Data mining and machine learning

dsminingf17vm

Physics MSc course

06 - Model selection, regularization

Pataki Bálint Ármin
ELTE, Physics of Complex Systems Department
2020.10.12.

Estimating accuracy of the model

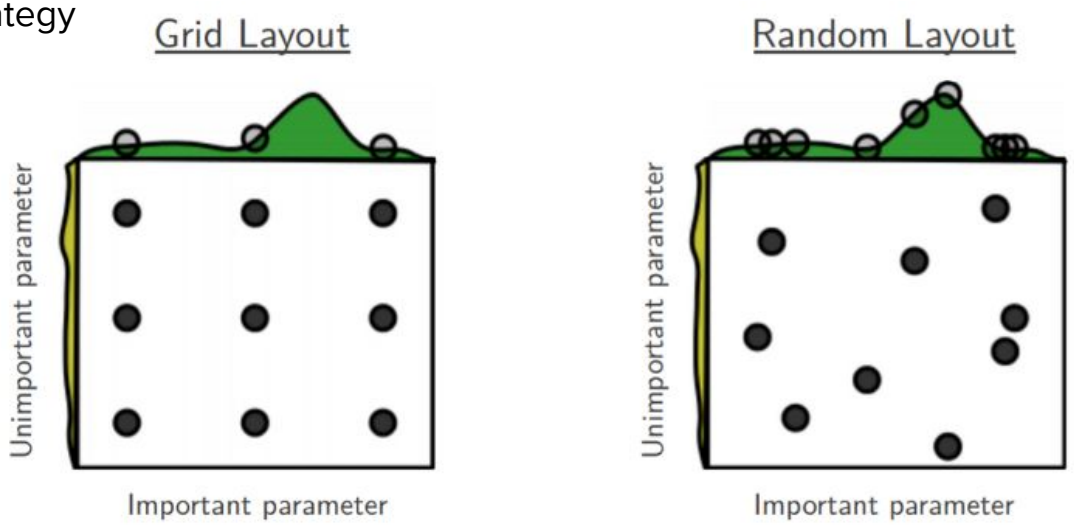
- We train the model with available data & make predictions on unseen examples
- Why do we need to estimate the accuracy of our model?

  (accuracy here can be MAE, MSE, correlation, AUC, etc..)

  - To know what to expect
    - Will it be profitable to trade stocks based on the model?
      - Do not want to rely on training data ➔ a complex model can memorize the whole training set
    - **To select the most accurate model!** ➔ can measure with cross-validation or validation set
      - You can overfit validation set too! ➔ use an unseen test set at the very end
- **Model selection**
  - Type of the model
  - Best combination of input variables or engineered features
  - Tune hyperparameters of the model

Hyperparameter search

- Model **parameters**
  - Learned from the data (eg: w_0, w_1, etc.)
- Model **hyperparameters**
  - Manually set (eg: how many dimensions to use? What degree of polynomial features to use?)
  - Based on intuition, model selection and hyperparameter search
- Hyperparameter search/tuning
  - Try a bunch of hyperparameters ➔ estimate model performance ➔ select the best
  - BUT: computational resource is usually limited
    - Cannot explore infinite options
    - Need some intuition, smart strategy



https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

Validation

Supervised learning is not a usual optimization!

The goal is to build a model that will work in the future during real-world conditions.

- Drives your car
- Detects cancer
- Makes you money on the stock market

You want to estimate the power of your model. How do you do that?

- Performance on the training set?
  - Complex models can memorize the training set ➔ bad idea
- Performance on an unseen dataset?
  - Much better idea, if you have enough data!
  - Try many different model, train all on test data and pick the one that is the best on test?
    - What if one was just lucky? If you have enough model one will be!
    - Would you accept the lottery winners' advice on how to play lottery?

Validation strategies I: train-validation-test

- Put test set away at the beginning
  - Test set should be as representative as possible for the future use

1. Train models on the train dataset
2. Evaluate the model on the validation dataset ➔ tweak your model and goto 1
3. Test your model only a very-very limited times on the test set

How to perform Train / Validation / Test split?

- No duplications over sets (clones, same patient in two different sets)
  - You must know your data to correctly split it! The test split should mimic your future data!
- Test set and validation sets should be large enough to have a stable score
- Train set should be as large as possible

Last two conflicts if you do not have too much data. Then other strategy is needed.

