CISC 372
Advanced Data Analytics
L6 – Ensemble Method

https://l1nna.com/372

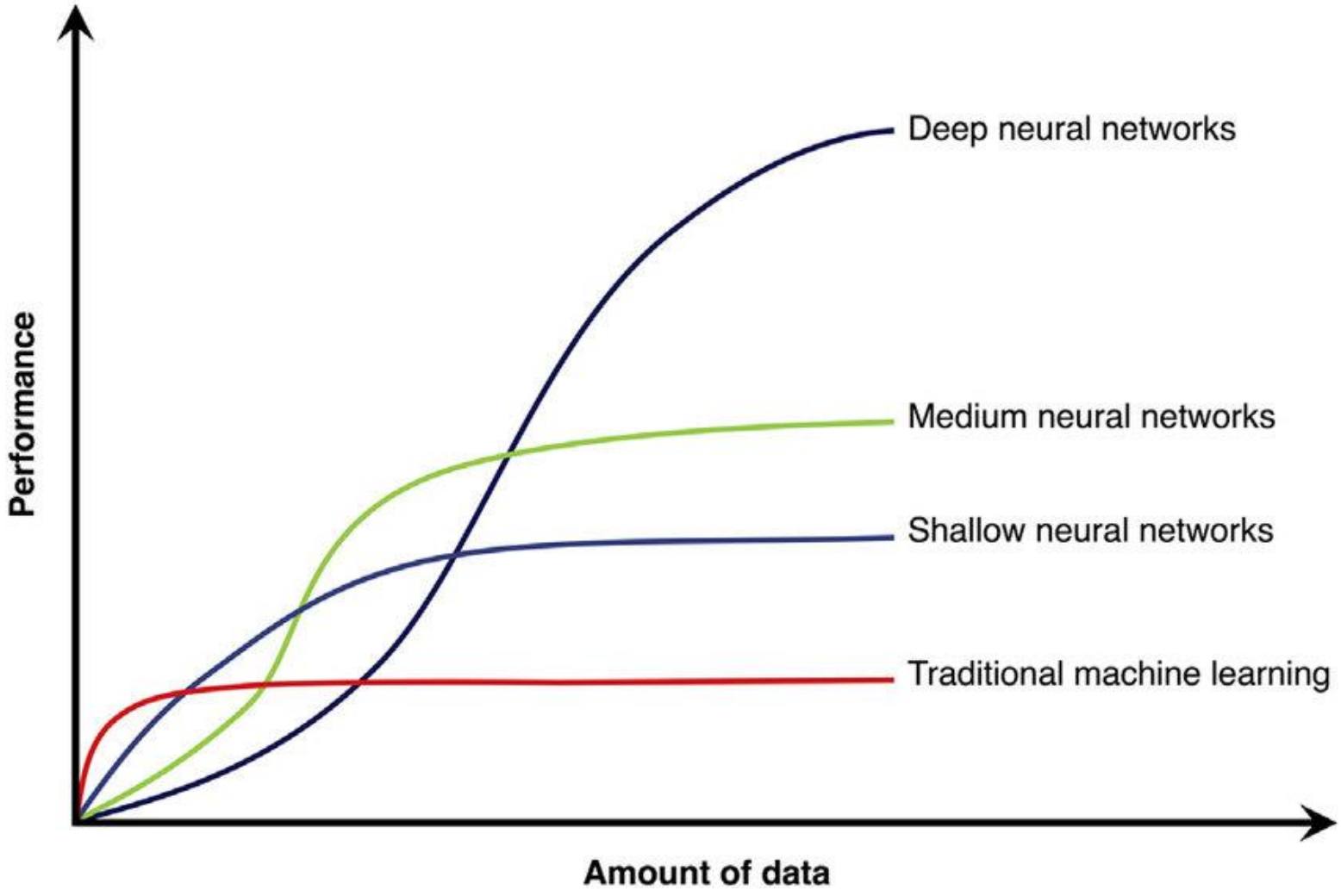
Yesterday

- Underfitting vs. Overfitting
- Hyper-parameter tuning & Experimental Protocol
- KDD Process
  - Iterative
  - Data lifecycle
- Data Attributes
  - Numeric/Nominal/Binominal/Ordinal
- Data Types:
  - Relational records
  - Data Metric
  - Document Data
  - Graph Data
  - Structured vs unstructured data
- Data Characteristics
  - Dimensionality/Sparsity
- Data Preprocessing
  - Normalization/Standardization/Encoding/OOV/Discretization/

