CSI 436/536
Introduction to Machine Learning

Model selection for LLSE

Professor Siwei Lyu
Computer Science
University at Albany, State University of New York

Model selection

- Training a model (e.g., linear model using LLSE) from a training data can determine the ***parameters*** in the model
- There are ***model families*** that cannot be determined from data alone, such as
  - The degree of polynomial in polynomial fitting
  - The type of nonlinear models in regression
- ***Model selection*** decides the form of the model and the model parameter to be learned
- ***Model training*** decides the specific value of the model parameter for a model from the model family based on training data

Model selection in LLSE

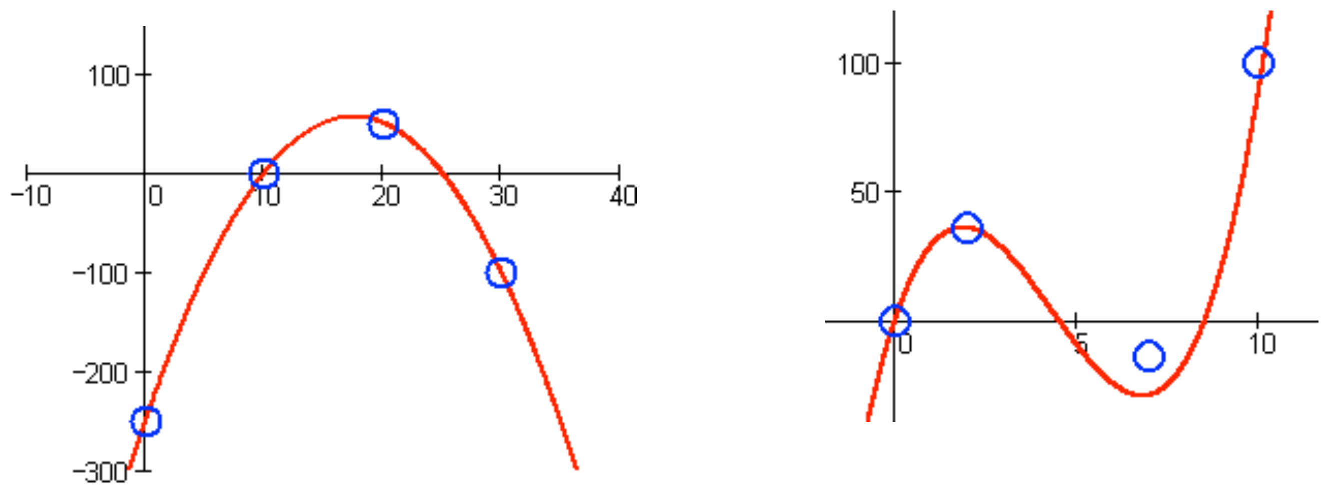
- find d-degree polynomial

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d$$

as

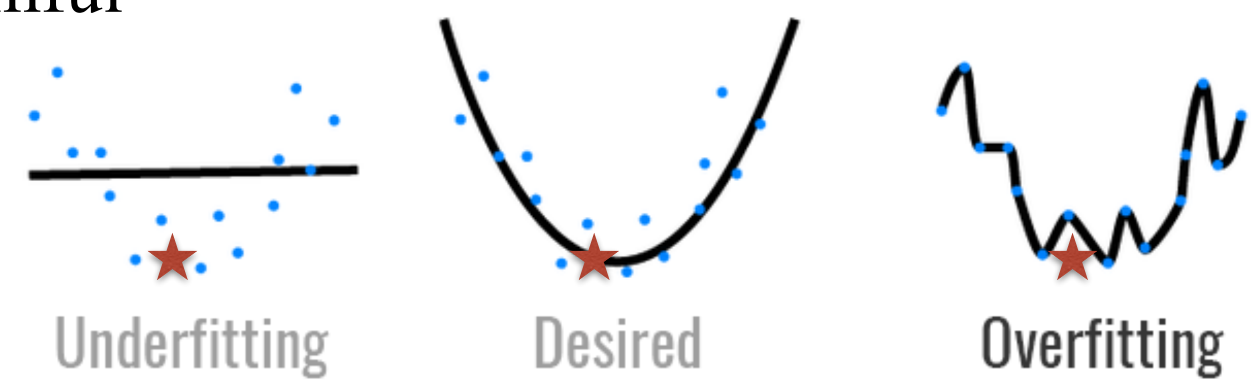
$$\min_{w=(a_0,\cdots,a_d)^T} \sum_{i=1}^{N} (y_i - f(x_i))^2$$

- What is the right d for a particular set of data?
  - This cannot be learned solely from data



