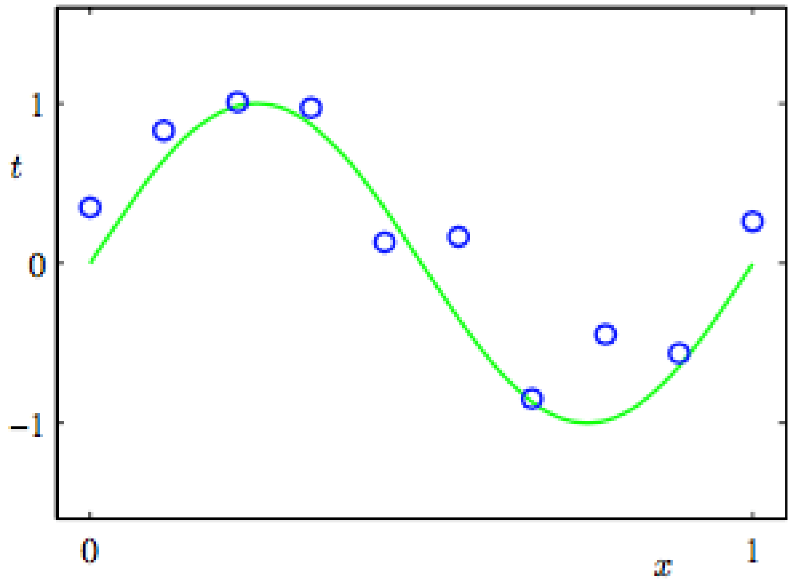
COMP3055
Machine Learning

Topic 5 – Machine Learning Theory and Practice

Ying Weng
2024 Autumn

Motivating Example

Polynomial Curve Fitting



Plot of a training data set of $N = 10$ points, each comprising an observation of the input variable x along with the corresponding target variable t. The green curve shows the function $\sin(2\pi x)$ used to generate the data. Our goal is to predict the value of t for some new value of x, **without** knowledge of the green curve.

Classification VS Regression

Classification
predict a label (discrete value)

Regression
predict a response (continuous value)

Regression Problem

- $X \equiv (x_1, \dots, x_N)^T,\ T \equiv (t_1, \dots, t_N)^T$
- ***Training set*** is generated with $t_n = sin(2\pi x_n) + b_n, n = 1,2, \dots, N$, where b_n is random noise having a Gaussian Distribution.
- ***Goal:*** predict the target value of t for some new input value x ➔ implicitly trying to discover the underlying function $sin(2\pi x_n)$.
- Generalize from a finite data set *(N=10)*
- **Uncertainty**: the observed data are corrupted with noise.

Polynomial Function

- We fit the training data using a polynomial function (linear models) of the form:

$$y(x, \boldsymbol{w}) = w_0 + w_1 x^1 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

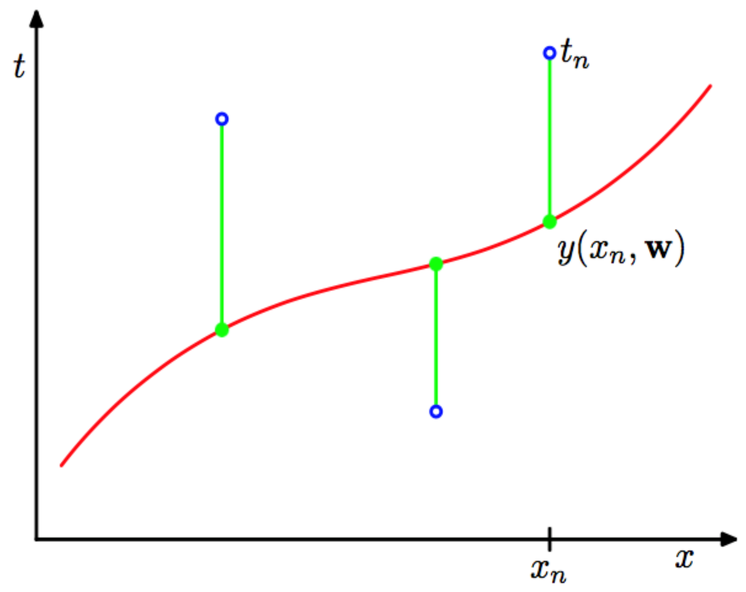
- Where M is the *order of the polynomial* (degree of freedom), and the *polynomial coefficients* $w_0, w_1, \ldots, w_M$ are collectively denoted by the vector $\boldsymbol{w}$

Solve a Polynomial Function

Target: find the values of *polynomial coefficients*

- Step 1: Fit the polynomial to the training data
- Step 2: minimize the *error function*