Today

- Ensemble Method
  - Bias-Variance decomposition
  - Bagging
  - Boosting

Error vs. Data size



Bias-Variance decomposition

- Suppose we have a dataset we want to model.

- Let **P** be some property of ***the perfect model*** which we can imagine, but can't build.

- Whenever we build an actual model, it has property **Q** which is an estimate of **P**, derived from the data.

Bias-Variance decomposition

If we used a different sample, we would get a different value of **Q**, so **Q** is a random variable from some distribution.

E[Q] is the expected value (mean) of **Q**

The bias of **Q** is

$$\text{bias}(Q) = E[Q] - P$$

which measures how **Q** differs from **P** systematically.

Bias-Variance decomposition

The variance of **Q** is

$$\mathrm{Var}(Q) = E[\, Q - E[Q] \,]^2$$

and measures the effect of **individual datasets** on the **observed** value of **Q**.

Bias-Variance decomposition

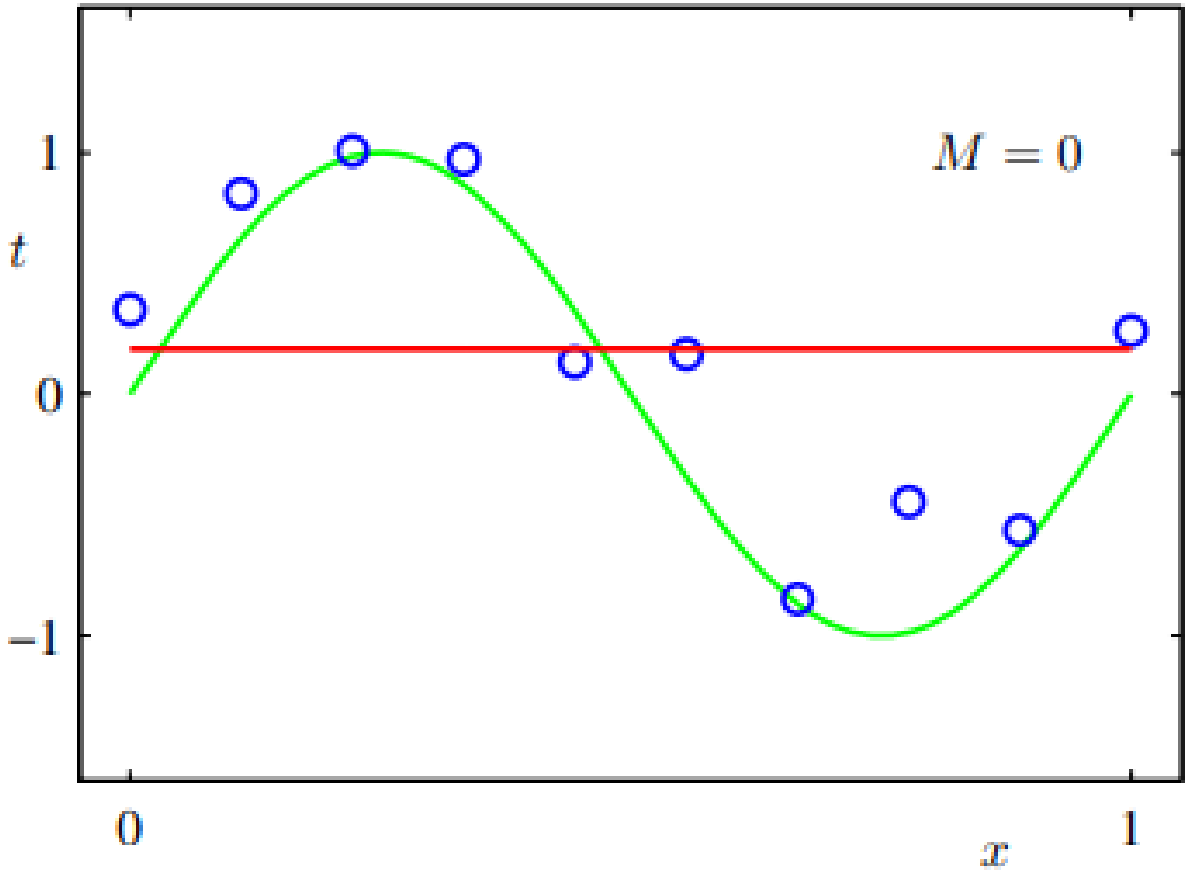
The complexity of the model used to represent the dataset has an effect on both bias and variance.