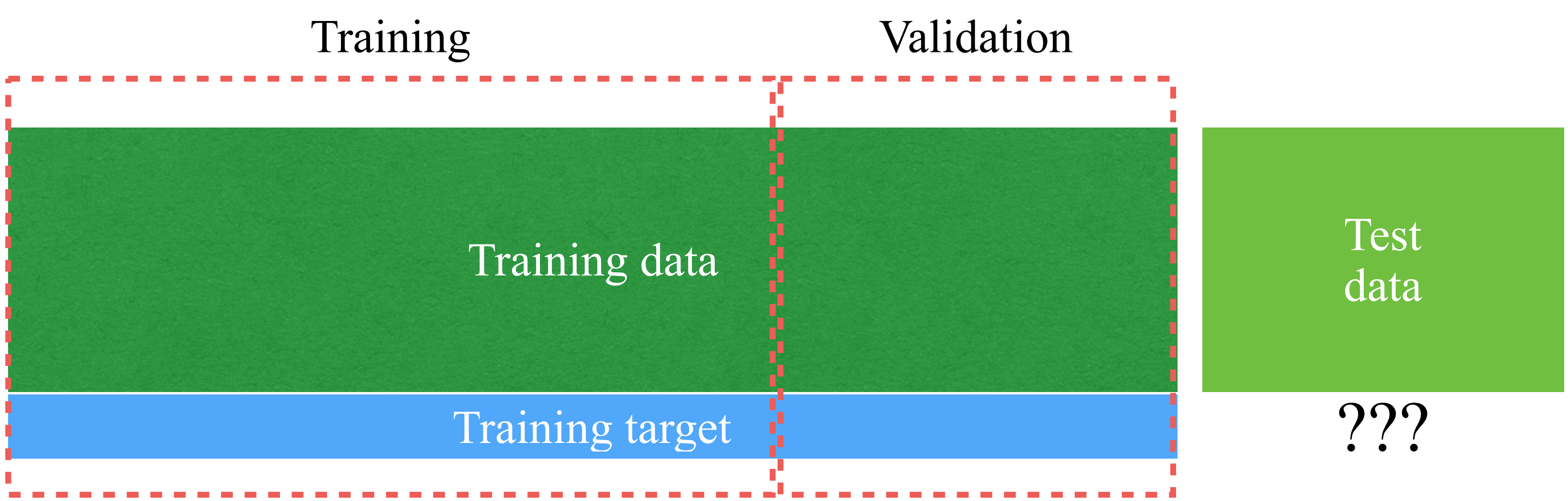
Overfitting and underfitting

- If models are not chosen carefully, overfitting or under-fitting will occur
  - When a model has low error on training data but high error on testing data, it ***overfits.*** When it has high error on training data, it ***underfits***
  - Both are undesirable, but overfitting may be more harmful



Model selection by validation

- Models cannot be chosen based on their performance on the training data
- Performance should be tested on a ***test dataset*** that are not used in training to avoid overfitting
- Model selection is performed on a part of training data that are not used in training — the ***validation dataset***



General procedure of model selection

- Decide a candidate set of model families
  - For LLSE, corresponding to choosing polynomials of different degrees
- For each candidate model family
  - Obtain optimal parameter using the training set
  - Compute the error metric on the validation dataset
- Choose the model family that leads to the minimum error on the validation dataset
- Deploy the best model of the chosen family on the test dataset to report results

Incremental LLSE

- Idea: fitting data with polynomials of different degrees,
  - A simple idea is to try for a range of d = 1, …, D, to fit the training data using LLSE of each degree
    - problem: each time we have to solve the normal equation by inverting the correlation matrix, leading to complexity $O(ND^4)$
- Better idea is to do this incrementally, using the result of previous step to bootstrap
- This is known as ***incremental LLSE***, which can be solved similarly as recursive LLSE using dynamic programming and matrix inverse lemma

Incremental LLSE

- Old data matrix X and correlation matrix XX^T
- New data matrix $\tilde{X} = \begin{pmatrix} X \\ \tilde{x}^T \end{pmatrix}$, where the new vector is N x 1 corresponding to the evaluation of additional degree monomial
- New correlation matrix

$$\tilde{X}\tilde{X}^T = \begin{pmatrix} X \\ \tilde{x}^T \end{pmatrix} \begin{pmatrix} X^T & \tilde{x} \end{pmatrix} = \begin{pmatrix} XX^T & X\tilde{x} \\ \tilde{x}^T X^T & \tilde{x}^T \tilde{x} \end{pmatrix}$$

- We need to compute $(\tilde{X}\tilde{X}^T)^{-1}$, can we use the result we already have for $(XX^T)^{-1}$?

