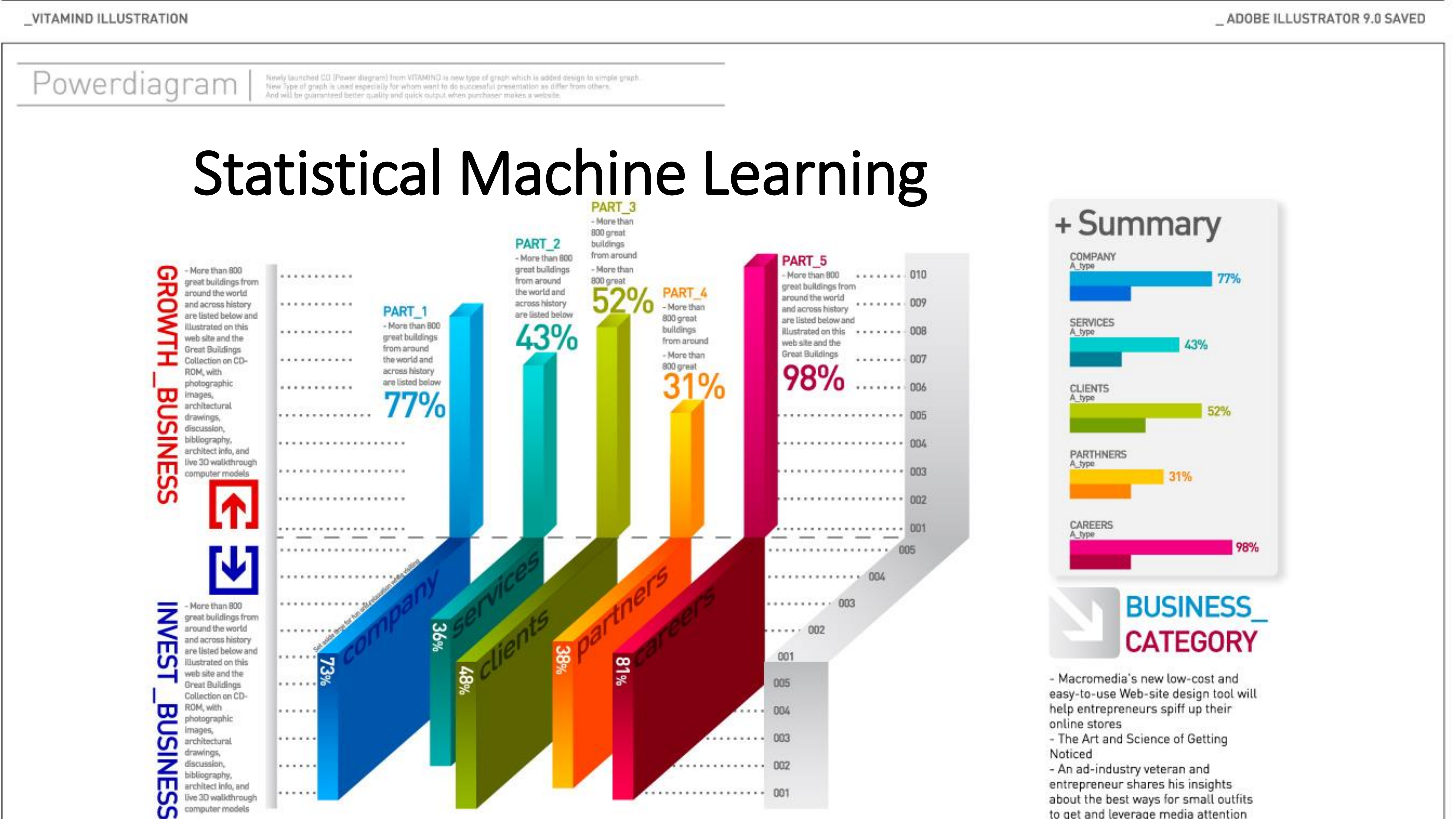
Statistical Machine Learning



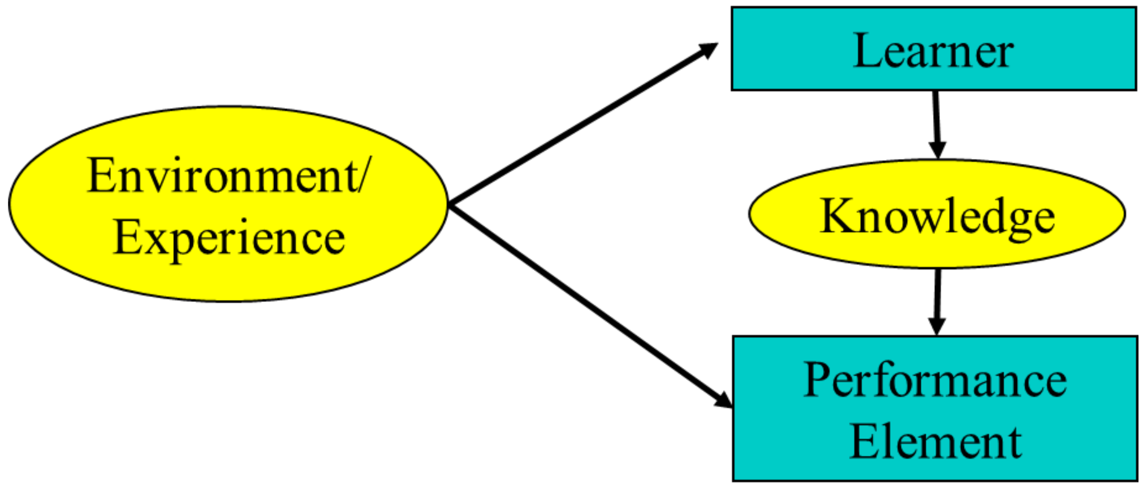
- More than 800 great buildings from around the world and across history are listed below and illustrated on this web site and the Great Buildings Collection on CD-ROM, with photographic images, architectural drawings, discussion, bibliography, architect info, and live 3D walkthrough computer models

- Macromedia's new low-cost and easy-to-use Web-site design tool will help entrepreneurs spiff up their online stores
- The Art and Science of Getting Noticed
- An ad-industry veteran and entrepreneur shares his insights about the best ways for small outfits to get and leverage media attention

Framework of Statistical Machine Learning

- Input: n i.i.d. training samples $(x_i, y_i) \in X \times Y, i = 1, \cdots, n$
- Target function: $f \in \mathcal{F}$
- Loss function: $L(f; x, y)$
- Expected Risk: $\int L(f; x, y) dP(x, y)$

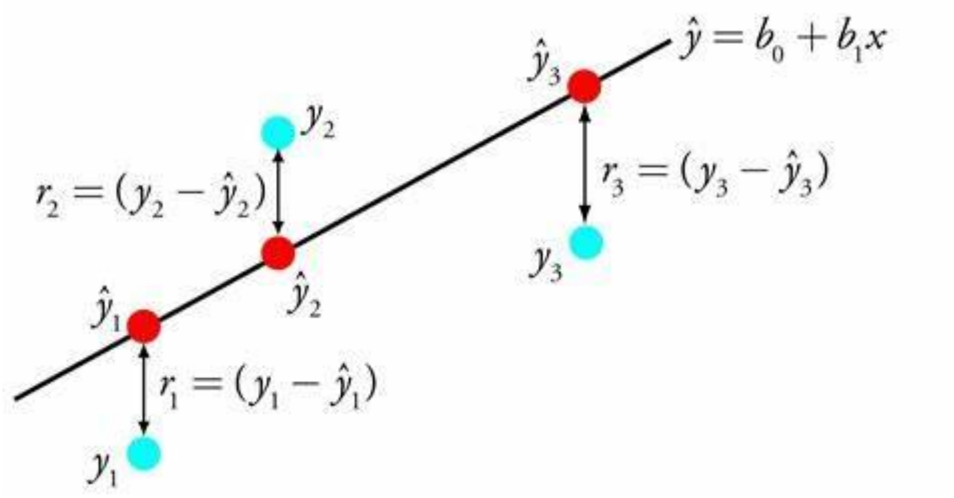
(1) Regression: Y is continuous
(2) Classification: Y is categorical
(3) Ranking: Y is ordinal