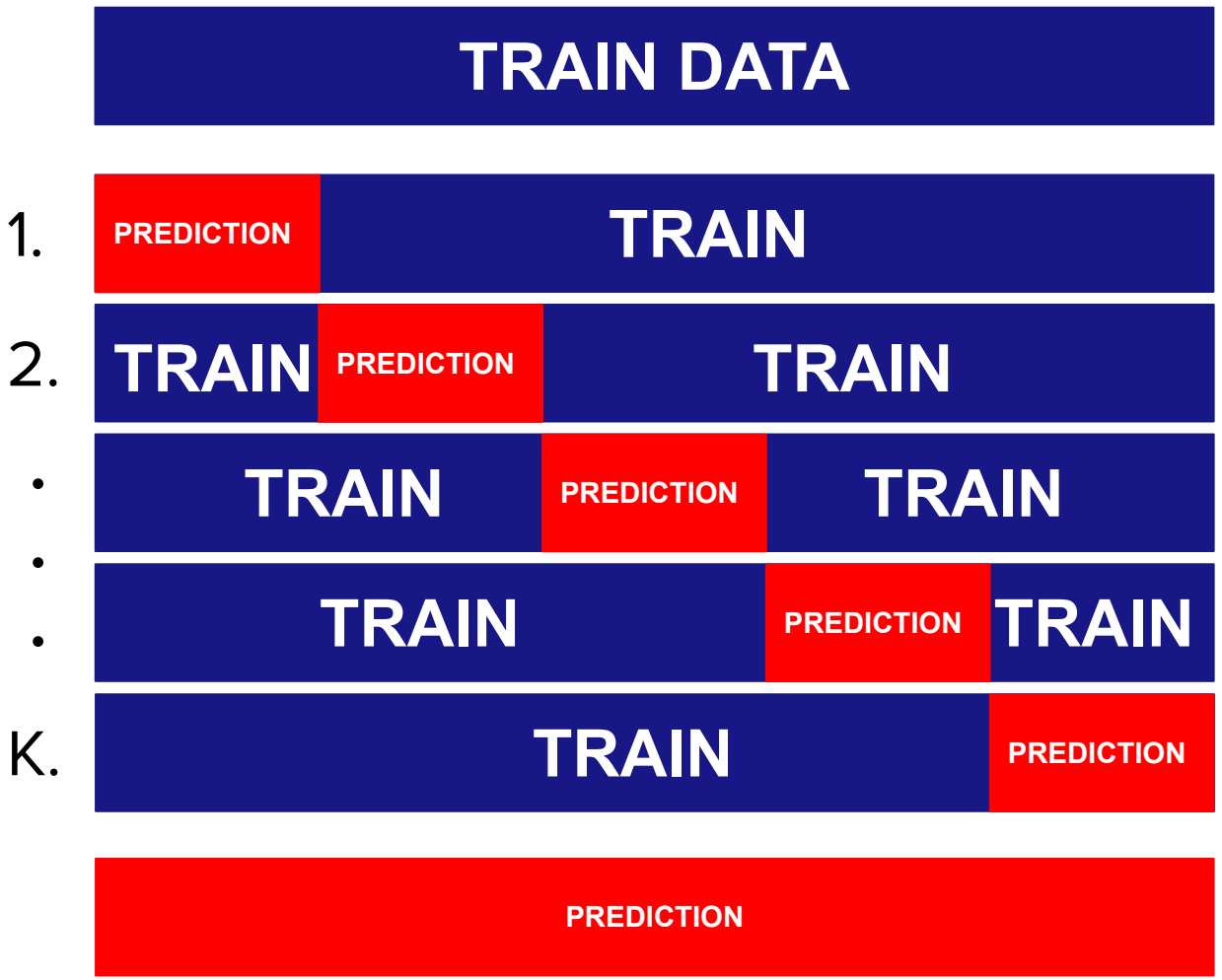
Usual split can be 70/15/15%. But when have a lot of data even 90/5/5% can work.

Validation strategies II: K-fold cross-validation

Train your model K times

- Fit the model on (K-1)/K data
- Make prediction on 1/K data
- End of day:
  - Prediction for whole dataset

    Without seeing in train time
- Tweak model accordingly
- Separate test set is needed
  - For final evaluation
- Shuffling the data?
  - Based on your data
- K is usually 3, 5, 10

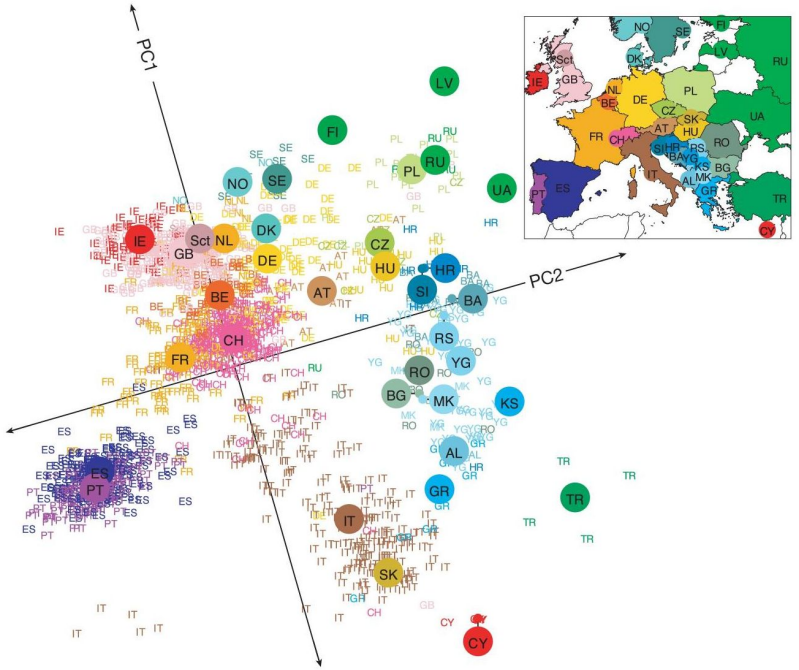


Special K-fold, when K = len(data): leave-one-out validation

Train-test split possible errors - a few example

1. Detect lung cancer based on X-ray images
   - Random train/test split
   - Model may learn person of origin
     - Broken bones / screws in bones
     - Just memorize that “in the training set broken collarbone meant lung cancer”
       - In the test set worked well, as there was a repeated control scan

2. Predict life expectancy from genetics
   - Random train/test split
   - Model may learn country of origin
     - GDP, smoking habits in the country etc.
     - Not the real genetics
     - May work awful for another country/continent