Block matrix inversion lemma

- Given a block matrix $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$, if D is invertible, define D's Schur complement as $\hat{D} = A - BD^{-1}C$, then we can show that
$$M = \begin{pmatrix} I & BD^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} \hat{D} & 0 \\ 0 & D \end{pmatrix} \begin{pmatrix} I & 0 \\ D^{-1} & I \end{pmatrix}$$
- Then it is easy to show that
$$M^{-1} = \begin{pmatrix} I & 0 \\ -D^{-1}C & I \end{pmatrix} \begin{pmatrix} \hat{D}^{-1} & 0 \\ 0 & D^{-1} \end{pmatrix} \begin{pmatrix} I & -BD^{-1} \\ 0 & I \end{pmatrix}$$
- Special case, when D = d, B = x, C = x^T, we have
$$M^{-1} = \begin{pmatrix} I & 0 \\ -x^T/d & I \end{pmatrix} \begin{pmatrix} (A - xx^T/d)^{-1} & 0 \\ 0 & 1/d \end{pmatrix} \begin{pmatrix} I & -x/d \\ 0 & I \end{pmatrix}$$

Block matrix inversion lemma

- Using matrix inversion lemma,

$$(A - xx^T/d)^{-1} = A^{-1} + \frac{A^{-1}xx^TA^{-1}}{d - x^TA^{-1}x}$$

then

$$M^{-1} = \begin{pmatrix} I & 0 \\ -x^T/d & I \end{pmatrix} \begin{pmatrix} A^{-1} + \frac{A^{-1}xx^TA^{-1}}{d - x^TA^{-1}x} & 0 \\ 0 & 1/d \end{pmatrix} \begin{pmatrix} I & -x/d \\ 0 & I \end{pmatrix}$$

$$= \begin{pmatrix} A^{-1} + \frac{A^{-1}xx^TA^{-1}}{d - x^TA^{-1}x} & 0 \\ \frac{-x^TA^{-1}}{(d - x^TA^{-1}x)} & 1/d \end{pmatrix} \begin{pmatrix} I & -x/d \\ 0 & I \end{pmatrix}$$

$$= \begin{pmatrix} A^{-1} + \frac{A^{-1}xx^TA^{-1}}{d - x^TA^{-1}x} & \frac{-A^{-1}x}{(d - x^TA^{-1}x)} \\ \frac{-x^TA^{-1}}{(d - x^TA^{-1}x)} & \frac{1}{d - x^TA^{-1}x} \end{pmatrix}$$

Incremental LLSE

- Computing

$$\hat{w} = \begin{pmatrix} (XX^T)^{-1} + \frac{(XX^T)^{-1}xx^T(XX^T)^{-1}}{d - x^T(XX^T)^{-1}x} & \frac{-(XX^T)^{-1}x}{(d - x^T(XX^T)^{-1}x)} \\ \frac{-x^T(XX^T)^{-1}}{(d - x^T(XX^T)^{-1}x)} & \frac{1}{d - x^T(XX^T)^{-1}x} \end{pmatrix} \begin{pmatrix} Xy \\ \tilde{x}^T y \end{pmatrix}$$

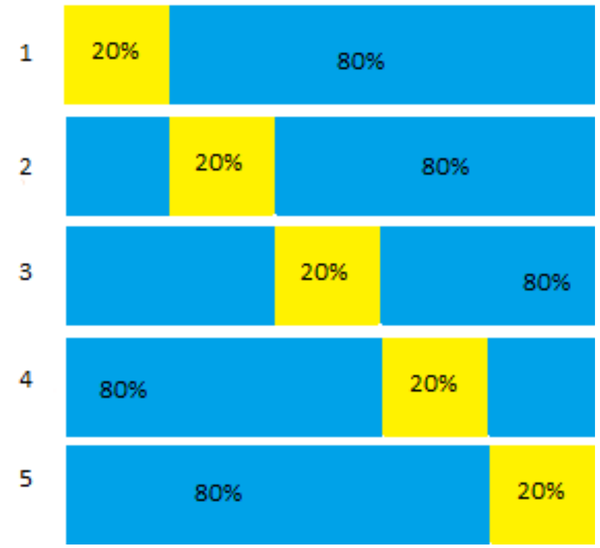
- After simplification we have $\hat{w} = \begin{pmatrix} w + w_0(XX^T)^{-1}X\tilde{x} \\ w_0 \end{pmatrix}$,

  where $w_0 = \dfrac{\tilde{x}^T(X^T w - y)}{\tilde{x}^T\tilde{x} - \tilde{x}^T X^T (XX^T)^{-1} X\tilde{x}}$

  - interpretation: $w_0 = 0$ if $X^T w - y = 0$, i.e., the previous model is enough to get perfect prediction on the data, so no need to add any further new components

Cross-validation

- Cross-validation is used to avoid any potential bias in the training-validation segmentation
- k-fold cross-validation: equally segment training dataset to k parts, then train on any k-1 parts and test the error on the remaining part