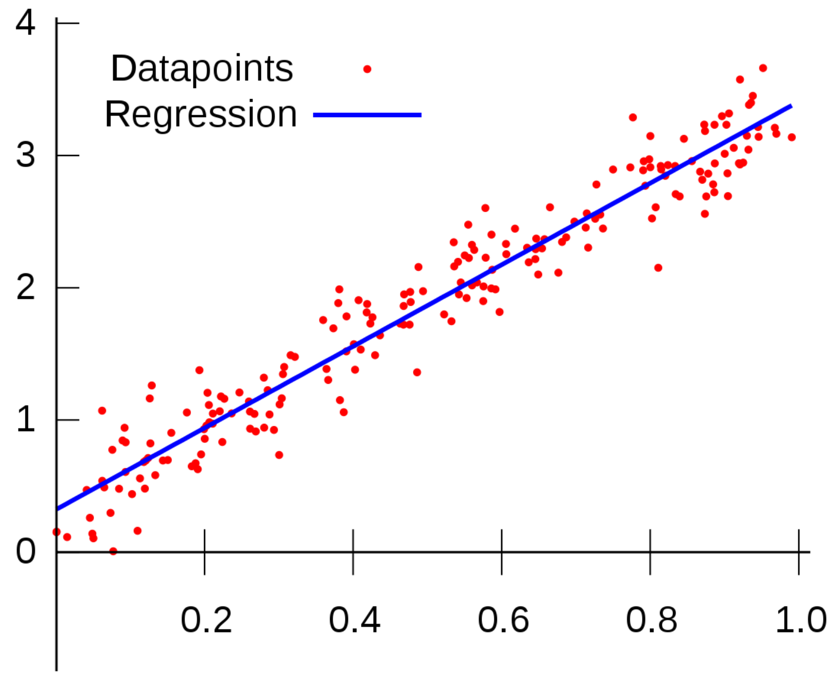
Regression Problem

- Input: n i.i.d. training samples $(x_i, y_i) \in X \times R, i = 1, \cdots, n$
- Target function: $f \in \mathcal{F}$
- Loss function: $L(f; x, y) = (f(x) - y)^2$
- Expected Risk: $\int (f(x) - y)^2 dP(x, y)$

(1) Linear Regression: F is Linear
(2) Generalized Linear: F is Non-Linear



$\hat{y} = b_0 + b_1 x$
$\hat{y}_3$
y_2
$r_2 = (y_2 - \hat{y}_2)$
$r_3 = (y_3 - \hat{y}_3)$
y_3
$\hat{y}_2$
$\hat{y}_1$
$r_1 = (y_1 - \hat{y}_1)$
y_1



Analysis of Optimal Function for Regression

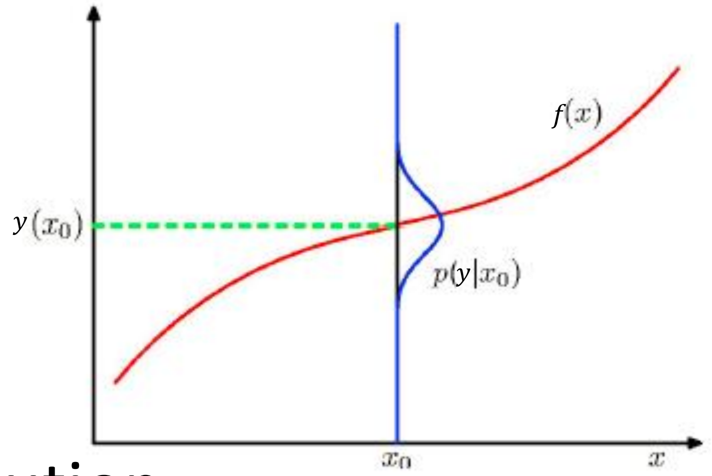
- Minimize Expected Loss $\int (f(x) - y)^2 dP(x,y)$

$$= \int (f(x) - y)^2 \, p(x,y) dx dy = \int Q(f(x), y) p(x) dx$$

- Where: $\boxed{Q(f(x), y) = f(x)^2 - 2E(y|x) f(x) + E(y^2|x)}$
- Take derivative and setting equal to zero yields a solution

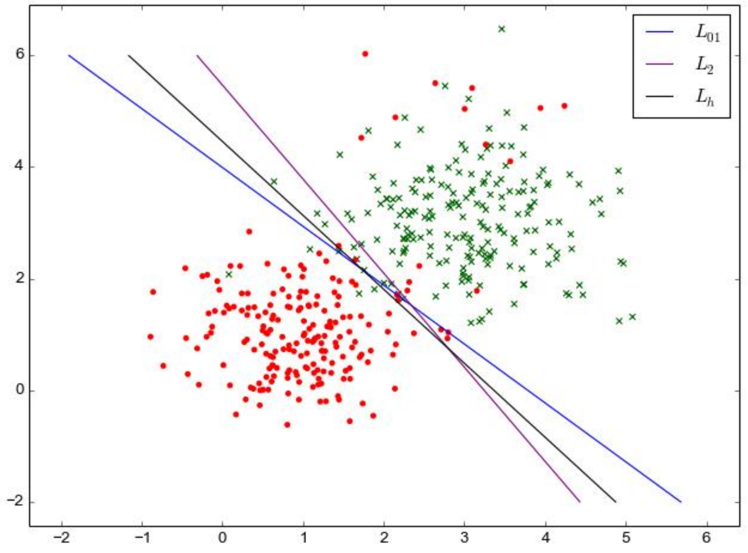
$$\boxed{f(x) = E(y|x) = \int y p(y|x) dy}$$

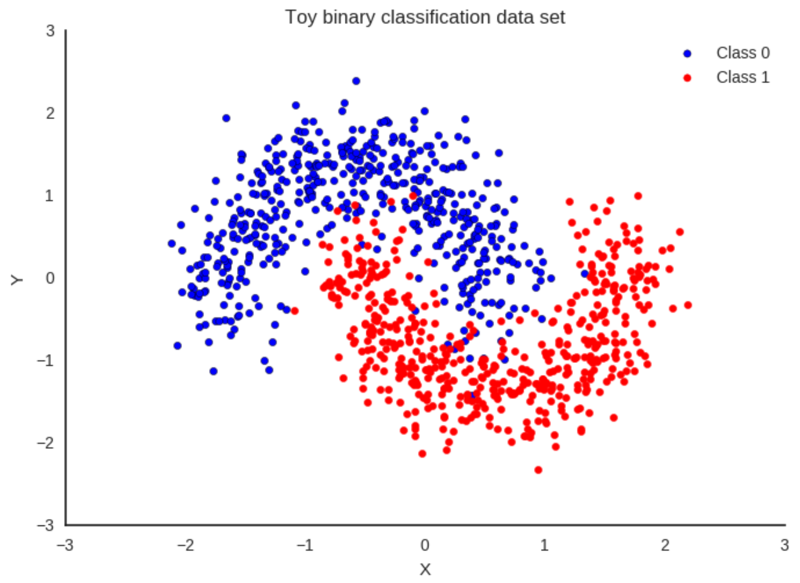
- Regression function, which minimize the expected squared loss, is given by the mean of the conditional distribution $p(y|x)$



Classification Problem

- Input: n i.i.d. training samples $(x_i, y_i) \in X \times C, i = 1, \cdots, n,$
- Target function: $f \in \mathcal{F}$
- Loss function:$L(f; x, y) = I_{\{f(x) \neq y\}}$
- Expected Risk:$\int I_{\{f(x) \neq y\}} dP(x, y) = P\{f(x) \neq y\}$





Analysis of Optimal Function for Classification

- Minimize Expected Loss $$\boxed{\int I_{\{f(x)\neq y\}} dP(x,y) = P\{f(x)\neq y\} = \sum_{f(x)\neq c_i} P(c_i|x)P(x)}$$
- Solution $\boxed{f(x) = \max_{c_i} P(c_i|x)}$
- Bayes Classifier, which minimizes the 0-1 loss, is chosen to be the class label with maximal conditional distribution $p(y|x)$
- Decision of Binary Classification:

$$\text{Choose} = \begin{cases} C=1 & \text{if } P(C=1|\mathbf{x}) > 0.5 \\ C=0 & \text{otherwise} \end{cases} \qquad \text{Choose} = \begin{cases} C=1 & \text{if } P(C=1|\mathbf{x}) > P(C=0|\mathbf{x}) \\ C=0 & \text{otherwise} \end{cases}$$

- Decision of Multi-Classification:

$$\text{Choose } C_i \text{ if } P(C_i|\mathbf{x}) = \max_k P(C_k|\mathbf{x})$$

Three Distinct Approaches: Generative Model

- First solve inference problem of joint distribution

$$p(x,y) = p(y)p(x|y)$$

- Then use Bayes theorem to determine conditional distribution $p(y|x)$

$$\boxed{p(y|x) = \frac{p(x,y)}{p(x)} = \frac{p(y)p(x|y)}{\int p(y)p(x|y)dy}}$$

- Determine the output accordingly

Three Distinct Approaches: Discriminative Model

- Directly solve inference problem to determine conditional distribution $p(y|x)$
- Determine the output accordingly

Three Distinct Approaches: Discriminant Functions

- Find a function $f(x)$ that maps each input directly to the output
  - e.g. Binary Classification, f is a binary valued function
  - $f = 1$ represents class C_1, $f = 0\ (-1)$ represents class C_2
- Probability play no direct role
  - No direct access to posterior probability
  - f is usually aimed to approximating the conditional distribution $p(y|x)$

- How to find a good discriminant function? Criteria?