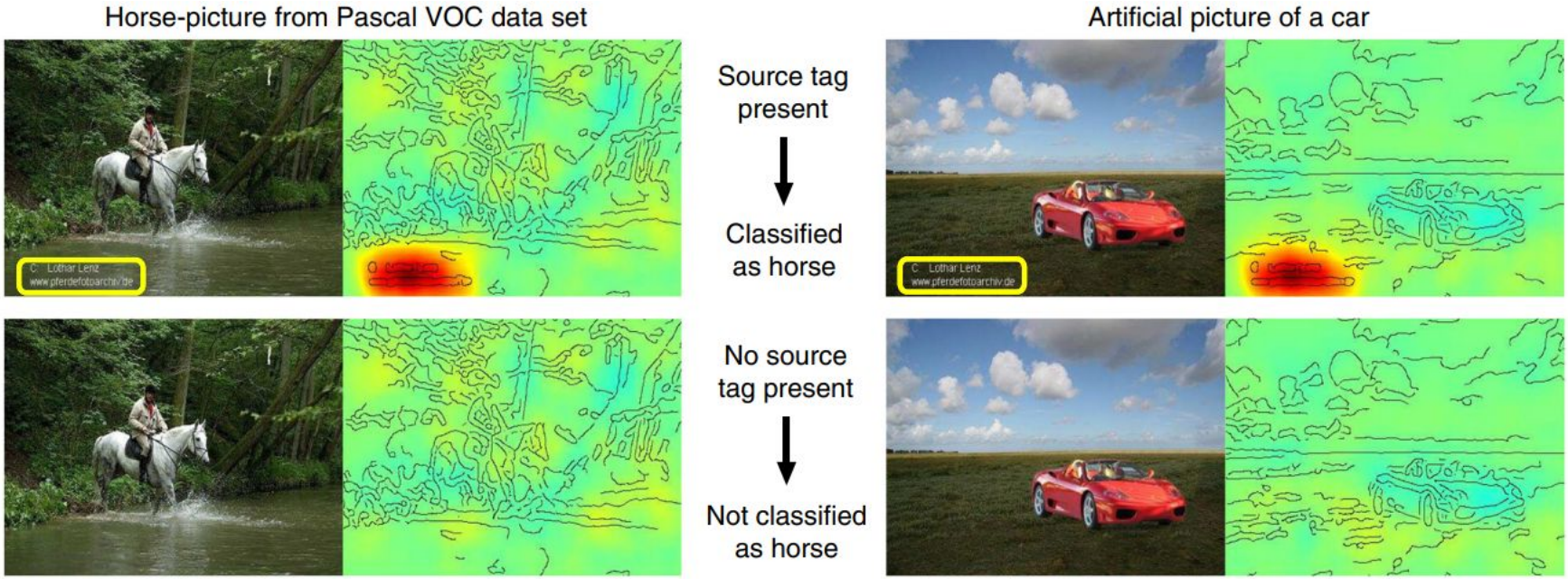


Novembre et al, Genes mirror geography within Europe, Nature, 2008

Train-test split possible errors - a few example

3. Clever hans

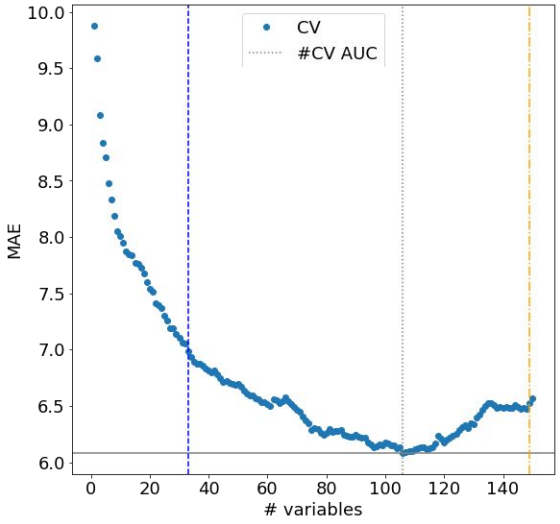
- Train data: horse images had copyright tag (professional photographer)
- Test data: the same
  - Work well in the test data! Will it work in real life usage? Ehh..
- Not exactly split example ➜ artifact in test data collection



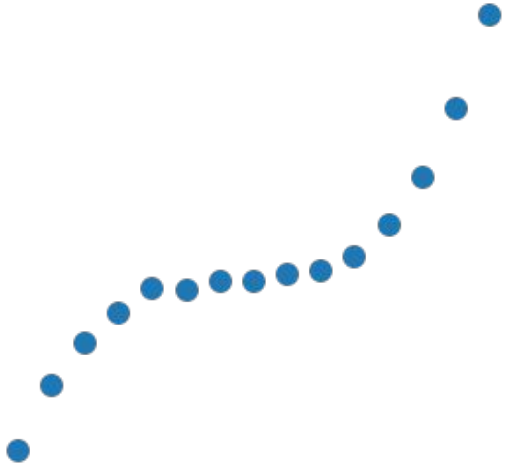
Lapuschkin et al, 2019: Unmasking **Clever Hans** predictors and assessing what machines really learn

