Lecture 3: Challenges in Machine Learning

DD2421

Atsuto Maki

Autumn, 2017

How should we select/determine the right model f from data?

Basic idea for classification:

Given training data

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$$

of inputs $\mathbf{x}_i \in \mathbb{R}^d$ and their labels y_i.

Compute the misclassification rate on D

$$err(f, D) = \frac{1}{N} \sum_{i=1}^{N} \mathrm{Ind}\left(f(\mathbf{x}_i) \neq y_i\right)$$

Note: $\mathrm{Ind}\,(x) = 1$ if $x =$ TRUE otherwise $\mathrm{Ind}\,(x) = 0$

Overfitting

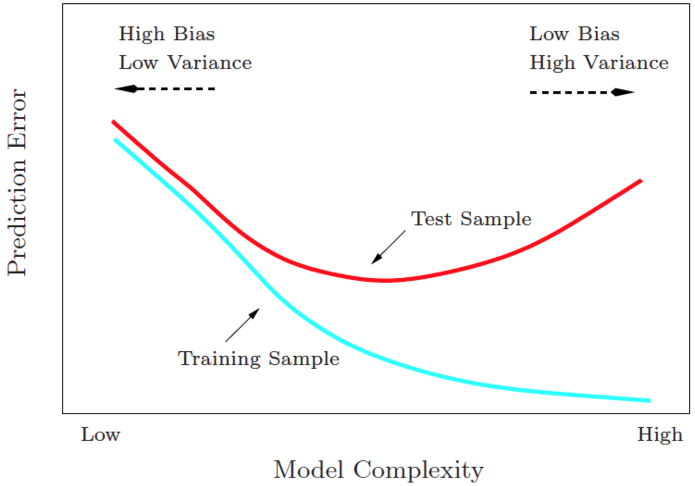
Visited in Lecture 2 using decision tree.

Good results on training data, but generalizes poorly.
This occurs due to

- Non-representative sample
- Noisy examples

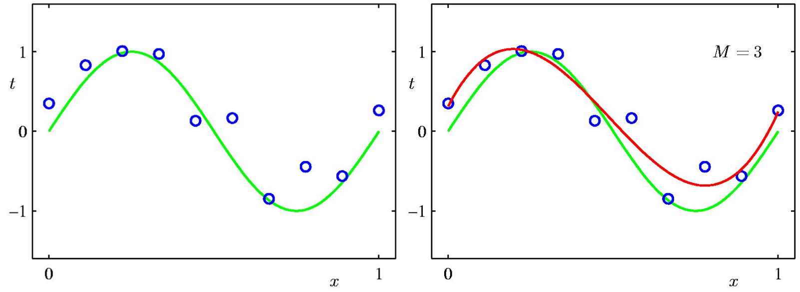
Overfitting

When the learned models are overly specialized for the training samples.



(T. Hastie et al, The Elements of Statistical Learning)

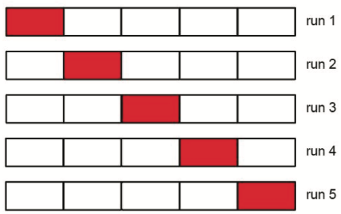
Example: Polynomial Curve Fitting (regression to sinusoidal)



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

(C. Bishop, Pattern Recognition and Machine Learning)

K-hold cross validation (schematic for $K = 5$)



(K. Murphy, Machine Learning – A probabilistic perspective)

- *Training* set T: to fit the models
- *Validation* set V: to estimate prediction error for model selection (i.e. to determine *hyperparameters*)

If we are in a data-rich situation:

$\rightarrow$ partition the data into three sets, *Training* set, *Validation* set, and *Test* set for assessment of the generalization error of the final chosen model.