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \boldsymbol{w}) - t_n\}^2$$

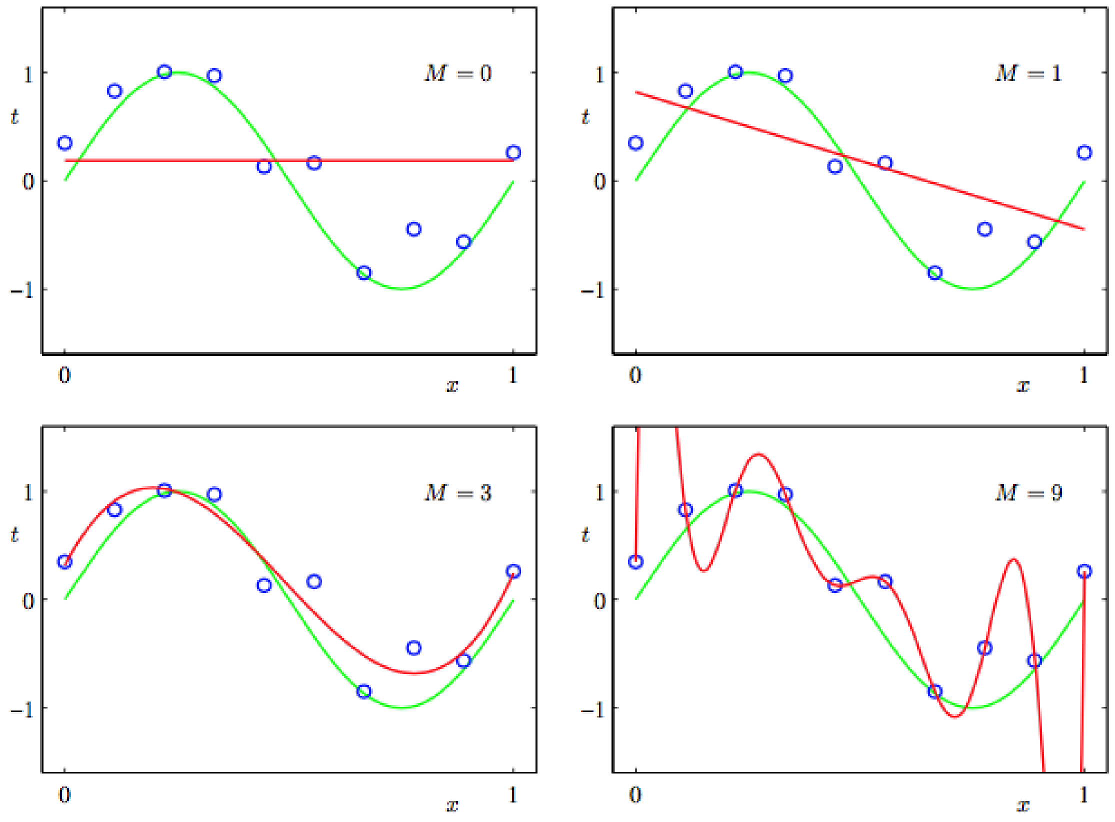


Where E(w) measures the misfit between the function $y(x, \boldsymbol{w})$ and the training set data points

½ is included for later convenience. E(w)=0 if and only if the function y(x,w) were to pass exactly through each training data point. The error function is a quadratic function of w, there is only one solution.

Model Selection

Choosing order M of the polynomial



Model Selection

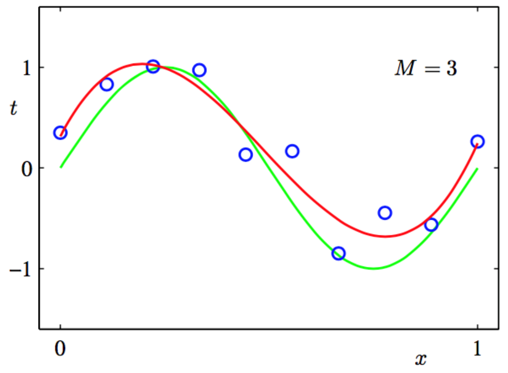
- **Goal**: achieve good *generalization* by making accurate predictions for new data.