Subset selection

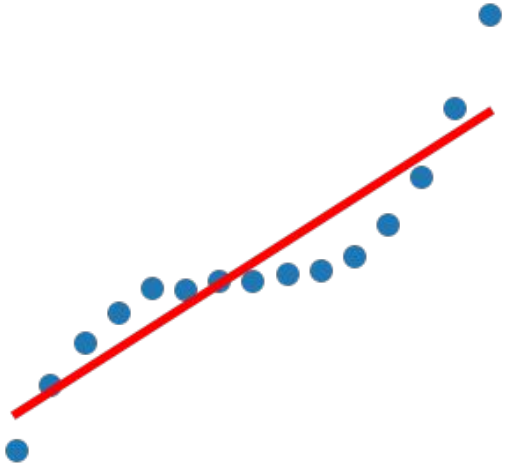
- Modify linear regression with least square fit
  - Why? Least square is the most accurate on training data, but we want high accuracy on test data
  - Least square fit uses all predictors ➜ hard to interpret & what if many variables are just noise?
  - If there are more variables than samples ➜ no unique solution
- Fewer variables
  - Simpler model, easier to interpret
  - How to restrict number of variables (let's say we have **p** variables)
  - $\mathbf{2^p}$ possibility ➜ brute-force hopeless to try all (if p is large)
  - Fast approach 1: forward stepwise greedy selection
    - Start with 0 variables, iteratively add the one that improves the performance the most
  - Fast approach 2: backward
    - Start with all the variables and remove the one that hurts the fit the least



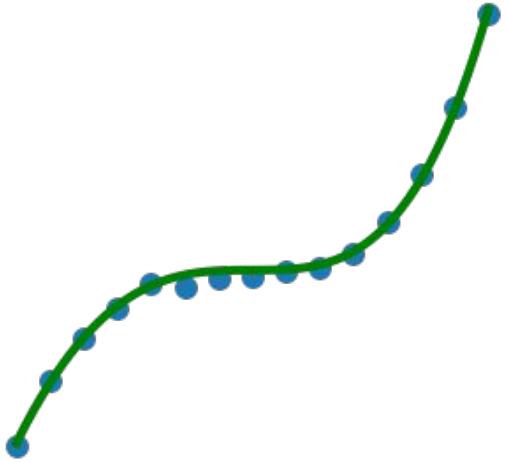
Model complexity: overfitting/underfitting



data points

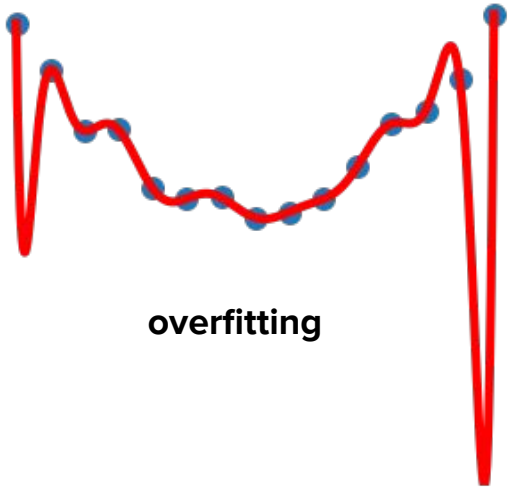


underfitting

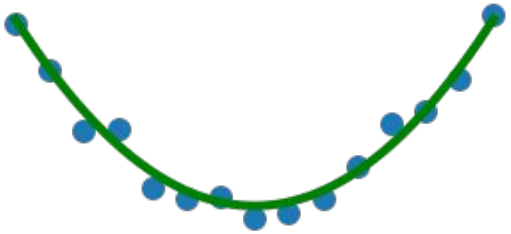


just right

data points

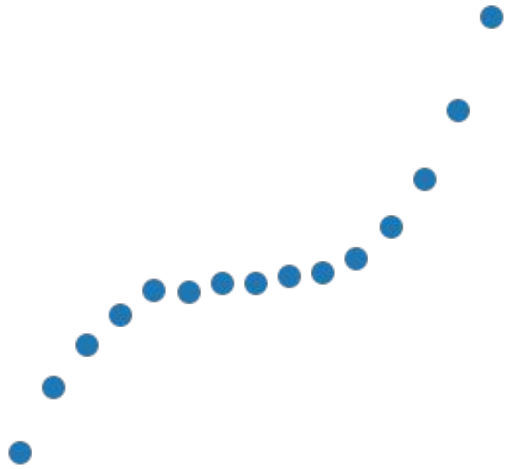


overfitting

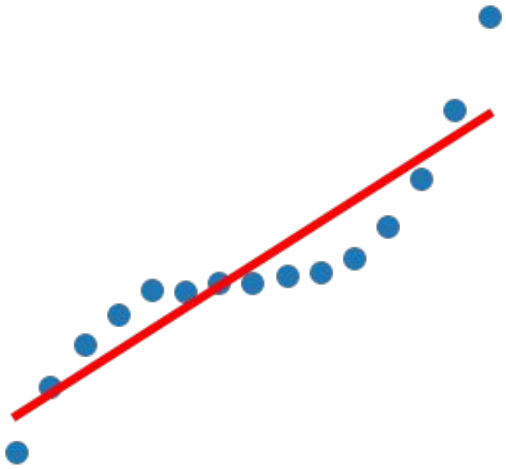


just right

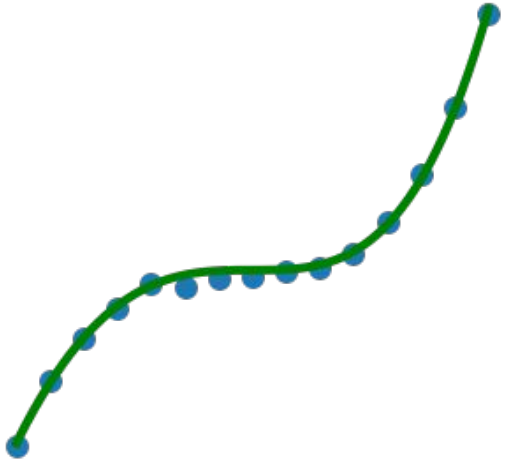
Model complexity: overfitting/underfitting