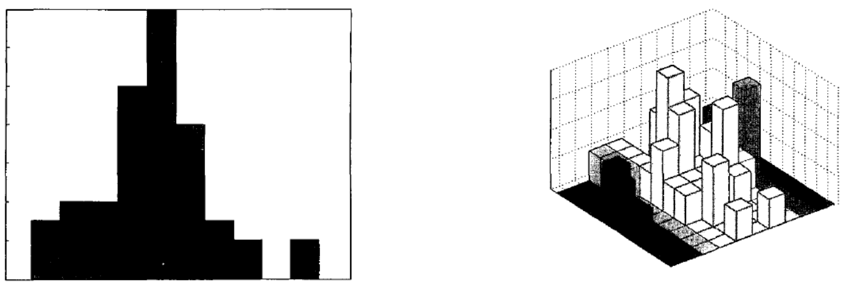
Curse of Dimensionality

Curse of Dimensionality

Imagine: inputs represented by 30 features but some of them are less relevant to target function. Will you use all of them?

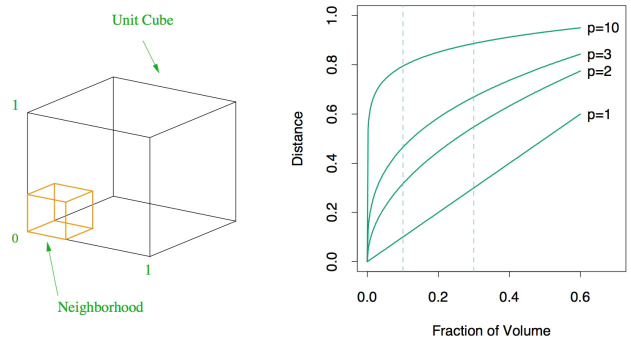
- Easy problems in low-dimensions are harder in high-dimensions
  – training more complex model with limited sample data
- In high-dimensions everything is far from everthing else
  – issues in Nearest Neighbours
- Any method that attempts to produce locally varying functions in small isotropic neighbourhoods will run into problems in high dimensions.

Example 1: Normal random numbers in 1-d and 2-d (both plots for 100 inputs)



Too few data to represent the probability density function in 2-d.

Example 2: A subcubical neighbourhood for uniform data in a unit cube.



(T. Hastie et al, The Elements of Statistical Learning)

Graph: The side-length of the subcube needed to capture a fraction of the volume of the data (for different dimensions p).

Intuitions in low-dimensions do not apply in high-dimensions
Real world is in 3-d, but we deal with data for instance in 1000-d

- Uniform distribution on hypercube
- Volume of hypersphere

Techniques for dimensionality reduction / feature selection exist.

The Bias-Variance Trade-off

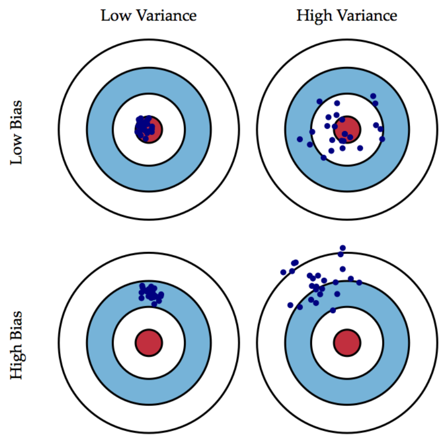
Concepts of prediction errors

Let us imagine we could repeat the modeling for many times – each time by gathering new set of training samples, $\mathcal{D}$.

The resulting models will have a range of predictions due to randomness in the underlying data set.

- Error due to **Bias**: the difference between the average (expected) prediction of our model and the correct value.
- Error due to **Variance**: the variability of a model prediction for a given data point between different realizations of the model.