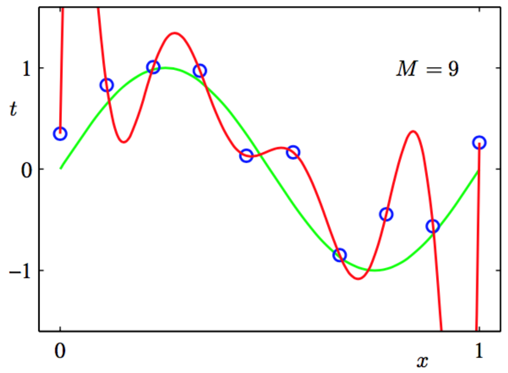
- We use Root-mean-square (RMS) error on data
$$E_{RMS} = \sqrt{2E(w^*)/N}$$

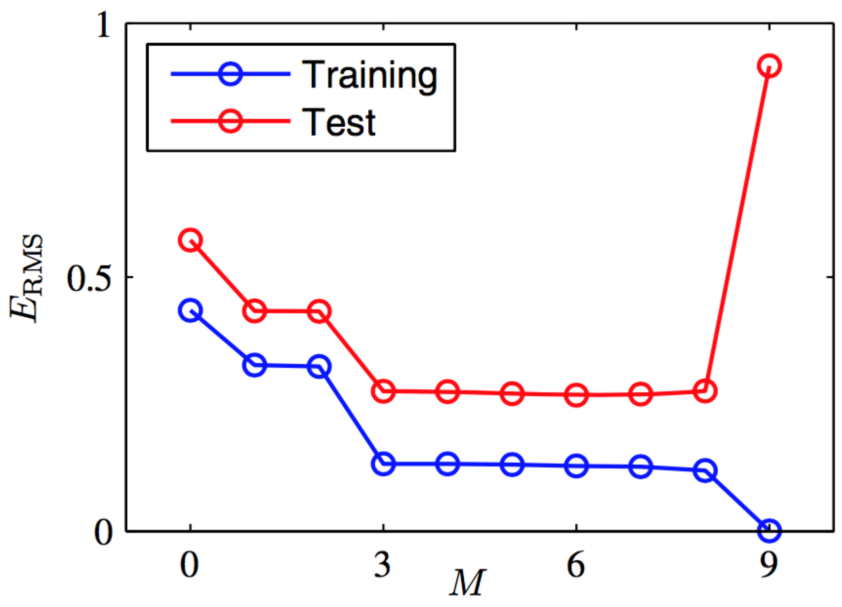
- Where N allows us to compare different sizes of data set, and w^* is the solution of minimizing $E(w)$ (*hypothesis*).

- It measures how well the model w^* doing in predicting the values of *t* for new data observations of *x*.

Overfitting







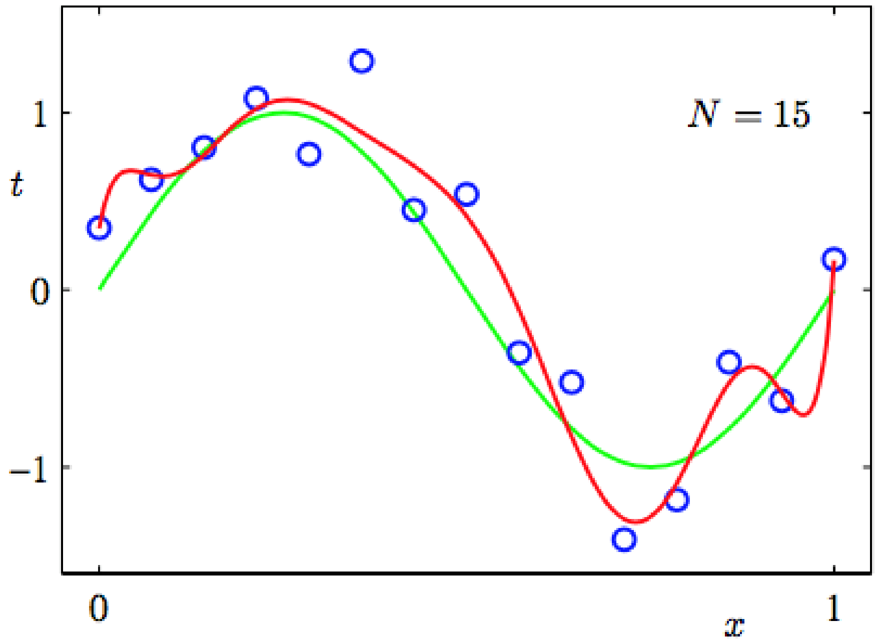
	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