data points



underfitting

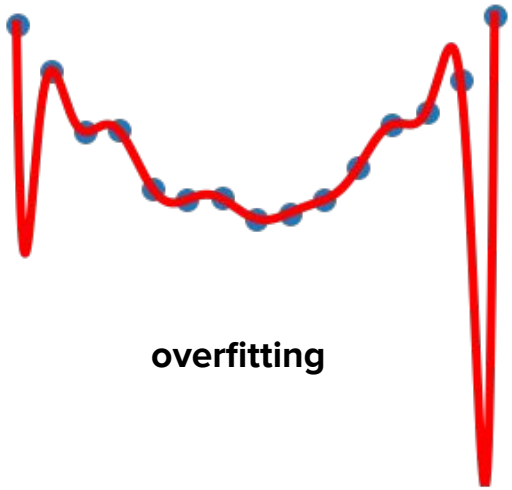


just right

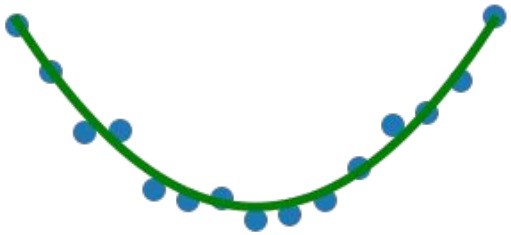
- Too simple model will not solve too complex problems
- Fitting a linear curve on these data points cannot solve out problem
    - The linear curve is fitted right, that is the best a linear model can give
    - A more complex model is needed

Model complexity: overfitting/underfitting

data points



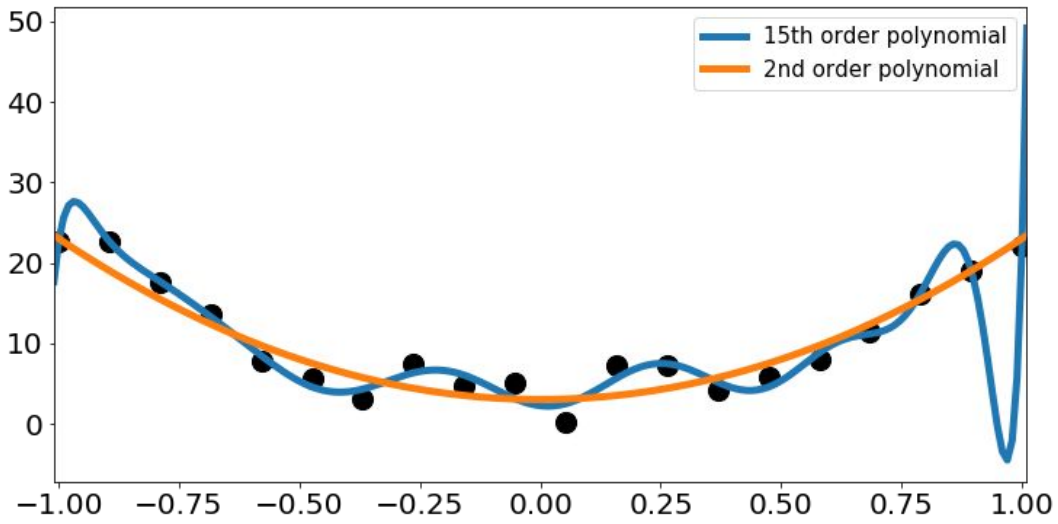
overfitting



just right

- Too simple model will not solve too complex problems
- Fitting a 10th order polynomial curve on these data points cannot solve out problem
    - Work perfectly on data that was used for fitting
    - It will work terribly on new data points
    - But it is possible that a complex model is needed... **what to do then?**

Regularization



How does the coefficients look like?

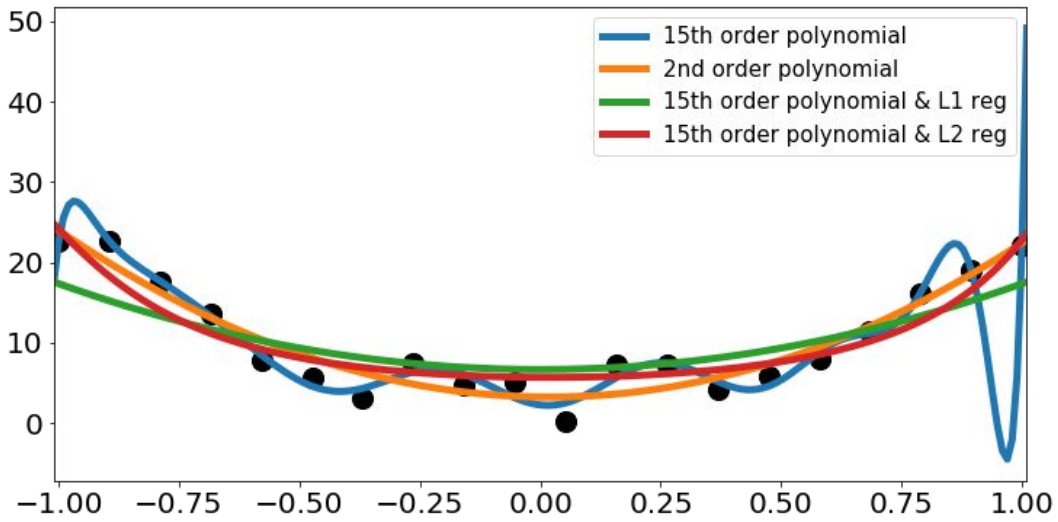
- 2nd order:
  - 19.97 ➔ 2nd order
  - -0.77 ➔ 1st order
  - 3.21 ➔ constant
- 15nd order:
  - 21656.60 ➔ 15th order
  - 18092.89 ➔ 14th order
  - -71081.85 ➔ 13th order
  - -60655.60 ➔ 12th order
  - 92107.45 ➔ 11th order
  - ...