A very **inflexible** model (hardly any settable parameters) has low variance, but almost certainly high bias.

Suppose we always estimate **Q = 1**. Then the variance is 0. But the bias is probably large.

Bias-Variance decomposition

A **inflexible** model



Bias-Variance decomposition

A **flexible** model has lots of settable parameters, so a small change in the data might produce a large change in Q.

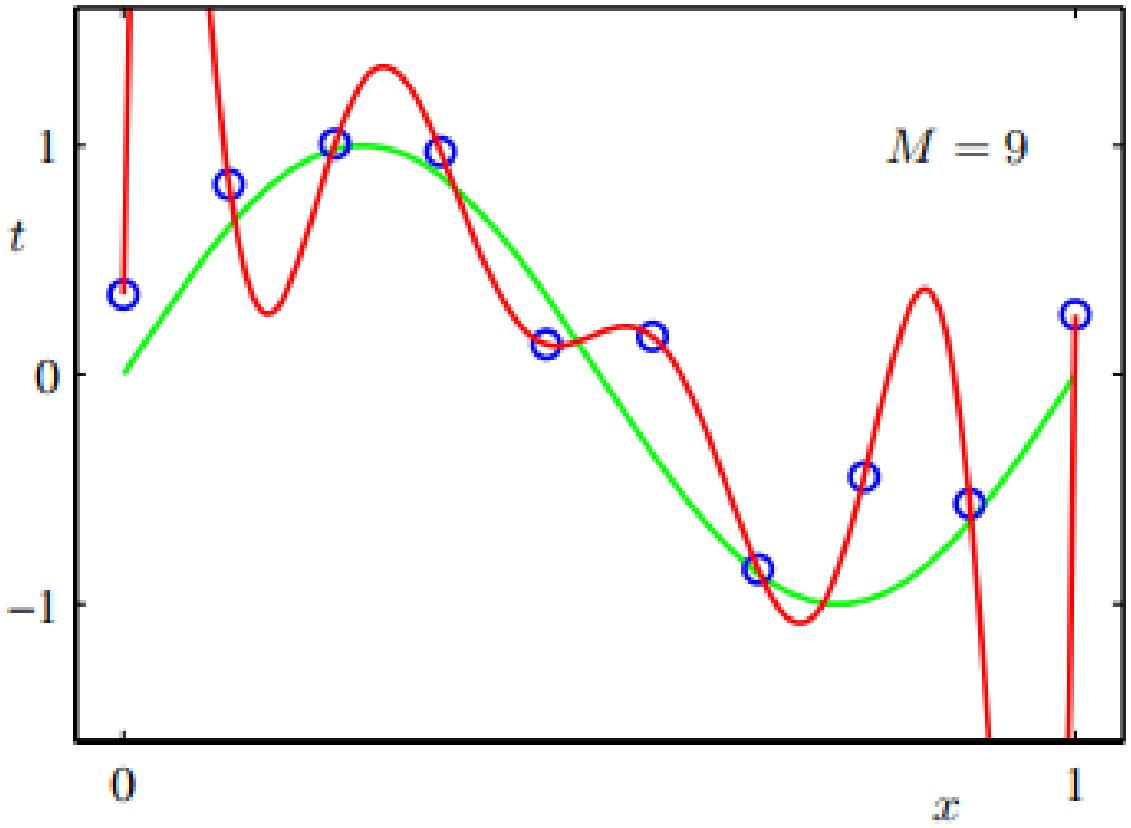
A flexible model has large variance.