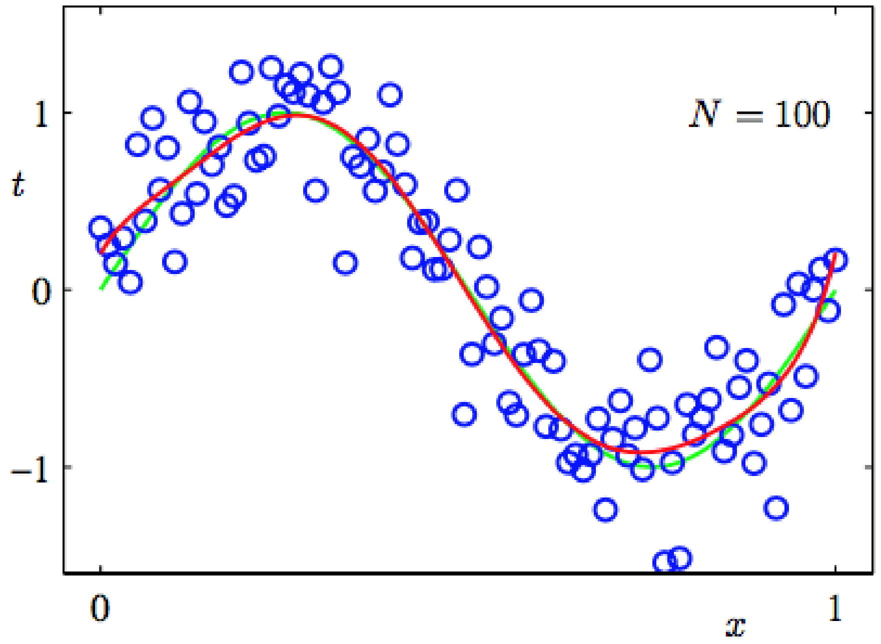
Overfitting

Overfitting can occur when:

- Learning is performed for too long (e.g. in Neural Networks).
- The examples in the training set are not representative of all possible situations (is usually the case!).
- Model parameters are adjusted to uninformative features in the training set that have no causal relation to the true underlying target function!

Overfitting





Increasing the size of the data set reduces the overfitting problem.

Regularization

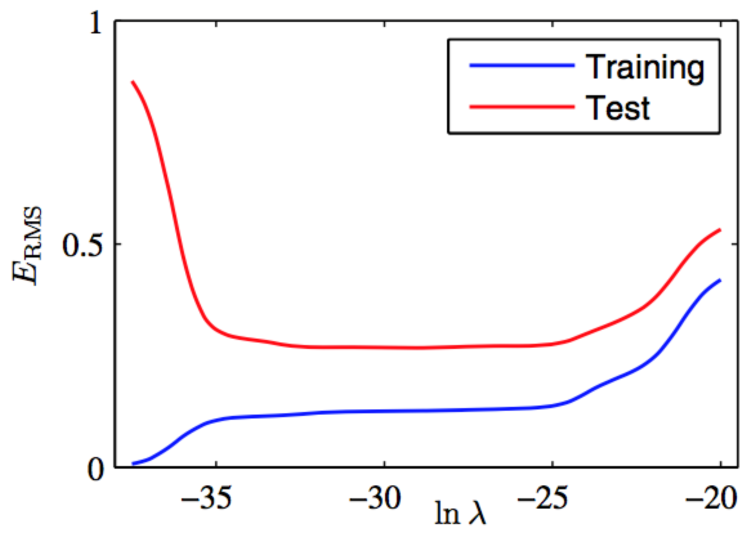
- *Regularization* can control the overfitting phenomenon, by adding penalty term to the error function to discourage the coefficients from reaching large values.

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \frac{\lambda}{2}\parallel w \parallel^2$$

- Where $\parallel w \parallel^2 \equiv w^T w = w_0^2 + w_1^2 + \cdots + w_M^2$, and the coefficient λ governs the relative importance of the regularization term compared with the sum-of-squares error term.

Regularization

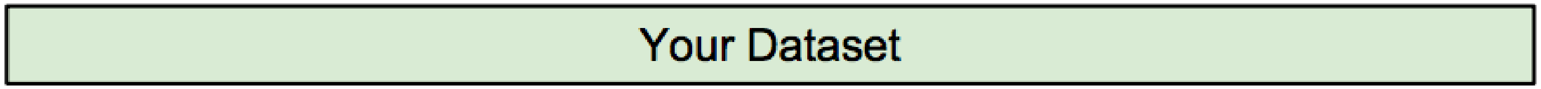
	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01



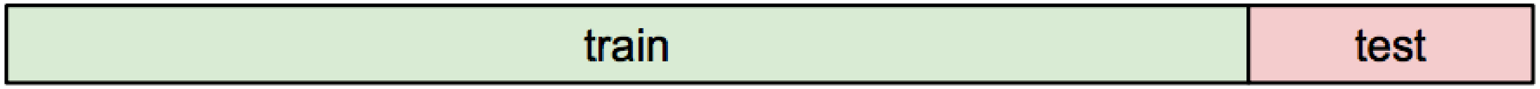
- Table of the coefficients $w^*$for $M = 9$ polynomials with various values for the regularization parameter λ.
- λ controls the *effective complexity* of the model and hence determines the *degree of overfitting*