Graphical illustration of bias and variance



(figure source: http://scott.fortmann-roe.com/docs/BiasVariance.html)

The bias-variance decomposition

Let us consider

$f(\mathbf{x})$: true function

$\hat{f}_{\mathcal{D}}(\mathbf{x})$: prediction function (= model) estimated with $\mathcal{D}$

and a conceptual tool:

$E_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(\mathbf{x})]$: average of models due to different sample sets (NOTE: it's shown simply as $E[\hat{f}_{\mathcal{D}}(\mathbf{x})]$ in the sequel)

The mean square error (MSE) for estimating $f(\mathbf{x})$

$$\begin{aligned} E_{\mathcal{D}}[(\hat{f}_{\mathcal{D}}(\mathbf{x}) - f(\mathbf{x}))^2] &= E_{\mathcal{D}}[(\hat{f}_{\mathcal{D}}(\mathbf{x}) - E[\hat{f}_{\mathcal{D}}(\mathbf{x})])^2] + (E[\hat{f}_{\mathcal{D}}(\mathbf{x})] - f(\mathbf{x}))^2 \\ &= \textbf{Variance} + (\textbf{Bias})^2 \end{aligned}$$

To complete, we compute: $E_{\mathbf{x}}[E_{\mathcal{D}}[(\hat{f}_{\mathcal{D}}(\mathbf{x}) - f(\mathbf{x}))^2]]$

The mean square error (MSE) for estimating $f(\mathbf{x})$ is two-fold.
(derivation of decomposition at the lecture)

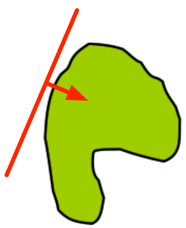
$$\begin{aligned} (\hat{f}_{\mathcal{D}}(\mathbf{x}) - f(\mathbf{x}))^2 &= (\hat{f}_{\mathcal{D}}(\mathbf{x}) - E[\hat{f}_{\mathcal{D}}(\mathbf{x})] + E[\hat{f}_{\mathcal{D}}(\mathbf{x})] - f(\mathbf{x}))^2 \\ &= (\hat{f}_{\mathcal{D}}(\mathbf{x}) - E[\hat{f}_{\mathcal{D}}(\mathbf{x})])^2 + (E[\hat{f}_{\mathcal{D}}(\mathbf{x})] - f(\mathbf{x}))^2 \\ &+ 2(\hat{f}_{\mathcal{D}}(\mathbf{x}) - E[\hat{f}_{\mathcal{D}}(\mathbf{x})])(E[\hat{f}_{\mathcal{D}}(\mathbf{x})] - f(\mathbf{x})) \end{aligned}$$

Taking $E_{\mathcal{D}}[\ \dots\ ]$ for both sides, the cross term disappears (!) while the second term stays the same.

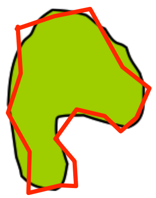
Characterization of a classifier: Bias

Bias of a classifier is the discrepancy between its averaged estimated and true function

$$E[\hat{f}_{\mathcal{D}}(\mathbf{x})] - f(\mathbf{x})$$



High-bias classifier



Low-bias classifier

Low model complexity (small # of d.o.f.) $\Longrightarrow$ High-bias
High model complexity (large # of d.o.f.) $\Longrightarrow$ Low-bias

Characterization of a classifier: Variance

Variance of a classifier is the expected divergence of the estimated prediction function from its average value:

$$E_{\mathcal{D}}[(\hat{f}_{\mathcal{D}}(\mathbf{x}) - E[\hat{f}_{\mathcal{D}}(\mathbf{x})])^2]$$

This measures how dependent the classifier is on the random sampling made in the training set.

Low model complexity (small # of d.o.f.) $\Longrightarrow$ Low-variance
High model complexity (large # of d.o.f.) $\Longrightarrow$ High-variance

High variance classifiers produce differing decision boundaries which are highly dependent on the training data.

Also called "flexible".

Examples:

1. *decision trees*

The depth of the tree determines the variance. How?

2. *k Nearest-Neighbour*

k determines the variance. How?

Our intuition may tell:

- The presence of bias indicates something basically wrong with the model and algorithm...
- Variance is also bad, but a model with high variance could at least predict well on average...

So the model should minimize bias even at the expense of variance??

Not really!

Bias and variance are equally important as we are always dealing with a single realization of the data set.

Take home message: Match the model complexity to the data resources, not to the target complexity