Regularization

Why don't we penalize coefficient explosion?

- Afterall, coefficient explosion is the guilty for that wavy curve!
- Loss function is mean squared error (MSE) $\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$
  - Add a punishment term to the loss! Alpha, beta ➔ hyperparameters
    - L1 regularization $\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 + \alpha\sum_{i}|w_i|$
    - L2 regularization $\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 + \alpha\sum_{i}|w_i|^2$
    - Mixed L1 & L2 $\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 + \alpha\sum_{i}|w_i| + \beta\sum_{i}|w_i|^2$

Regularization: L1 vs L2

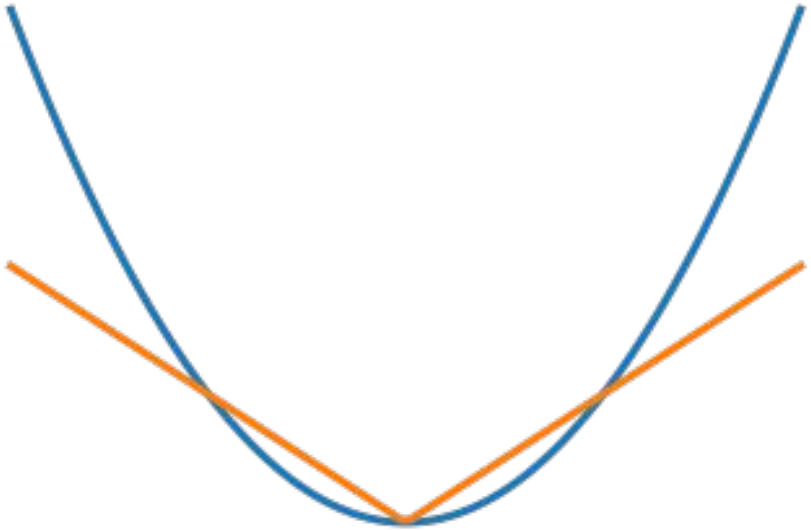


How does the coefficients look like?

- 2nd order:
  - 19.97 → 2nd order
  - -0.77 → 1st order
  - 3.21 → constant
- 15th order & L2:
  - 0.39 → 15th order
  - -0.52 → 14th order
  - ...
  - 3.26 → 6th order
  - -0.43 → 5th order
  - 5.23 → 4th order
  - -0.675 → 3rd order
  - 7.26 → 2nd order
- 15th order & L1:
  - 0 → 15th order
  - 0 → 14th order
  - ...
  - 0 → 4th order
  - 0 → 3rd order
  - 10.69 → 2nd order
  - 0 → 1st order
  - 0 → constant

Regularization: L1 vs L2

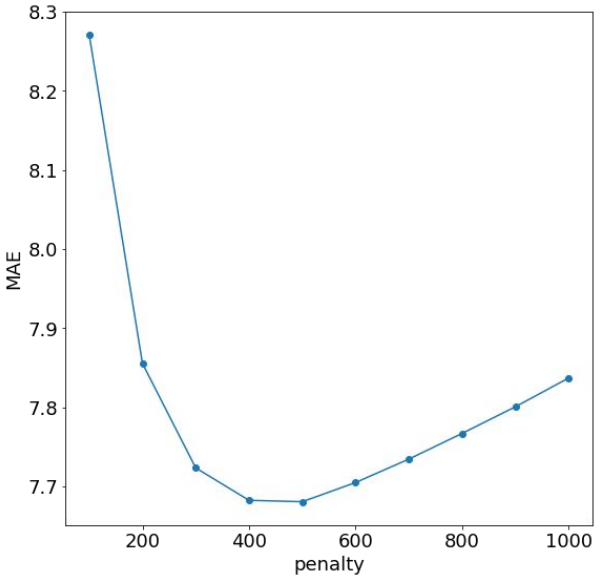
- Linear regression + L1 regularization
  - Lasso regression
  - Usually sparse coefficients ➔ fewer parameters, easier to interpret the model
  - Slope is 1 around 0
- Linear regression + L2 regularization
  - Ridge regression
  - Usually non-sparse coefficients
  - Slope is << 1 around 0
- Linear regression + L1 + L2 regularization
  - Elastic Net regression