Cross Validation

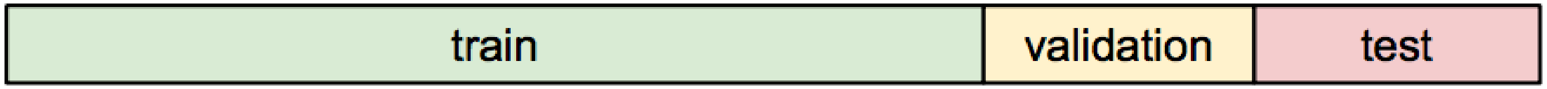
- **Idea #1**: Choose hyperparameters that work best on the data



- **Idea #2**: Split data into **train** and **test**, choose hyperparameters that work best on test data

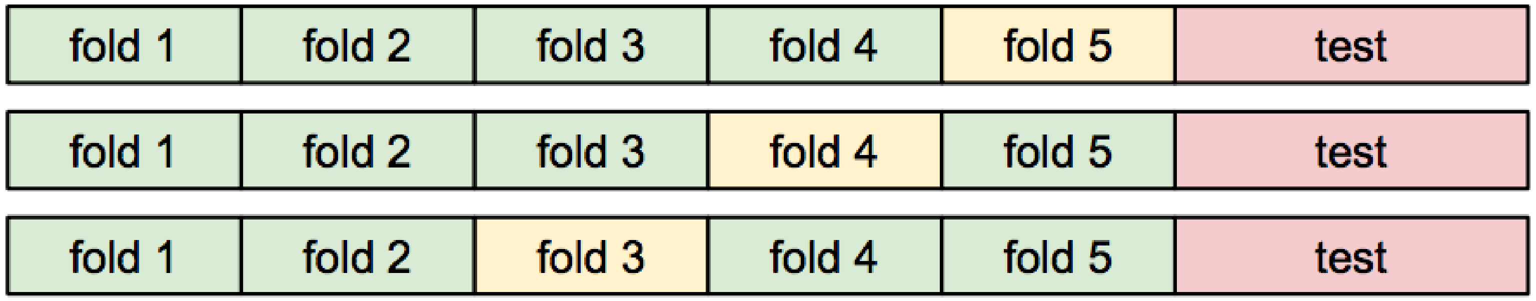


- **Idea #3**: Split data into **train**, **val**, and **test**; choose hyperparameters on **val** and evaluate on **test**



Cross Validation

- **Idea #4**: **Cross-Validation**: Split data into **folds**, try each fold as validation and average the results