5-fold cross-validation

  - For training single model, choose the best model out of the k-fold estimates
  - For model selection, find a model family that gives the smallest average k-fold training losses
  - Special case: k = N, known as the ***leave-one-out*** (LOO) cross-validation
- In the case of LLSE, LOO can be computed in closed form

Matrix inversion lemma

- Woodsbury identity: when A and D are invertible

  $(A + BDC^T)^{-1} = A^{-1} - A^{-1}C(D^{-1} + CA^{-1}B^T)^{-1}B^T A^{-1}$

  - Proof: multiply the matrix on both sides
- important special case
  - B=C=z, a vector, D=I

    $(A + zz^T)^{-1} = A^{-1} - (A^{-1}zz^T A^{-1})/(1 + z^T A^{-1}z)$
  - B=-C=z, a vector, D=I

    $(A - zz^T)^{-1} = A^{-1} + (A^{-1}zz^T A^{-1})/(1 - z^T A^{-1}z)$
  - caching A^{-1} and computing the inversion recursively, typical inversion will take $O(n^3)$, while this special case it is O(n)

Correlation matrix

- Data matrix

$$X = \begin{pmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_N \\ | & | & & | \end{pmatrix}$$

- Correlation matrix $XX^T = x_1x_1^T + x_2x_2^T + \cdots + x_Nx_N^T$
- Inverse of correlation matrix when adding x

$$(XX^T + xx^T)^{-1} = (XX^T)^{-1} - \frac{(XX^T)^{-1}xx^T(XX^T)^{-1}}{1 + x^T(XX^T)^{-1}x}$$

- Inverse of correlation matrix when removing x

$$(XX^T - xx^T)^{-1} = (XX^T)^{-1} + \frac{(XX^T)^{-1}xx^T(XX^T)^{-1}}{1 - x^T(XX^T)^{-1}x}$$

Review of over-complete LLSE

- LLSE objective $\min_w \|y - X^T w\|^2$
  - Solution (over-complete) $w = (XX^T)^{-1}Xy$
  - LS Error at optimum $y^T y - (Xy)^T (XX^T)^{-1} Xy$
    - To prove: the prediction $(I_N - X^T(XX^T)^{-1}X)y$ matrix $A = (I_N - X^T(XX^T)^{-1}X)y$ is idempotent, i.e., AA = A, so the LS error

      $$y^T(I_N - X^T(XX^T)^{-1}X)^T(I_N - X^T(XX^T)^{-1}X)y$$

      becomes $y^T y - (Xy)^T(XX^T)^{-1}Xy$

LOO for LLSE

- Each LOO pass corresponds to the removal of a particular data point from the data matrix, so correspondingly we can compute the updated parameter and error using matrix inversion lemma
- Parameter

$$
\begin{aligned}
\hat{w} &= (XX^T - x_i x_i^T)^{-1}(Xy - y_i x_i) \\
&= \left((XX^T)^{-1} + \frac{(XX^T)^{-1} x_i x_i^T (XX^T)^{-1}}{1 - x_i^T (XX^T)^{-1} x_i} \right)(Xy - y_i x_i) \\
&= w + \frac{x_i^T w - y_i}{1 - x_i^T (XX^T)^{-1} x_i} (XX^T)^{-1} x_i
\end{aligned}
$$

LOO for LLSE

- The prediction on the single data point that is left out is

$$y_i - x_i^T \hat{w} = y_i - x_i^T \left(w + \frac{x_i^T w - y_i}{1 - x_i^T (XX^T)^{-1} x_i} (XX^T)^{-1} x_i \right)$$

$$= y_i - x_i^T w - \frac{x_i^T w - y_i}{1 - x_i^T (XX^T)^{-1} x_i} x_i^T (XX^T)^{-1} x_i$$

$$= \frac{y_i - x_i^T w}{1 - x_i^T (XX^T)^{-1} x_i}$$

- So LSE loss on the single data point that is left out is

$$\frac{(y_i - x_i^T w)^2}{(1 - x_i^T (XX^T)^{-1} x_i)^2}$$

Overall computation complexity

- Instead of inverting N matrices of dimension m x m, which will have a time complexity of $O(Nm^3)$, the LOO algorithm can compute the inverse of the overall correlation matrix once and reuse it for all LOO objective computation, which leaves a time complexity of $O(Nm+m^3)$
- Averaging LOO LSE loss can be used to choose from different model families, e.g., when fitting data, this will be polynomials of different degrees

Summary

- Incremental LLSE model selection based on LOO error
  - For each degree d = 1, …, D:
    - Compute optimal parameter using incremental LLSE w_d
    - Compute LOO error using w_d
  - Select the optimal model with the minimum LOO error