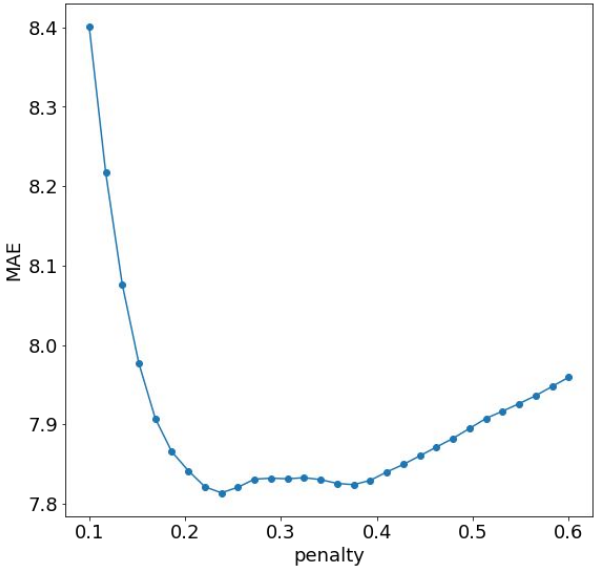


They usually increase train error ➔ less overfitting ➔ test error can improve

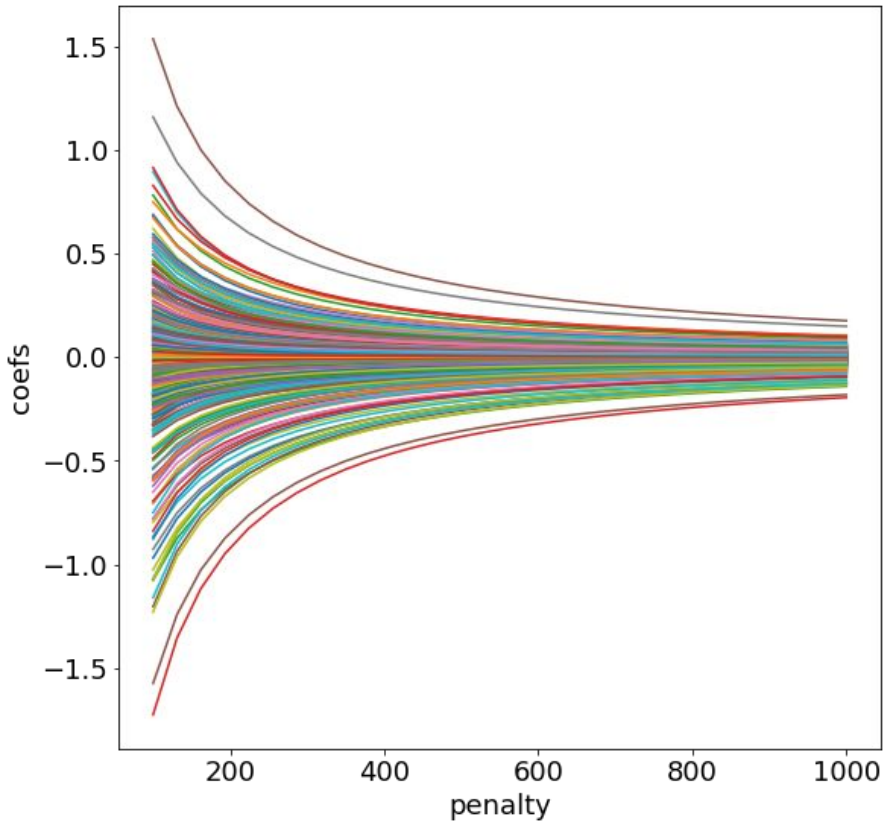
Regularization hyperparameter search

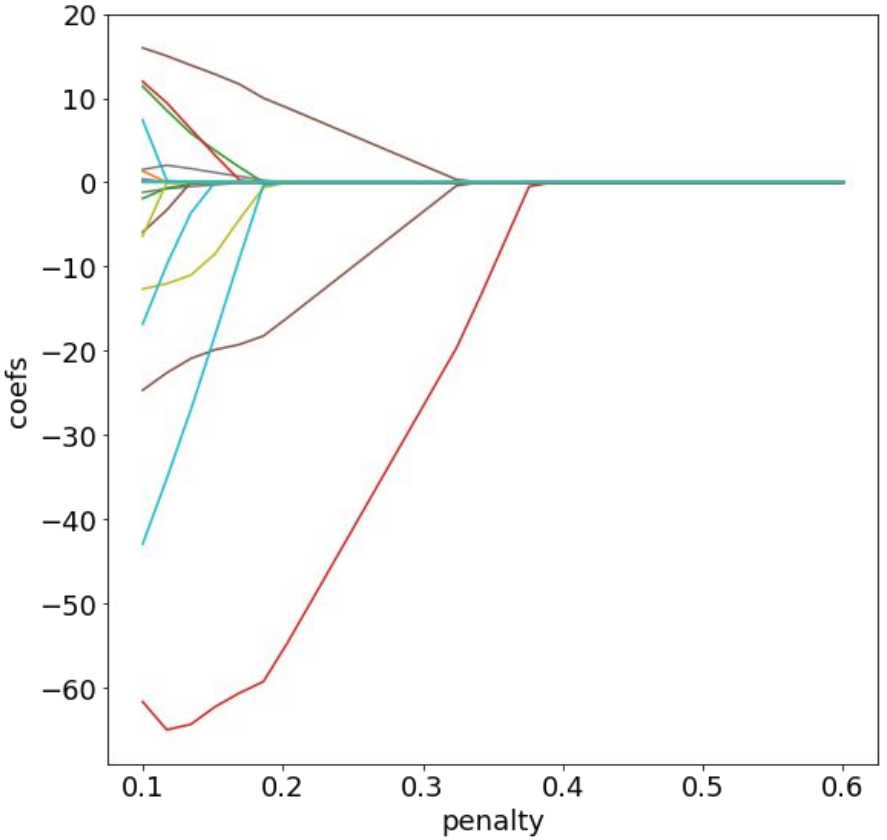
- Scan penalty parameter
  - Plot cross-validation/validation error
    - MAE/MSE/Accuracy etc..
- Scan the meaningful range
  - Quick scan with a few points
  - Log-scale scan
  - More detailed scan around the minima
  - Usually it is not monotonic!





Shrinkage: L1 vs L2. Which one is which?