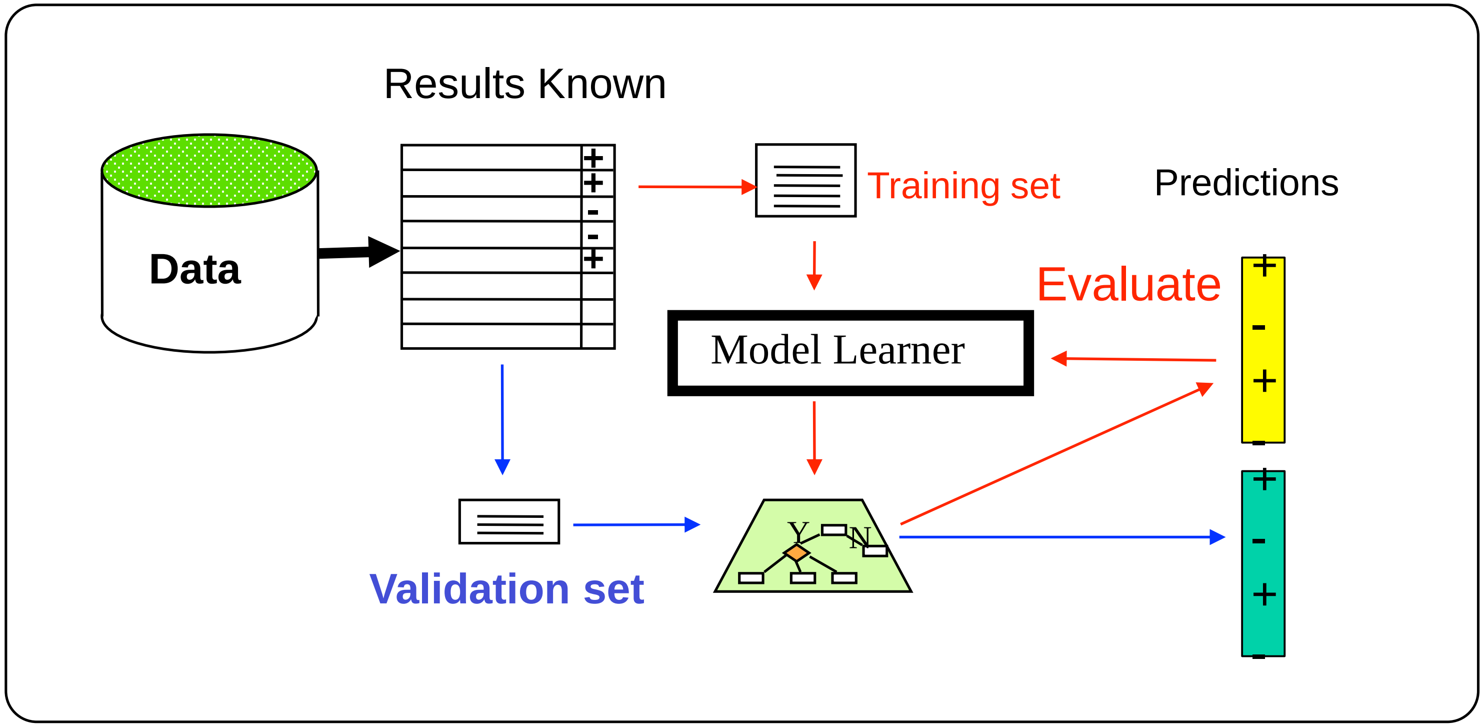
Cross Validation

- **Cross Validation** is often used to counter overfitting.

- Partition the dataset into S groups, with $(S\text{-}2)$ training sets, a validation set and a testing set.
  - The training set is used to determine the coefficients $\boldsymbol{w}$
  - The validation set is used to optimize the model complexity (hyperparameters, either M or λ in the previous example)
  - The testing set is used to evaluate the final selected mode

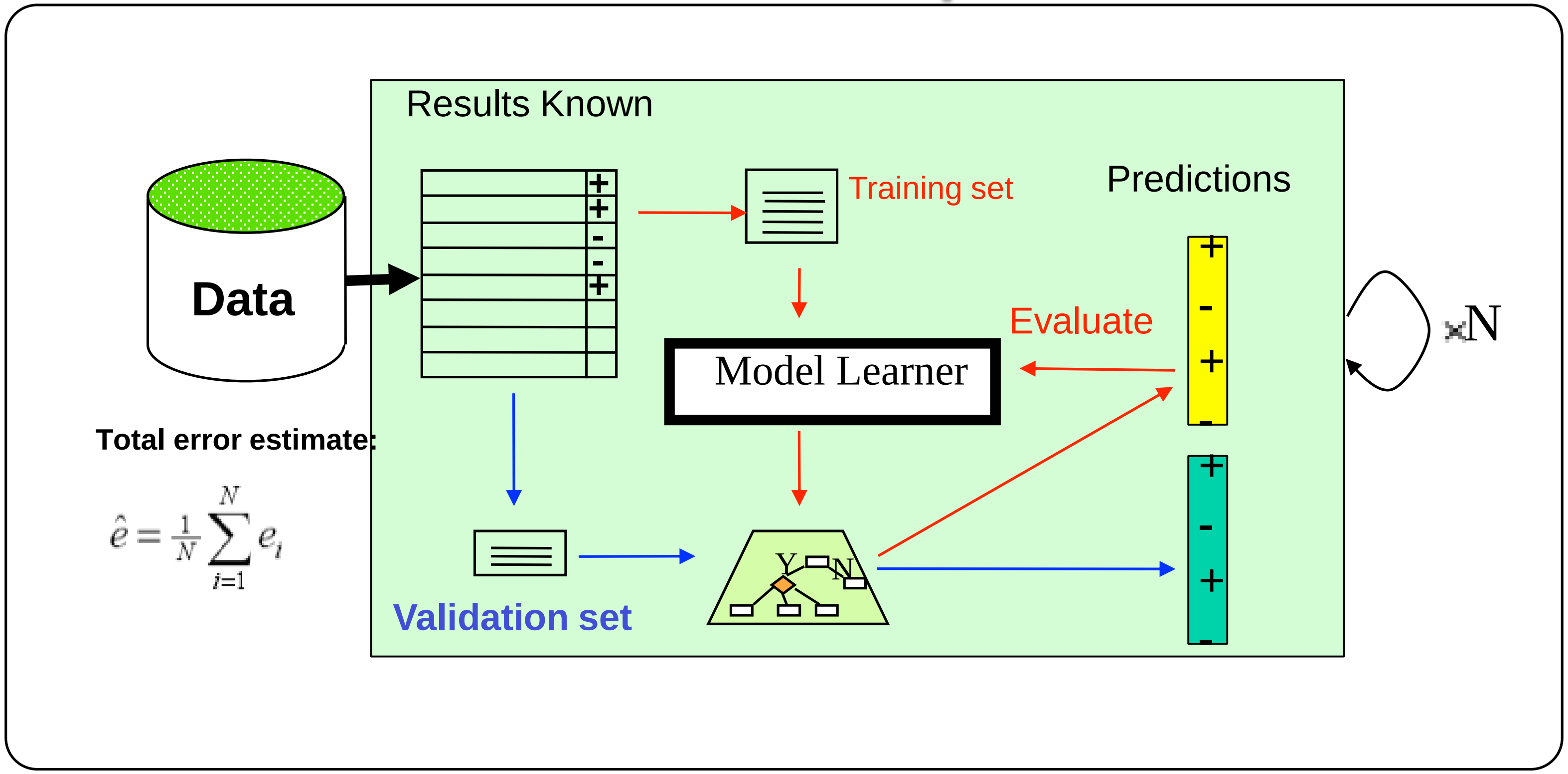
- The procedure is then repeated for all S possible choices, the performance scores from the S runs are then averaged.