ERM vs. SRM

Expected Risk Minimization

$$\int L(f;x,y)dP(x,y)$$

- Empirical Risk Minimization

$$\sum_{i=1}^{n} L(f;x_i,y_i)$$

- Structural Risk Minimization

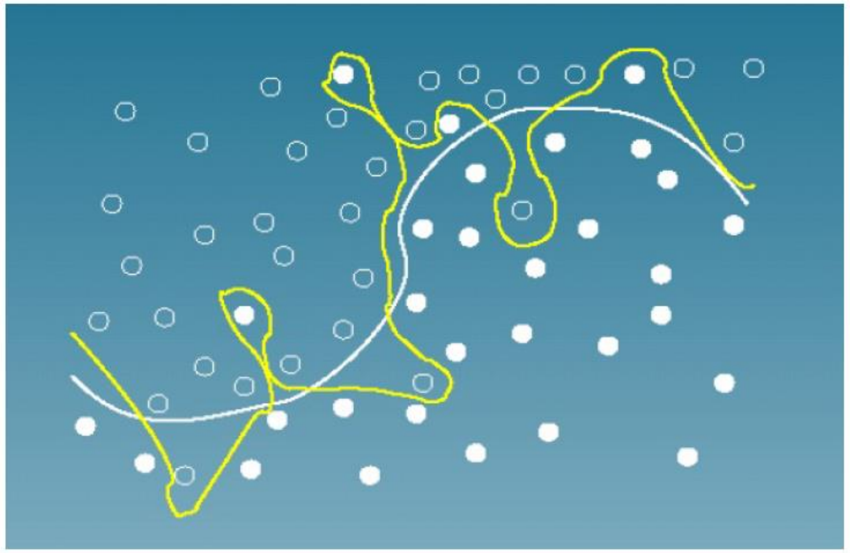
$$\sum_{i=1}^{n} L(f;x_i,y_i) + \lambda J(f)$$

Regularizer/
Penalty function

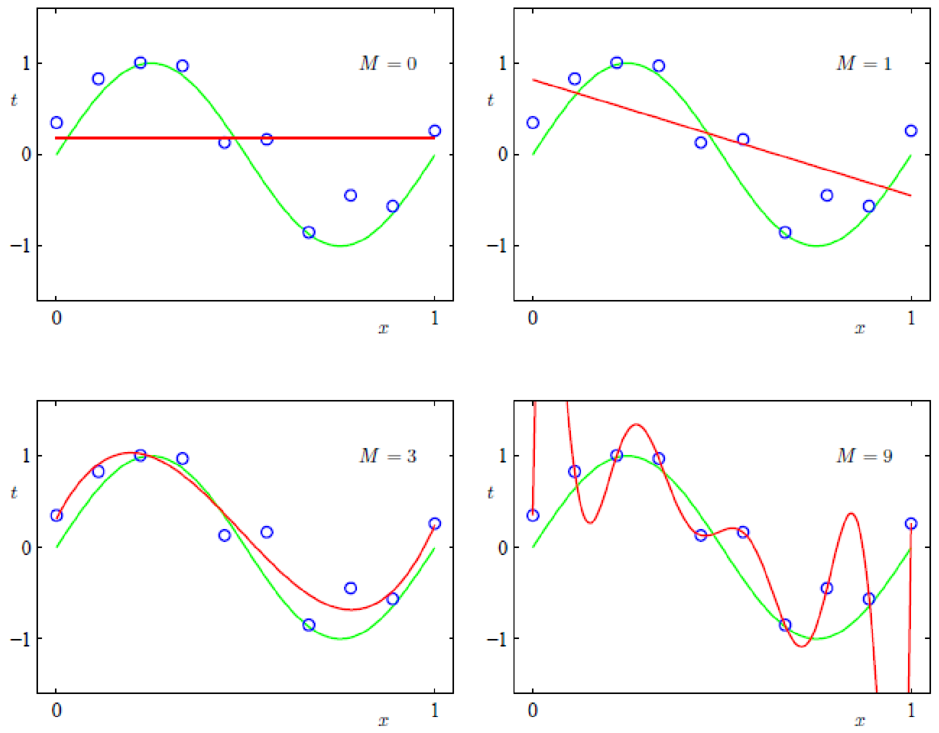
Overfitting!

Overfitting and Model Selection

Overfitting-Classification

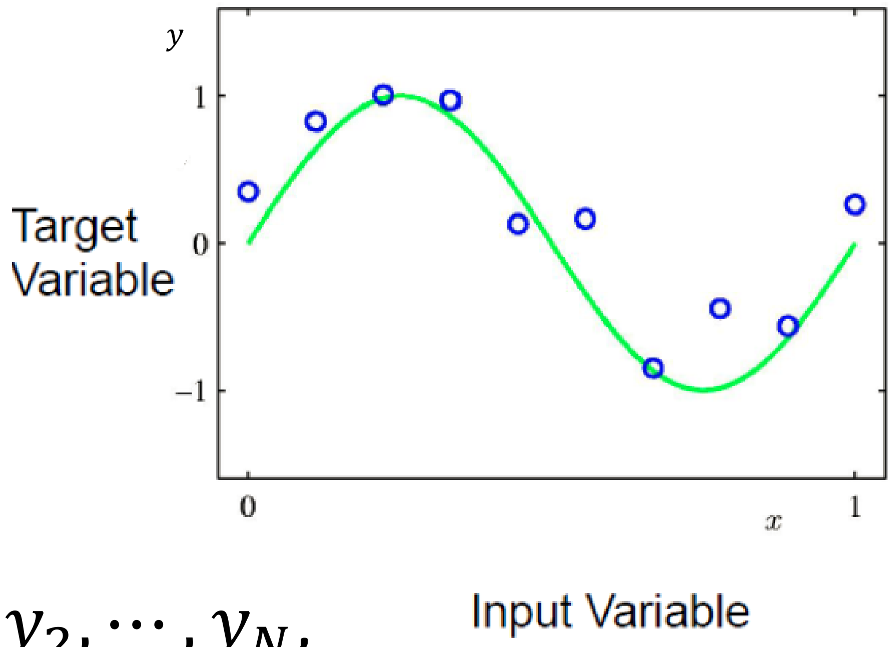


Overfitting-Regression



Case: Polynomial Curve Fitting

- Observe real-valued input variable x
- Use x to predict value of target variable y
- Synthetic data generated by $sin(2\pi x)$
- Random noise in target values



- N observations of x, i.e. $x_1, x_2, \cdots, x_N$ and $y_1, y_2, \cdots, y_N$,
- Data Generation:
  - N=10,
  - Uniformly from $\sin(2\pi x)$ by adding small Gaussian noise
- The goal is to exploit training data to find a prediction function f

Polynomial Curve Fitting

- Polynomial function

$$f(x;\vec{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{i=0}^{M} w_i x^i$$

- M is the order of the polynomial.
- How to choose the value of M?
- $w_0, \cdots w_M$ are denoted by a vector $\vec{w}$.
- Nonlinear function of x, linear function of $\vec{w}$.

ERM for Polynomial Curve Fitting

$$\hat{E}(f) = \sum_{i=1}^{N} L(f; x_i, y_i) = \sum_{i=1}^{N} (f(x_i; \vec{w}) - y_i)^2$$

- Solve by choosing value of $\vec{w}$ for which $\hat{E}(f)$ is as small as possible
- Error function is quadratic in coefficients $\vec{w}$

$$\hat{E}(f) = \sum_{i=1}^{N} \left(\sum_{j=0}^{M} w_j x_i^j - y_j \right)^2$$

ERM for Polynomial Curve Fitting

- Derivative with respect to coefficients will be linear in elements of $\vec{w}$

$$\frac{\partial \hat{E}(f)}{\partial w_m} = \sum_{i=1}^{N} 2\left(\sum_{j=0}^{M} w_j x_i^j - y_j\right) x_i^m \Rightarrow \sum_{i=1}^{N}\sum_{j=0}^{M} w_j x_i^{j+m} = \sum_{i=1}^{N} y_j x_i^m \qquad M+1 \text{ equations}$$

