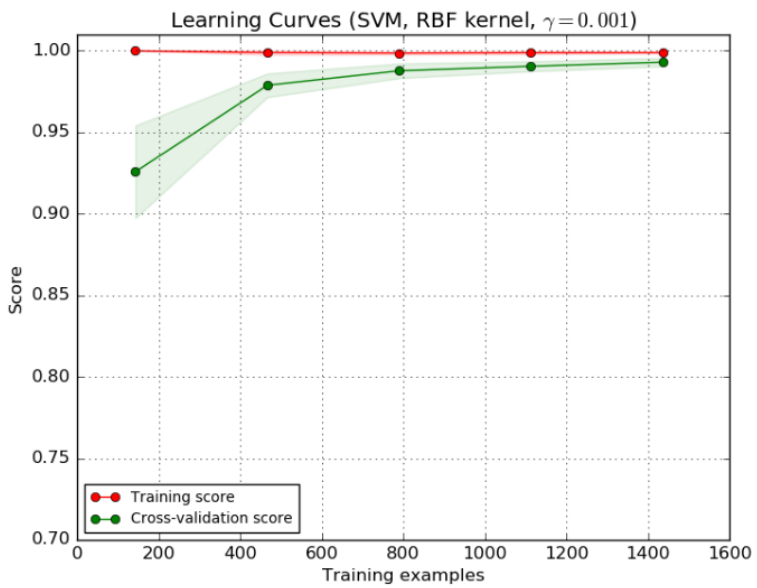
- Q_{train}
  - When set is small, training error is small too
  - As training set sizes increases, value is still small
  - But slowly increases (in a near linear fashion)
  - Error is still low
- Qtrain
  - Error remains high, even when you have a moderate number of examples
  - Because the problem with high variance (overfitting) is your model doesn't generalize

Learning Curves (Python)

Python package **sklearn.learning_curve**

```
train_sizes, train_scores, test_scores = learning_curve(
estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
```





Learning Curves (Python)

Python package **sklearn.learning_curve**

Model parameters:

estimator : object type that implements the "fit" and "predict" methods An object of that type which is cloned for each validation.

title : string Title for the chart.

X : array-like, shape (n_samples, n_features) Training vector, where n_samples is the number of samples and n_features is the number of features.

y : array-like, shape (n_samples) or (n_samples, n_features), optional Target relative to X for classification or regression; None for unsupervised learning.

ylim : tuple, shape (ymin, ymax), optional Defines minimum and maximum yvalues plotted.

cv : integer, cross-validation generator, optional If an integer is passed, it is the number of folds (defaults to 3). Specific cross-validation objects can be passed, see sklearn.cross_validation module for the list of possible objects

n_jobs : integer, optional Number of jobs to run in parallel (default 1).

Lecture plan

- Bias vs variance
- Learning curve
- Hyperparameter optimization

Typical naïve approaches

Hyperparameter optimization or model selection is the problem of choosing a set of hyperparameters for a learning algorithm, usually with the goal of optimizing a measure of the algorithm's performance on an independent data set. Often cross-validation is used to estimate this generalization performance.

Typical naïve approaches

- Use the default parametrization
- Use single parametrization
- Use several parametrization
- Use grid-search

Sequential model-based optimization

Sequential Model-Based Global Optimization (SMBO) optimization for cases where evaluation of the fitness function f is expensive.

Main idea: approximate f with a **surrogate** that is cheaper to evaluate.

It is in treating $R_S(h_\gamma) := f(\gamma)$ as a learnable function of γ, which we can learn from the observations $\{(\gamma_i, R_S(h_{\gamma_i}))\}$ collected during the hyperparameter selection process.

SMBO loop

Typically the inner loop in an SMBO algorithm is the numerical optimization of this surrogate, or some transformation of the surrogate.

The point x that maximizes the surrogate (or its transformation) becomes the proposal for where the true function f should be evaluated.