Evaluation procedure



- For large datasets, a single split is usually sufficient.
- For smaller datasets, rely on cross validation

Cross validation procedure



$$\hat{e} = \frac{1}{N}\sum_{i=1}^{N} e_i$$

- Split the data into training, validation and test sets in a repeated fashion.
- Estimate the total error as the average of each fold error.

Classification Measures - Error Rate

- Common performance measure for classification problems
  - Success: instance's class is predicted correctly (True Positives (**TP**) / Negatives (**TN**))
  - Error: instance's class is predicted incorrectly (False Positives (**FP**) /Negatives (**FN**))
  - False positives - **Type I error**. False Negative - **Type II error**

- Classification **error rate**: proportion of instances misclassified over the whole set of instances

  Error Rate = (**FP**+**FN**)/(**P**+**N**)

  a.k.a, **accuracy**

  Accuracy = 1 - **Error Rate**

- Classification Error Rate on the Training Set can be too optimistic!

Unbalanced data

- Balanced set: (roughly) equal number of positive / negative examples:

Classifier	TP	TN	FP	FN	Recall Rate
A	25	25	25	25	50%
B	37	37	13	13	74%

Recall = TP/(TP+FN)

Unbalanced data

- Unbalanced set: unequal number of positive / negative examples

Classifier	TP	TN	FP	FN	Recall Rate
A	25	75	75	25	50%
B	0	150	0	50	0%

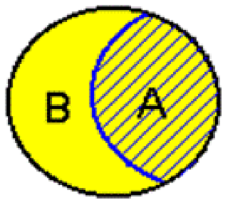
Classifier B cannot predict any positive examples!

Recall = TP/(TP+FN)

Classification Measures

- Error Rate (Accuracy)
- Precision/Recall
- F-measure
- ROC curve
- Confusion matrix
- …

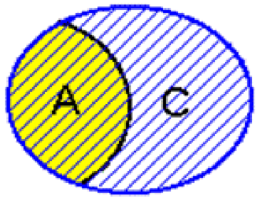
Recall/Precision



B: Number of positive examples not retrieved

A: Number of positive examples retrieved

RECALL: $\frac{A}{A+B} \times 100\%$



C: Number of negative examples retrieved

A: Number of positive examples retrieved

PRECISION: $\frac{A}{A+C} \times 100\%$

More insight over a classifier's behavior

For the positive class:

Classifier A: Recall = 50%, Precision = 25%

Classifier B: Recall = 0%, Precision = 0%

Classifier B is useless!