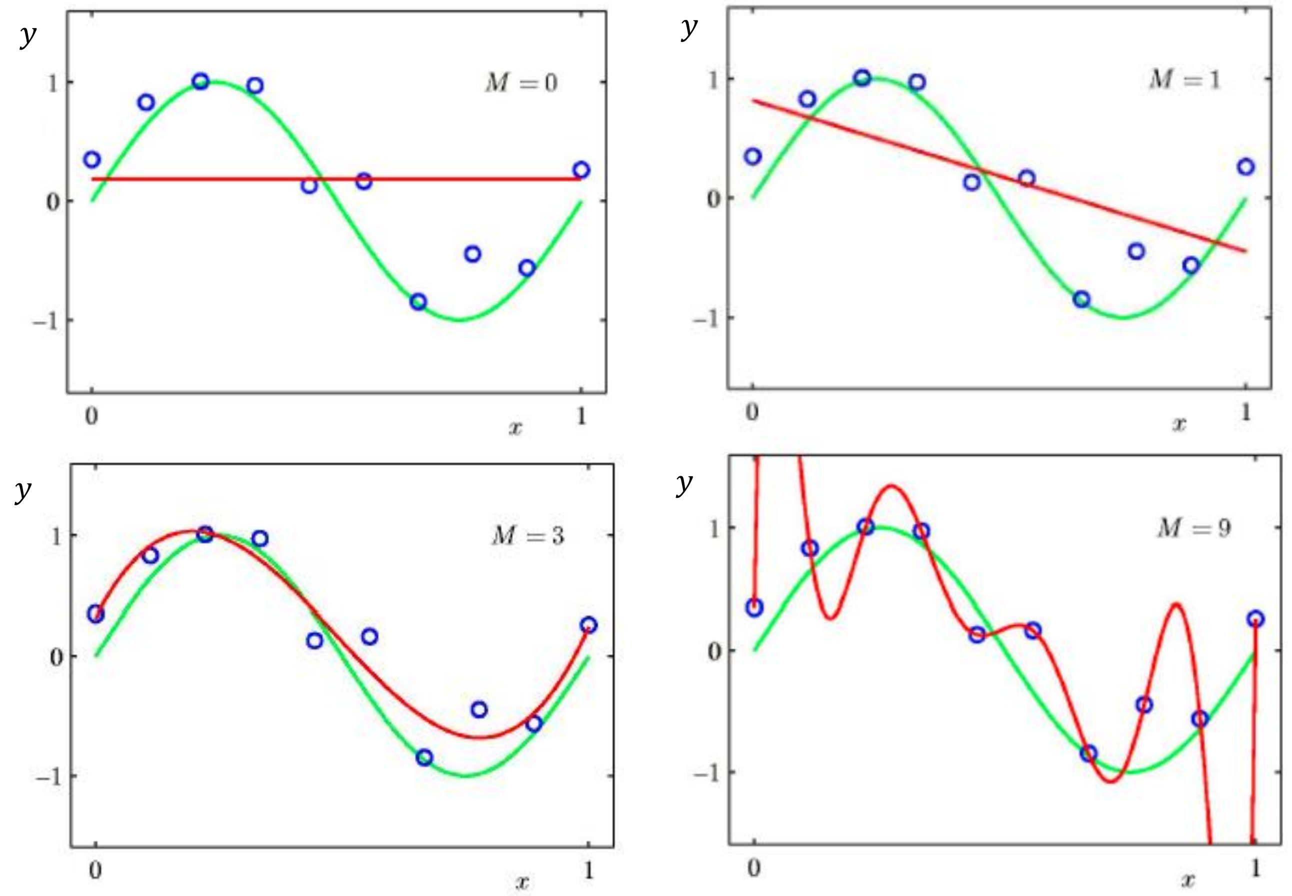
- Thus Empirical loss has a closed form solution $Aw = b \Rightarrow w = A^{-1}b$
- Unique minimum denoted as $\vec{w}^*$
- Resulting polynomial is $f(x, \vec{w}^*)$

Choosing Different M

- Model Selection
- Red line are best fits with

$M = 0,1,3,9$ and $N = 10$

Poor Representation of $\sin(2\pi x)$



Best Representation of $\sin(2\pi x)$

overfitting

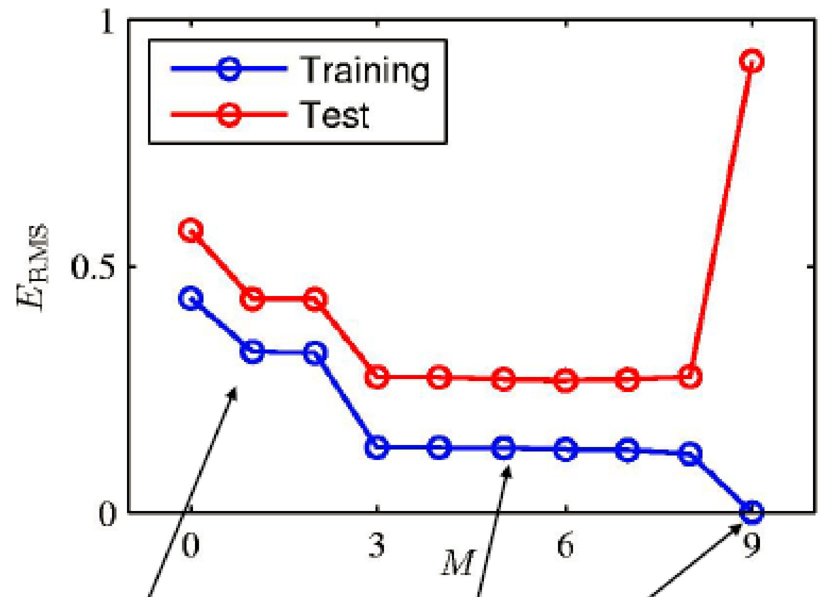
Results of Overfitting

- Consider separate test data of 100 points
- For each value of M evaluate:

$$\boxed{\hat{E}(f) = \sum_{i=1}^{N} L(f; x_i, y_i) = \sum_{i=1}^{N} (f(x_i; \vec{w}^*) - y_i)^2}$$

for training and testing data

- Using RMS error $\hat{E}_{RMS}(f) = \sqrt{\hat{E}(f)/N}$
  - Devision by N allow different size of N to be compared
  - Squared root ensures it is measured in the same units as y



Poor due to Inflexible polynomials

Small Error

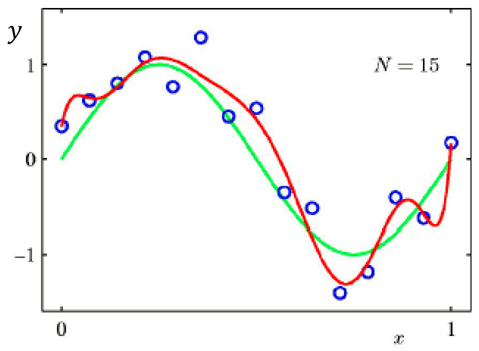
M=9 means ten degrees of freedom. Tuned exactly to 10 training points (wild oscillations in polynomial)

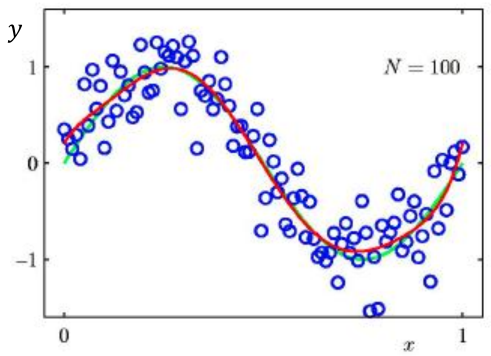
Training vs. Testing

- Training Data: $S = \{(x_1, y_1), \cdots, (x_n, y_n)\}$
- Our goal of Machine Learning:
  - Learn a function with training data
  - Achieve a good result with any future observation $(x, y) \sim P$
- Expected risk Minimization $\int L(f; x, y) dP(x, y)$
- Testing Data: $T = \{(x_1', y_1'), \cdots, (x_n', y_n')\}$, which are usually different from training data.

Goal: Predict Well on Testing Data!

How to Predict Well on Testing Data?





- Two aspects:
  - Model fits training data well
    - Requires a complex model
  - Model has some capacity to accommodate different behaviors of testing data
    - Requires a stable model (less complex)
- Model complexity vs. size of data set
  - N=15,100
  - For a given model complexity, overfitting problem is less severe as size of data set increases
  - Larger the data set, the more complex we can afford to fit the data
  - Data should no less than 5 to 10 times adaptive parameters in model