But its **flexibility** allows it to model the data well, so it typically has **low bias**.

We want to build models with the best tradeoff between bias and variance.

Bias-Variance decomposition

A **flexible** model



Bias-Variance decomposition

The **mean squared error** is

$$E[\ (\mathbf{Q} - \mathbf{P})^2\ ]$$

With a bit of algebra

$$E[\ (Q - P)^2\ ] = E[\ (Q - E[Q] + E[Q] - P)^2\ ]$$

$$= (\ E[Q] - P\ )^2 + E[\ (Q - E[Q])2\ ]$$

$$= \mathbf{(Bias(Q))^2} + \mathbf{Var(Q)}$$

Bias-Variance decomposition

This is why the mean squared error is such a common metric for model quality.

But note that **it pays too much attention to bias**.

=> mean squared error gives a lot of **weight to bad errors**;

and the same weight to errors under and over – both of which might not be appropriate in some settings.

The decomposition theorem

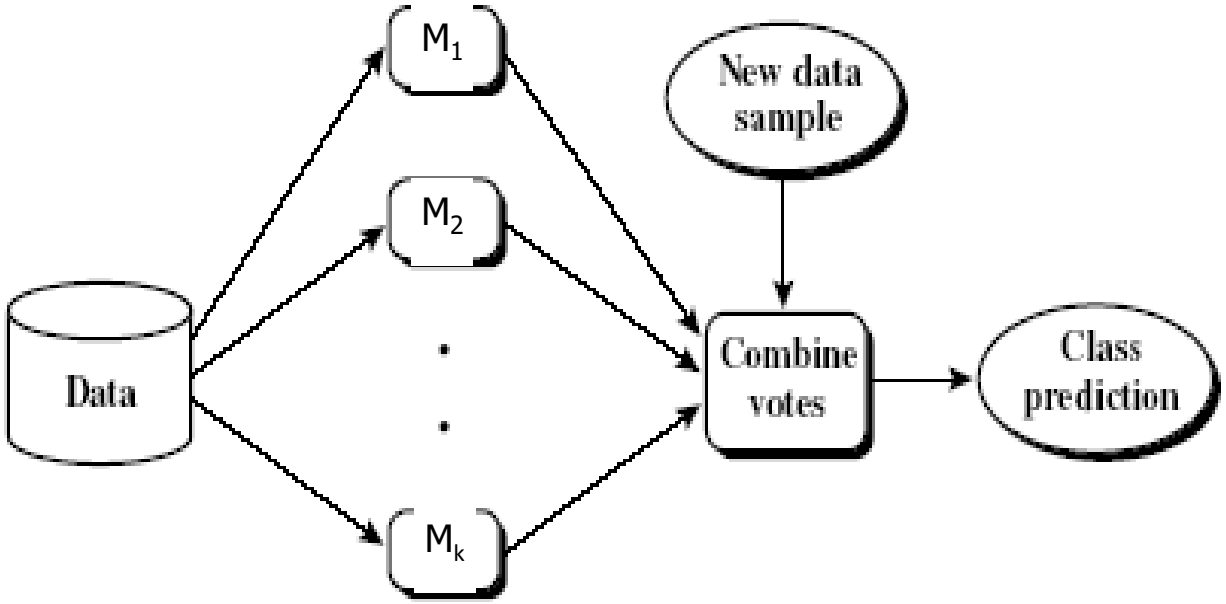
$$\text{Error}(Q) = \text{BCE}(Q) + \text{Bias}(Q) + \text{Var}(Q)$$

where **BCE(Q)** is ***the irreducible error*** of a perfect model (comes from the variance of the errors).