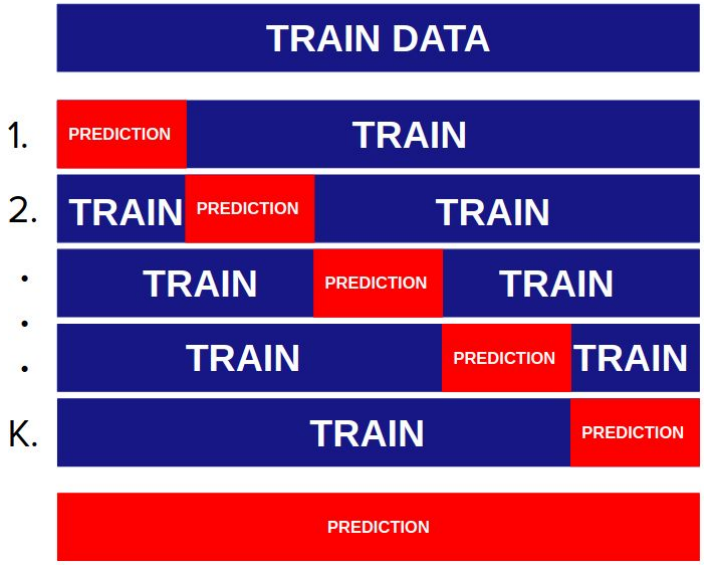
For other kind of models other kind of regularization might be available.
Goal & concept is similar.

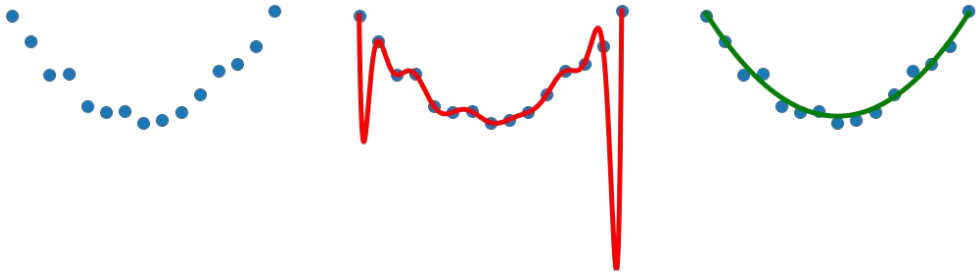
Which model to use?

- There is no universal, “superior model”
  - It worth check winning models on Kaggle (winning is often an ensemble of dozens of models)
    - https://www.kaggle.com/sudalairajkumar/winning-solutions-of-kaggle-competitions
  - Feature engineering can be the ‘joker’ ➔ generate new features based on the data exploration
  - Tabular data ➔ random forest, gradient boosting decision trees are really strong
    - But if the connection is linear ➔ linear models can outperform
  - Images, sound ➔ convolutional neural networks are hard to beat
- Each model has some advantage / disadvantage
  - Easier to interpret (eg linear regression)
  - Robust (random forest / xgboost)
  - Can extrapolate?
- Correct validation is a must
- So as exploring & understanding the data

Summary

- Validation
  - How to estimate model performance?
  - Train / validation / test split
  - Cross-validation
- Regularization
  - Reduce overfitting ➜ improve generalization
  - L1/L2 for linear regression
  - General idea can be applied to many models
  - Shrinkage





Reference

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani: An Introduction to Statistical Learning, Chapter 6 http://faculty.marshall.usc.edu/gareth-james/ISL/

Assignment 6

1. Implement a linear model

- return the weight parameters w = (w1, w2, ... , wP) and the intercept parameter w0 separately where:

$$\hat{y}(\vec{w}, \vec{x}) = w_0 + w_1 x_1 + \ldots + w_p x_p$$

- check your returned coefficients with the built in `LinearRegression` class from the `sklearn` library, they should be within tolerance `1e-6` to each other
- use a generated regression dataset `from sklearn.dataset import make_regression` API with parameters `n_samples=1000` and `n_features=20`

2. Use of real data

- download the Communities and Crime Data Set from UCI, the task includes understanding the dataset: naming the appropiate data fields, handling missing values, etc.
- fit a `LinearRegression` model with 5-fold cross-validation - compare training and testing scores (R^2 by default) for the different CV splits, print the mean score and its standard deviation
- find the best `Lasso` regression model with 5-fold grid search cross validation (`GridSearchCV`) on the parameters: `alpha, normalize, max_iter` and show the best parameter set

3. Shrinkage

- interpret Lasso model's coefficients based on its descriptive parameters by the shrinkage method described during the lecture (make a plot and check the names of the features that are not eliminated by the penalty parameter) on the data we have here (this is an explanatory data analysis problem, be to be creative)
- fit Ridge model and apply the shrinkage method as well, did you get what you expect?
- do you think normalization is needed here? If so, do not forget!

4. Subset selection

- Split the data to a training and test set and do recursice feature elimination until 10 remaining predictors with 5-fold cross-validated regressors (`RidgeCV` , `LassoCV` , `ElasticNetCV`) on the training set, plot their names and look up some of their meanings (recursive feature elimination is part of `sklearn` but you can do it with a for loop if you whish).
- Do all models provide the same descriptors? Check their performance on the test set! Plot all model predictions compared to the `y_test` on 3 different plots, which model seems to be the best?

5. ElasticNet penalty surface

- visualize the surface of the $objective(\alpha, \beta)$ parameters corresponding to the L1 and L2 regularizations. Select the best possible combination of the hyper-parameters that minimize the objective (clue: `from scipy.optimize import minimize`)
  - this task is similar to what you've seen during class, just not for MSE vs. single penalty parameter but MSE vs. two penalty parameters α, β
- interpret the findings! do you think linear models are powerful enough on this dataset?