Regularization

$$\sum_{i=1}^{n} L(f; x_i, y_i) + \lambda J(f)$$

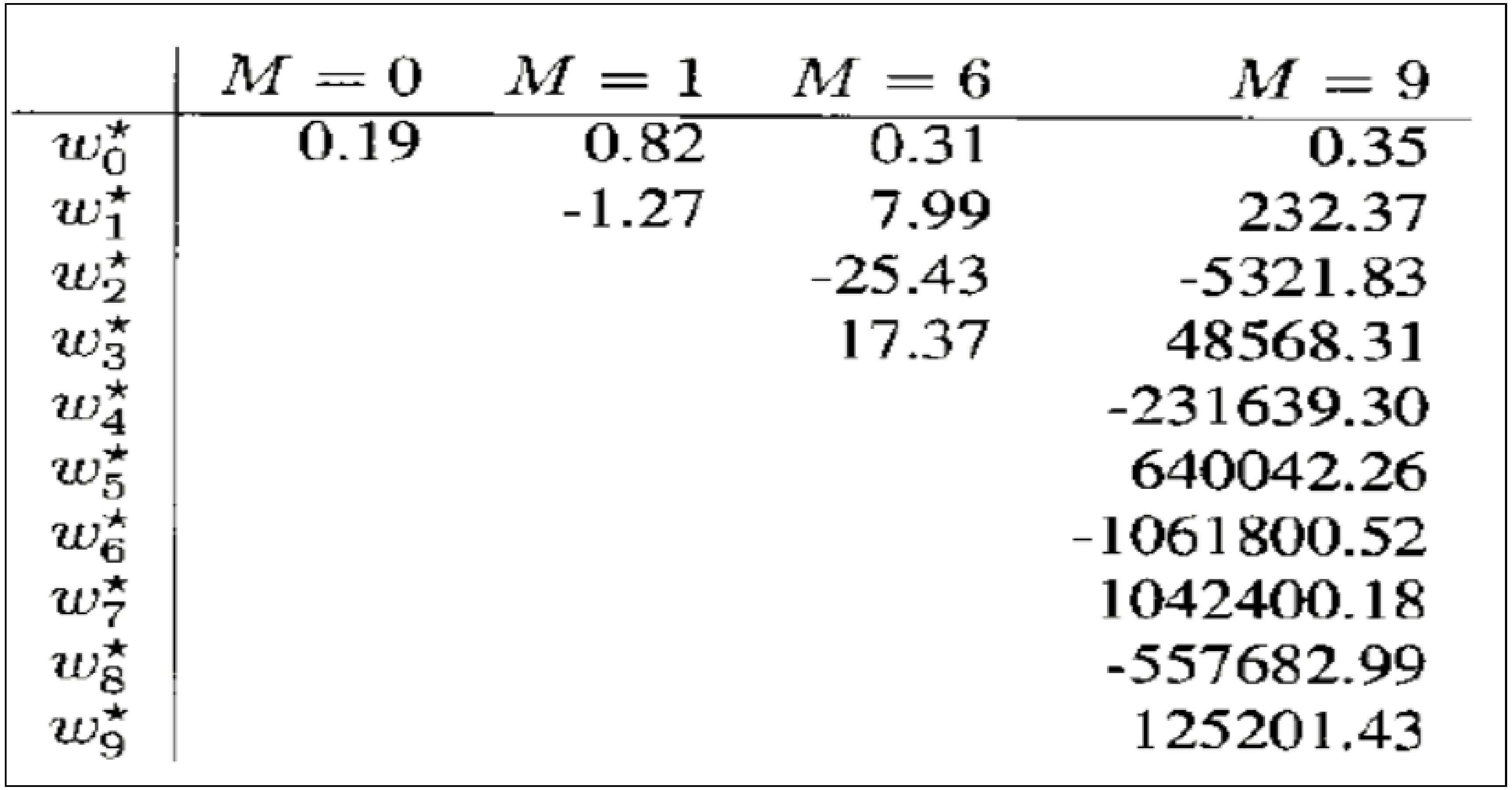
- Add a penalty term to the original empirical loss to discourage complex model
- L2 norm/Ridge regression

$$\|\vec{w}\|_p{}^p$$

$$\boxed{\hat{E}(f) = \sum_{i=1}^{N} L(f; x_i, y_i) + \lambda J(f) = \sum_{i=1}^{N} (f(x_i; \vec{w}) - y_i)^2 + \lambda \|\vec{w}\|_2{}^2}$$

  - where $\|\vec{w}\|^2 = \vec{w}^T \vec{w} = w_0^2 + w_1^2 + \cdots + w_M^2$
  - λ determines relative importance of regularization term to error term
  - Small λ corresponds to large complexity
  - Can be minimized exactly in closed form, known as shrinkage in statistics, weight decay in neural networks

Values of Coefficients w^* for different M



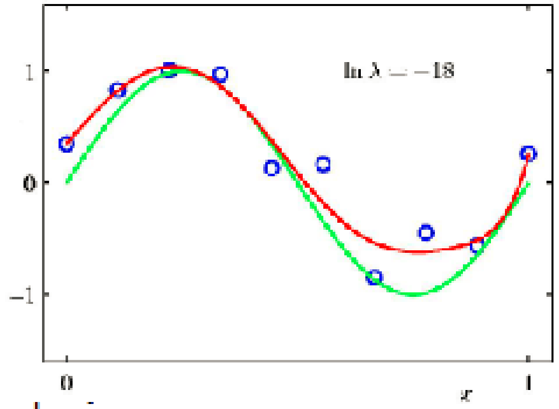
	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

As M increases magnitude of coefficients increases
At M=9 finely tuned to random noise in target values

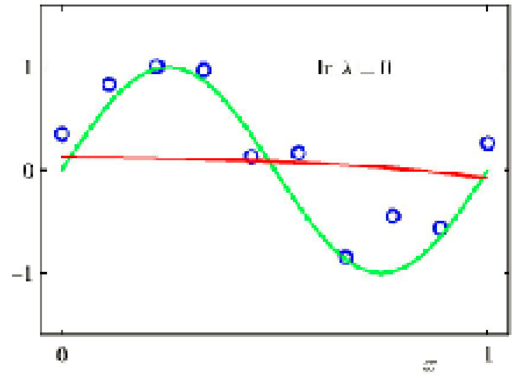
Effects of Regularizer

- $M = 9$ polynomial using regularized error function

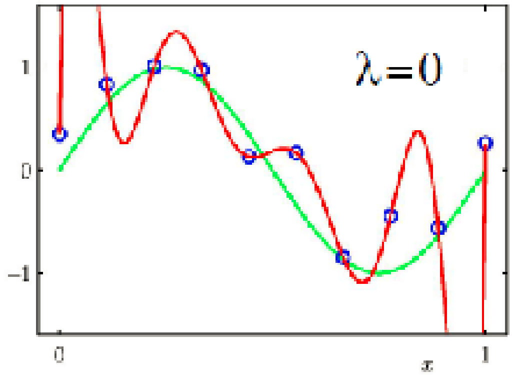
Optimal



Large Regularizer

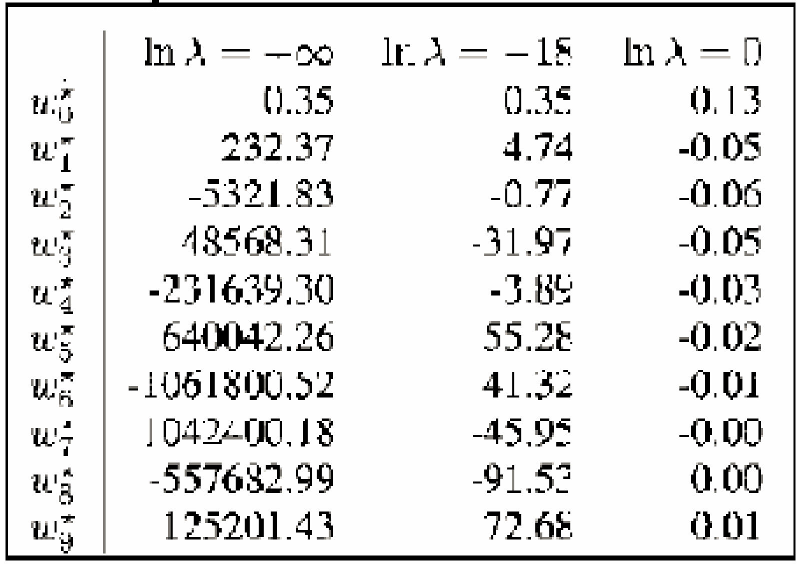


No Regularizer



No Regularizer $\lambda=0$

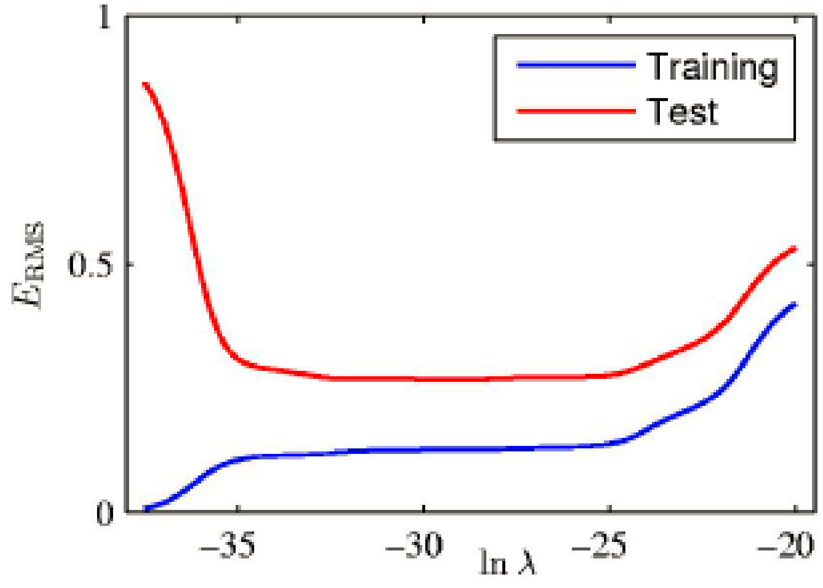
Large Regularizer $\lambda=1$



	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	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