$\Rightarrow$the error that can't be reduced by using only observed

- $\Rightarrow$intrinsic to the process being observed.
- $\Rightarrow$the amount of noise in **our data**.

Ensemble methods



Use a combination of models to increase accuracy
Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*

Bagging: Boostrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote

Bagging: Boostrap Aggregation

- Training
  - Given a set D of *d* records, at each iteration *i*, a training set $D_i \subset$ D tuples is **sampled with replacement** from D
    - Like we did for bootstrap, but this is only for training
  - A classifier model M_i is learned for each training set D_i

- Classification: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the majority votes to **X**

- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple

Bagging: Boostrap Aggregation

As long as the model being used is **complex *enough***, each of the individual predictors will have small biases.

The effect of the voting is ***to cancel out the errors due to variance***.

So overall the errors will tend to be small.

- Accuracy
  - Often significantly better than a single classifier derived from D
  - For noisy data: not considerably worse, more **robust**
  - Proved improved accuracy in prediction

Bagging: Boostrap Aggregation

- Required: sampling with replacement
  - Samples have the right variance properties.

In practice, many variations seem to give decent results. N.B. especially a partition instead of a set of samples, which is required for a parallel algorithm.

This is probably because real datasets tend to be repetitive.

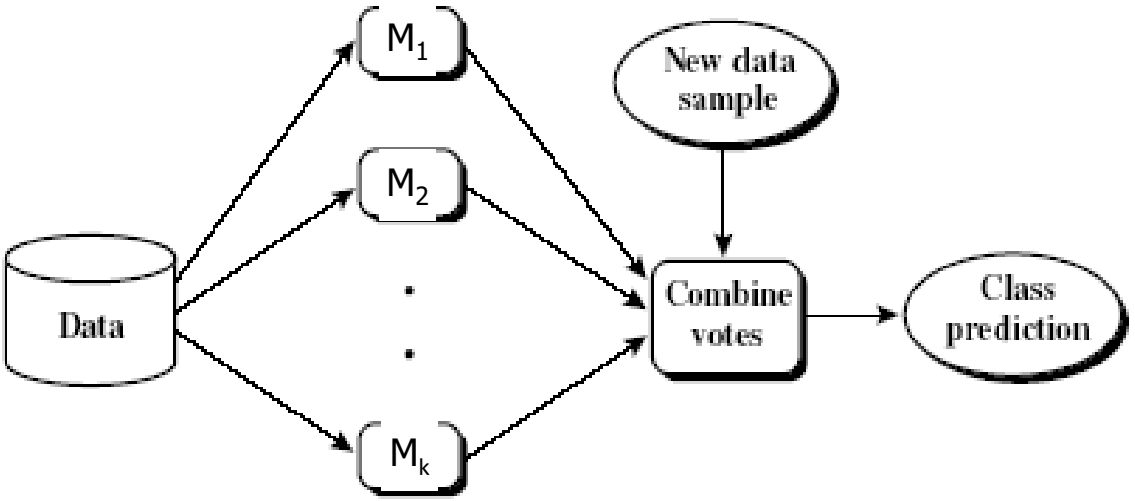
Boosting

In bagging, all samples and so all predictors are created equal.

However, as a predictor is built from the first sample, we learn something about **which objects are the hardest to classify**.