F-measure

- Comparing different approaches is difficult when using multiple evaluation measures (e.g. Recall and Precision)

- F-measure combines recall and precision into a single measure:

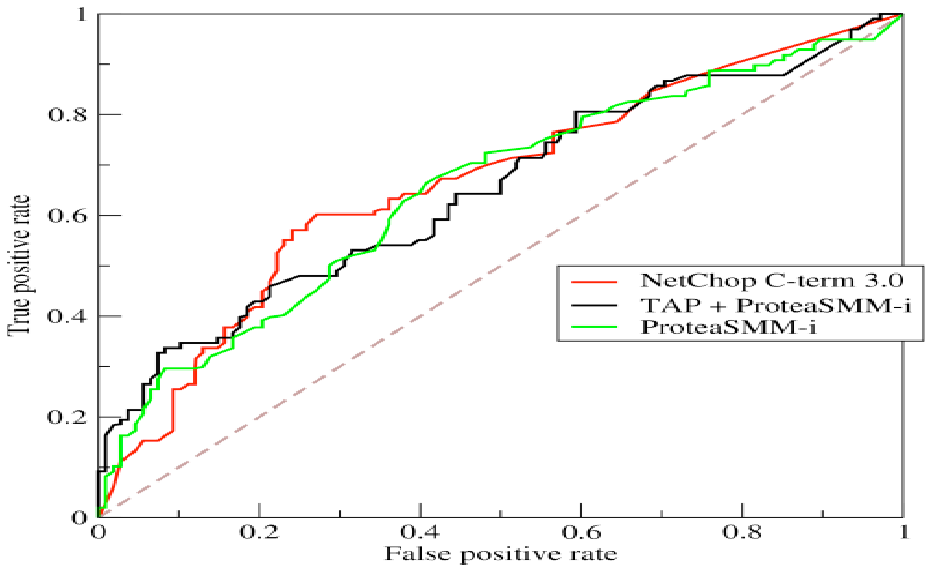
$$f_\beta = (1+\beta^2)\frac{\overset{\text{Precision}}{P}\overset{\text{Recall}}{R}}{(\beta^2 P) + R}$$

β is a non-negative real values

- We often use f1 measure.

ROC curves

- Receiver Operator Characteristic (ROC) curves plot TP vs FP rates



- Can be achieved by e.g. varying decision threshold of a classifier
- Area under the curve is often used as measure of goodness

Confusion matrix

- A visualization tool used to present the results attained by a learner.
- Easy to see if the system is commonly mislabeling one class as another.

Predicted True	**A**	**B**	**C**
A	5	3	0
B	2	3	1
C	0	2	11

Curse of Dimensionality

- As dimension D increases, the number of independent coefficients grows proportionally

$$y(x, \boldsymbol{w}) = w_0 + w_1 x^1 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

$$\Downarrow$$

$$y(x, \boldsymbol{w}) = w_0 + \sum_{i=1}^{D} w_i x_i + \sum_{i=1}^{D} \sum_{j=1}^{D} w_{ij} x_i x_j + \sum_{i=1}^{D} \sum_{j=1}^{D} \sum_{k=1}^{D} w_{ijk} x_i x_j x_k \ldots$$

- The amount of data needed to support the result often grows exponentially with the dimensionality.

Curse of Dimensionality

- Consider a sphere of radius $r = 1$ in a space of D dimensions, what is the fraction of the volume of the sphere that lies between radius $r = 1 - \epsilon$ and $r = 1$?

- The volume of a sphere of radius r in D dimensions must scale as r^D, we have

$$V_3(r) = \frac{4}{3}\pi r^3$$

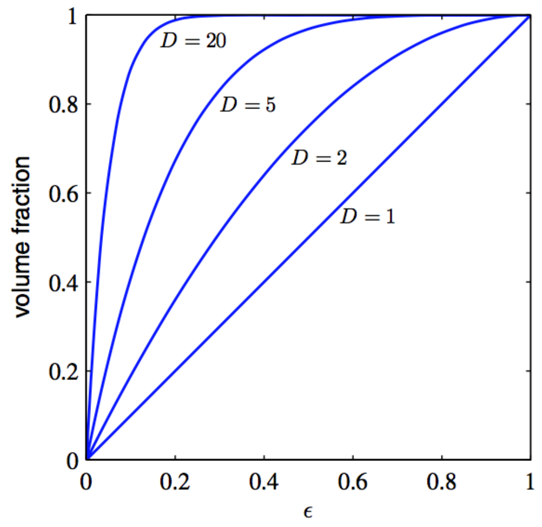
$$V_D(r) = K_D r^D$$

Curse of Dimensionality

- Where the constant K_D depends only on D, thus the required volume fraction is given by

$$\frac{V_D(1) - V_D(1-\epsilon)}{V_D(1)} = 1 - (1-\epsilon)^D$$

- For large D, the fraction *tend to 1* even for small values of ϵ.



Curse of Dimensionality

- For large D, the fraction *tend to 1* even for small values of ϵ.
- In space of high dimensionality, most of the volume of a sphere is concentrated in a thin shell near the surface.
- Distance functions losing their usefulness (for the nearest-neighbor criterion in feature-comparison algorithms, for example) in high dimensions.