Impact of Regularization on Error

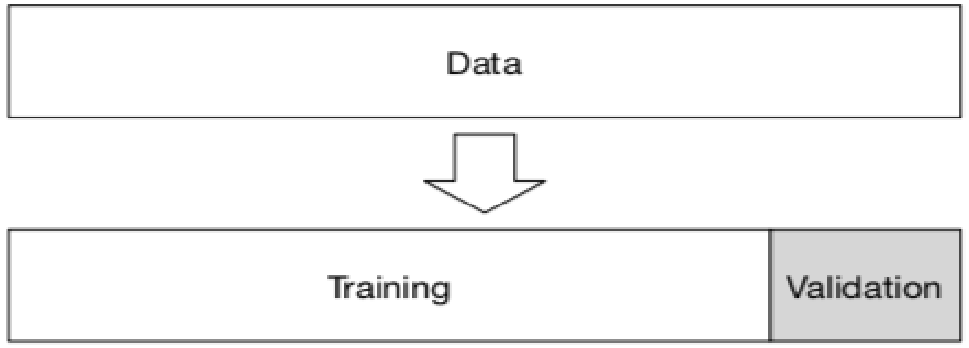
- λ controls the complexity of the model and hence degree of overfitting
  - Analogous to choice of M
- Suggested Approach: Validation
  - Training set
    - To determine coefficients $\vec{w}$
    - For different values of (M or λ)
  - Validation set
    - To optimize model complexity (M or λ)
    - Cross validation
      - Hold out, k-fold, leave-one-out



M=9 polynomial

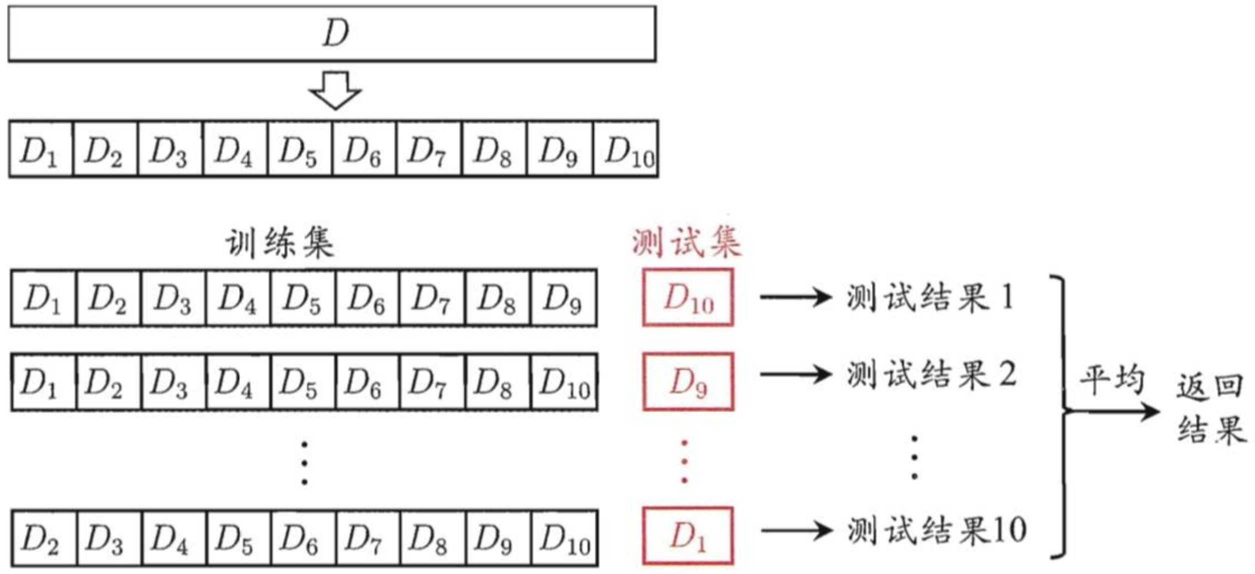
Validation: Hold Out

- Random split all data into two subsets: training set and validation set.
- Train machine learning model on training set.
- Pick model with lowest error on validation set.
- The size of the training set is usually 2/3 ~ 4/5 of all data.



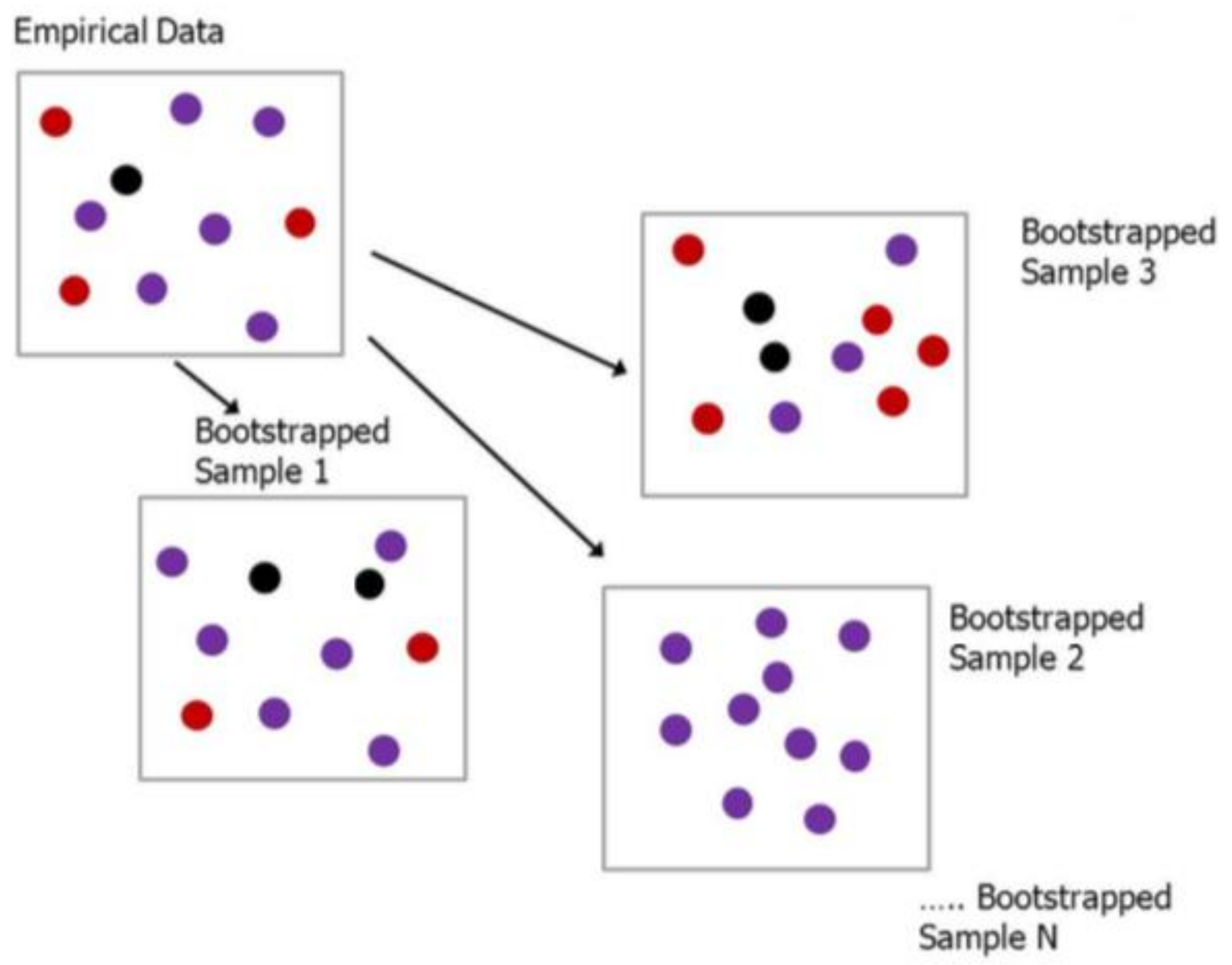
Validation: k-fold

- Create K-fold partition of the dataset.
- From K hold-out predictions, each time using one partition as validation and rest $K-1$ as training datasets.
- Final predictor is average/majority vote over the K hold-out estimates.
- K=10


D
D_1 D_2 D_3 D_4 D_5 D_6 D_7 D_8 D_9 D_{10}
训练集
测试集
测试结果 1
测试结果 2
测试结果10
平均
返回结果

Validation: Bootstrap

- Randomly draw datasets with replacement from training data, each sample the same size as the original training set.