Use this information to select a better sample to learn from the second time, the third time,

Boosting



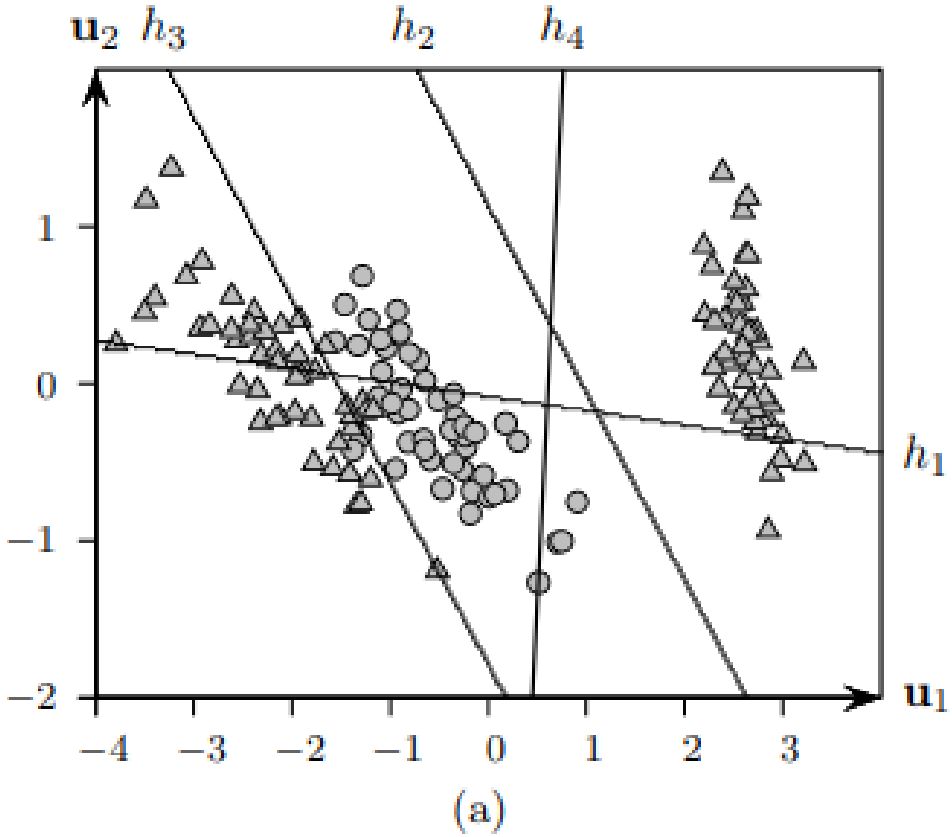
- How boosting works?
  - **Equal Weights** are assigned to each training tuple
  - A series of **k** classifiers is iteratively learned
    - Sample data based on the **weights**
    - Build classifier $\mathbf{M_i}$
    - Weights updated for the whole dataset, $\mathbf{M_{i+1}}$,
    - Until built k classifiers
  - The final M* combines the votes of each individual classifier, where the **weight** of each classifier's vote is a function of its **accuracy**

Boosting

Weight	Rec ID	Attribs.	Class	Correct?
1	100	…	Yes	✓
1	101	…	Yes	✓
1	102	…	Yes	✗
1	103	…	No	✓
1	104	…	No	✓

Weight	Rec ID	Attribs.	Class	Correct?
1	100	…	Yes	✓
1	101	…	Yes	✗
1.2	102	…	Yes	✓
1	103	…	No	✓
1	104	…	No	✓

Boosting – SVM (Linear kernel)



M_t	h_1	h_2	h_3	h_4
e_t	0.280	0.305	0.174	0.282

combined model	$\mathbf{M}^1$	$\mathbf{M}^2$	$\mathbf{M}^3$	$\mathbf{M}^4$
training error rate	0.280	0.253	0.073	0.047

Boosting

In bagging, all of the predictors are equal.

In boosting, the predictors form a sequence, with the later predictors `specializing' in difficult objects.

- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: boosting tends to achieve **greater accuracy**, but it also **risks overfitting** the model to misclassified data

Summary

- More is not necessarily better.
- Although model quality improves rapidly with exposure to new objects, and then flattens, this is surprisingly tricky to exploit.
- It turns out that the best thing to do is to build models from small bitesize chunks of data, and then vote their predictions.