Surrogate

Usually function family are treated as described with Gaussian processes.

$$p(f(\gamma^*)|R) = N(f(\gamma^*); \mu(\gamma^*; R), \sigma^2(\gamma^*; R),$$

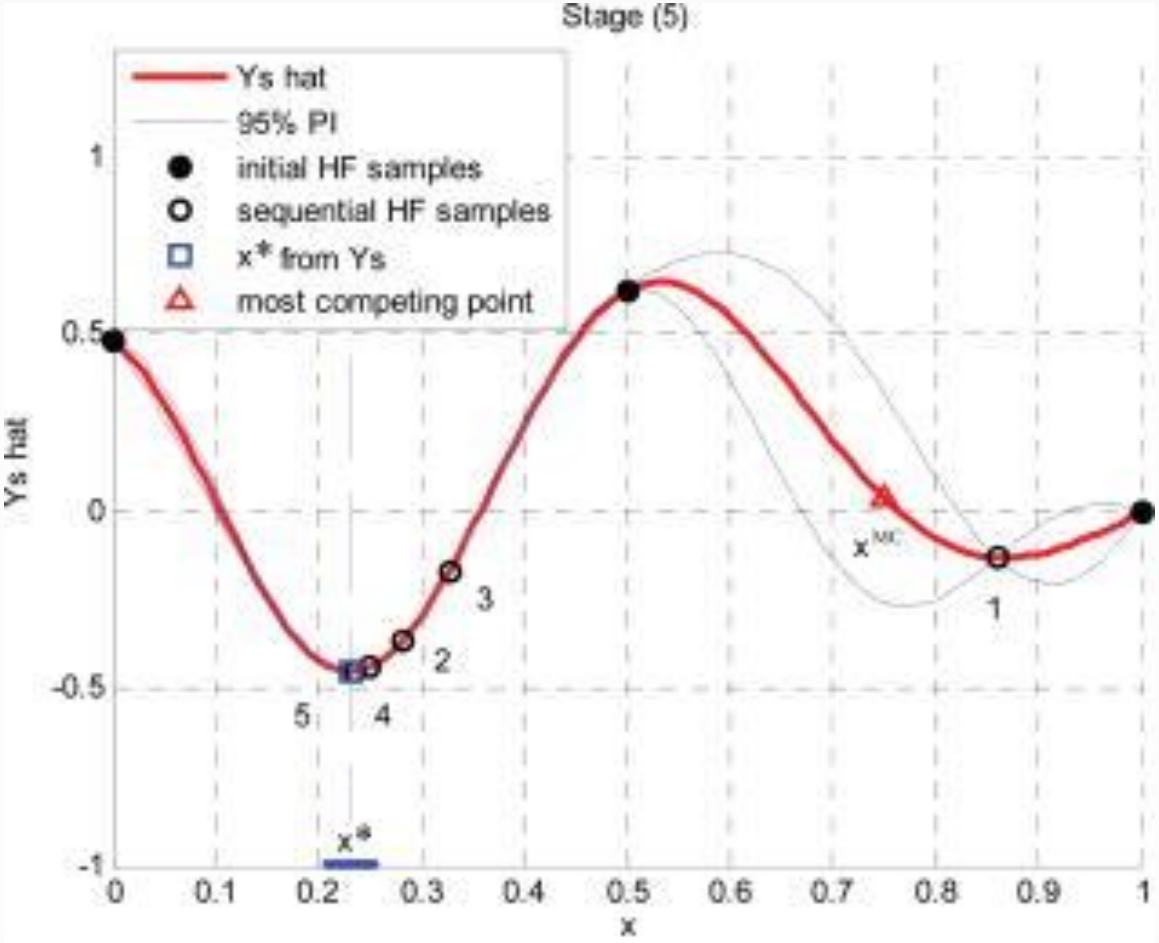
where R is history.

Optimization criteria

Types:

- Expected Improvement
- Probability of Improvement
- Expected Improvement, minimizing the Conditional Entropy of the Minimizer

Example


Stage (5)
Ys hat
95% PI
initial HF samples
sequential HF samples
x* from Ys
most competing point
Ys hat
x
x*
x^MC
1
2
3
4
5

SMBO discussion

Advantages:

- Can find any value of hyperparameters
- Can work for any algorithm and any hyperparameter

Disadvantages:

- Consumes plenty of time
- Depends on initial guess
- Sometimes random walk works better

Implementation

Python: **GridSearchCV**
RandomizedSearchCV
hyperopt

WEKA: **GridSearch**
Auto-WEKA