Validation: Bootstrap

- Given dataset D containing n training examples, create D' by drawing n examples at random with replacement from D.
  - A particular training data has a probability of $1 - \frac{1}{n}$ of not being picked.
  - Thus the limit of probability that a training data is never sampled is:

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^n \mapsto \frac{1}{e} \approx 0.368$$

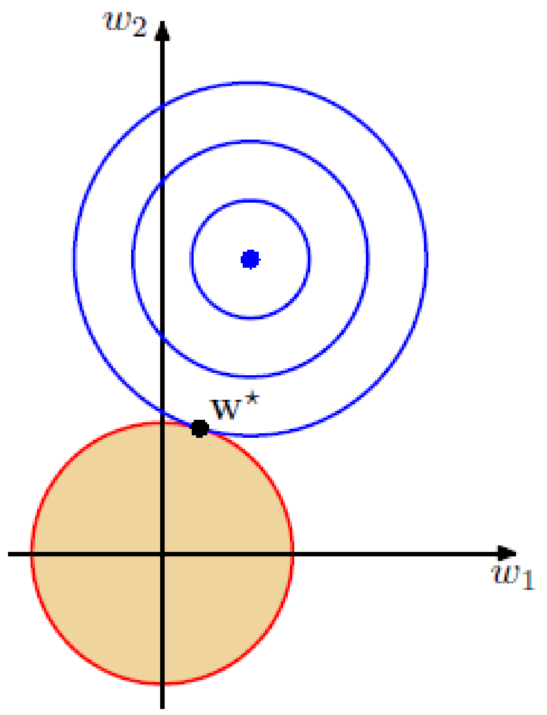
  - This means the training data will contain approximately 63.2% of the instances.
  - The remaining $D \backslash D'$ is used as the test dataset.

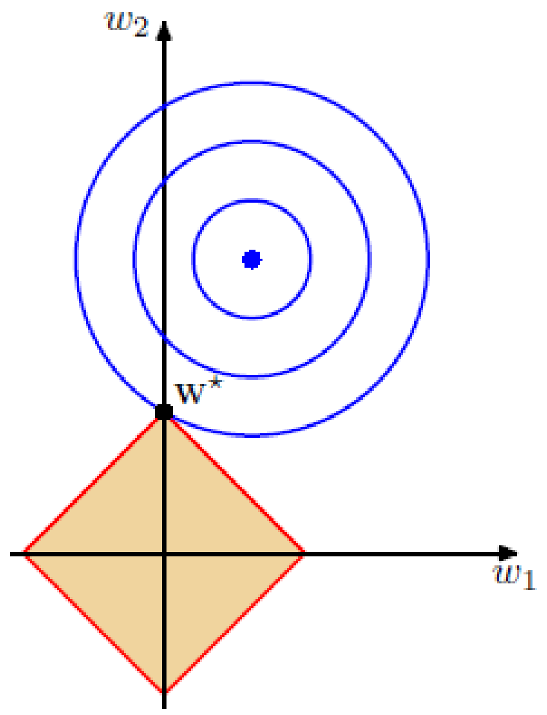
Training-Validation

- Since we split the data set into training set and validation set to do the model selection. After completing model selection, we should retrain the model on the whole data set.

- The validation set is only for parameter tuning.
- Evaluate the generalization ability of a model on the other test set.

The effect of L1 and L2 norm





- L2: using the l2 norm pulls directly towards the origin
- Lasso: using the l2 norm pulls towards the coordinate axes, i.e., it tries to set some of the coordinates to 0.

Bias-Variance Decomposition:

$$expected\ loss = bias^2 + variance + noise$$

- A frequentist viewpoints of the model complexity issue
- Denote:
  - $f_D(x)$ as the prediction function trained on data set D
  - $h(x)$ as the optimal function
    - For regression, $h(x) = E(y|x)$
- Expected loss of regression:

$$E(f_D) = E_D \iint (f_D(x) - y)^2 p(x,y)dxdy$$

$$= E_D\{\int \left(f_D(x) - h(x)\right)^2 p(x)dx + \iint (h(x) - y)^2 p(x,y)dxdy\}$$

$$= \boxed{\left(E_D(f_D(x)) - h(x)\right)^2} + \boxed{E_D\left(f_D(x) - E_D(f_D(x))\right)^2} + \boxed{\iint (h(x) - y)^2 p(x,y)dxdy}$$

Difference between expected value and optimal value

Bias^2

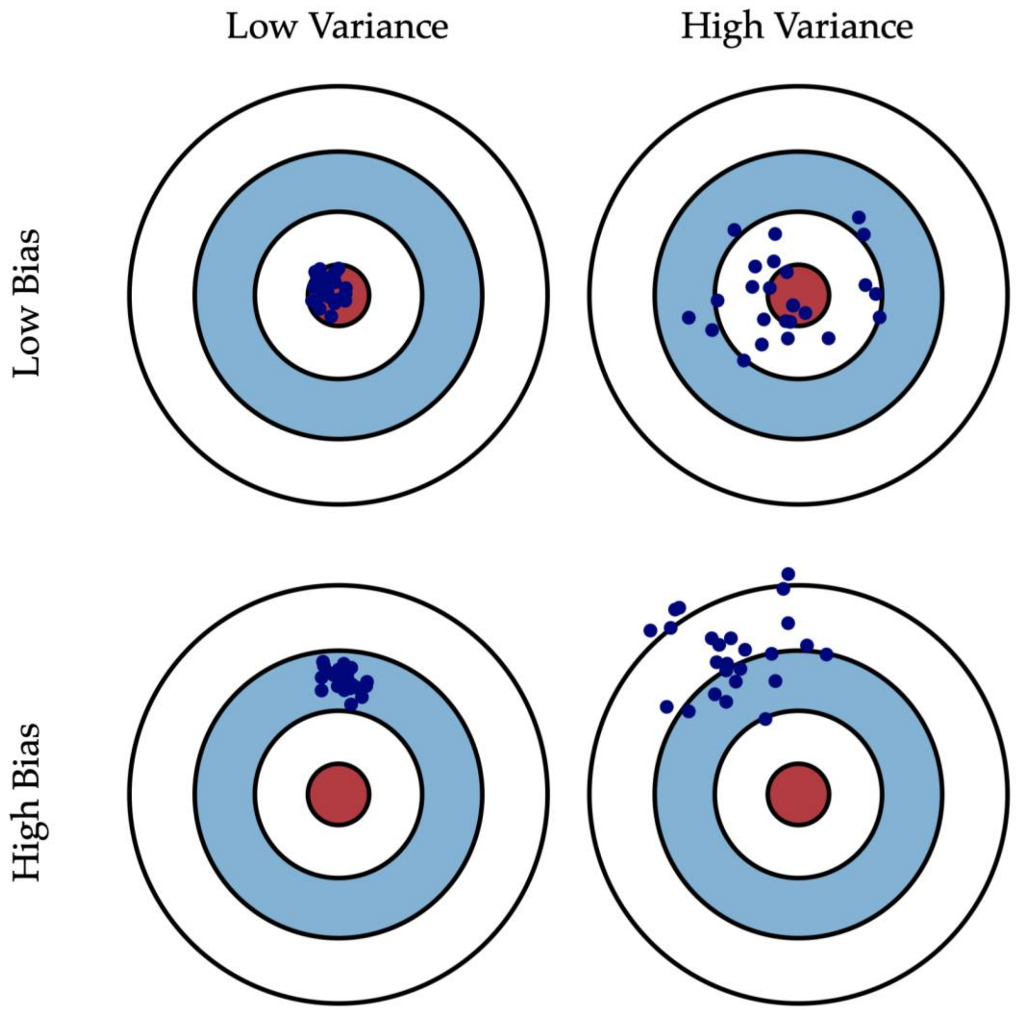
Variance

Variance between different training data

Noise

Independent of f

Bias-Variance Illustration



Example on Bias-Variance Decomposition

- $h(x) = \sin(2\pi x)$
- 100 independent data sets $D_i, i = 1, \cdots, 100$, each containing $N = 25$ data points, generated from $\sin(2\pi x)$
- Model:
  - Linear regression with 24 Gaussian basis function ($M = 25$):

$$f(x) = w_0 + \sum_{i=1}^{24} w_j \phi_j(x), \phi_j(x) = \exp\left(-\left(x - \mu_j\right)^2 / 2s\right),$$

  - Regularized least square error:

$$\sum_{i=1}^{N} (f(x_i) - y_i)^2 + \lambda \|\vec{w}\|_2^2$$

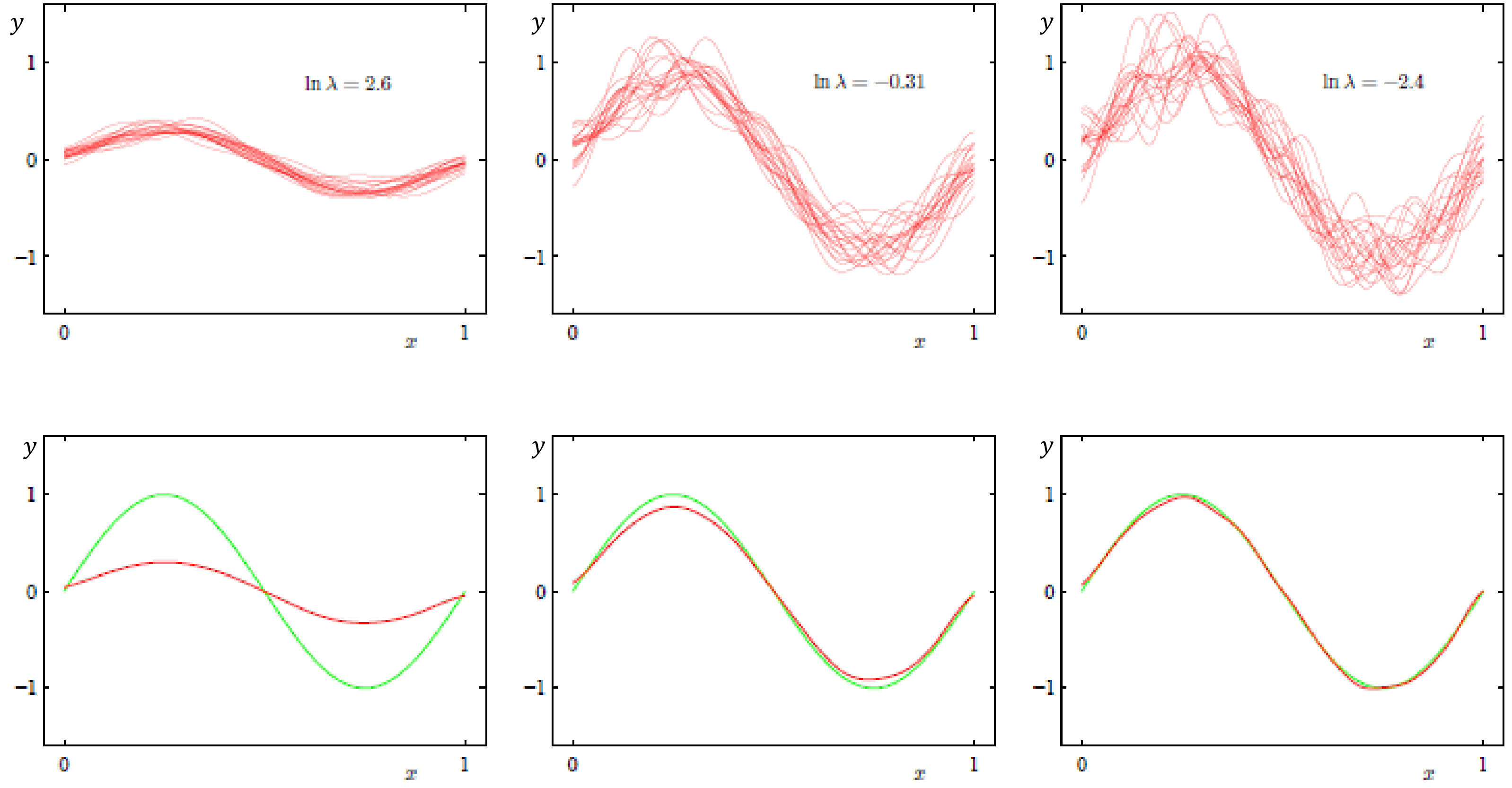
Example on Bias-Variance Decomposition

- Illustration of the dependence of bias and variance on model complexity, governed by regularization parameter λ

Low Variance

Simple Model

High Bias

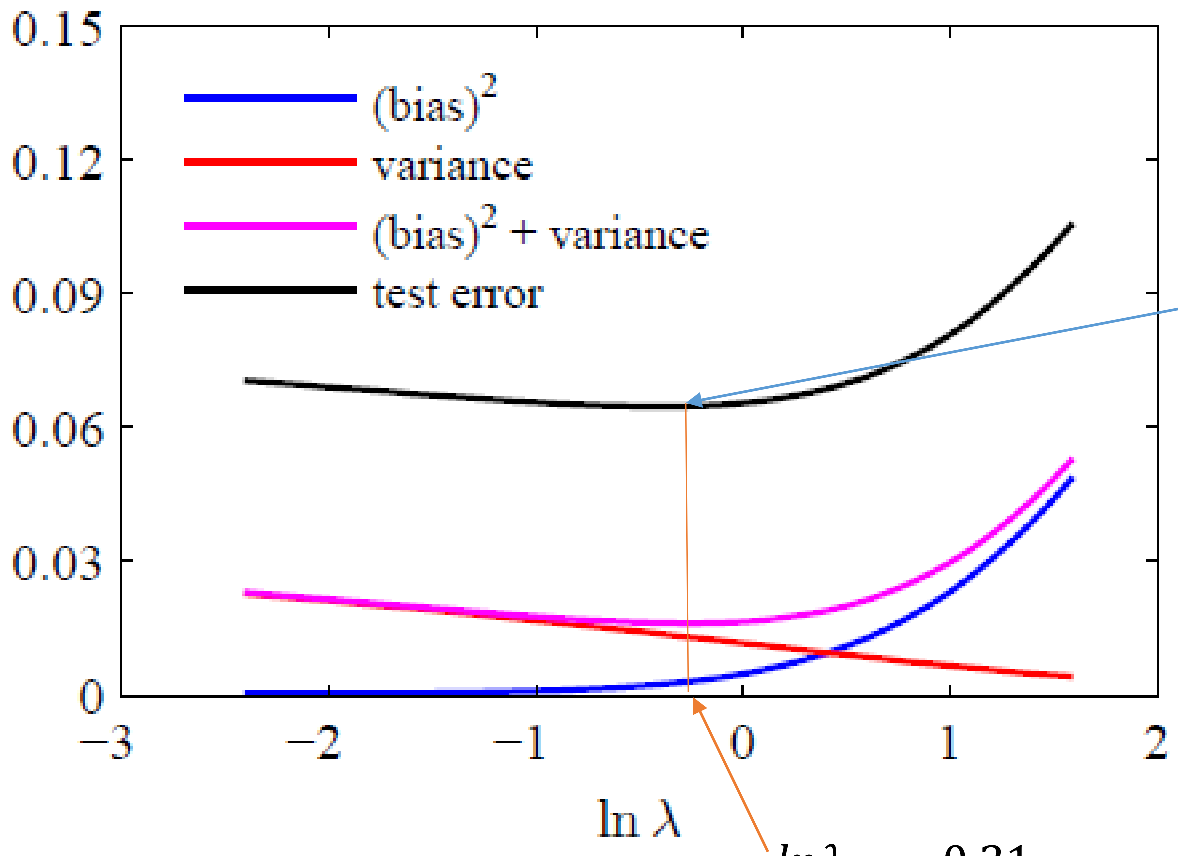


High Variance

Complex Model

Low Bias

Bias and Variance vs. λ



Test error minimum occurs close to minimum of $bias^2 + variance$

$ln\lambda = -0.31$

Small values of λ allow model to become finely tuned to noise leading to large variance

Large values of λ pull weight parameters to zero leading to large bias

Bias-Variance Tradeoff

- A good insight into model complexity issue:
  - Very flexible models having low bias and high variance.
  - Relatively rigid models having high bias and low variance.
  - The model with the optimal predictive capacity is the one that leads to the best balance between bias and variance.
- Bias-Variance decomposition has limited practical value
  - Bias and variance cannot be computed since it relies on knowing the true distribution of x and y
  - Bias-variance decomposition is based on averages with respect to ensembles of data sets, whereas in practice we have only the single observed data set.

Statistical Learning Theory

- Generalization Analysis for binary classification
  - Generalization Error:

$$E(f) = \iint I_{\{f(x)\neq y\}} p(x,y)dxdy$$

  - Training Error:

$$\hat{E}(f) = \sum_{i=1}^{N} I_{\{f(x_i \neq y_i)\}}$$

- ERM selects $\hat{f}$ to be the hypothesis with the smallest training error:

$$\hat{f} = \arg\min_{\underset{\xrightarrow{\text{Hypothesis class}}}{f\in\mathcal{F}}} \hat{E}(f)$$

- How about the generalization error of $\hat{f}$? Relations between training error and generalization error?

Generalization Bound

Theorem

Given $|\mathcal{F}| = k, N$ and δ fixed, with probability at least $1 - \delta$, we have that

$$E(\hat{f}) \leq \min_{f \in \mathcal{F}} \hat{E}(f) + 2\sqrt{\frac{\log 2\mathrm{k}/\delta}{2N}}$$

- Suppose we switch from $\mathcal{F}$ to some larger hypothesis class $\mathcal{F}' \supseteq \mathcal{F}$, then
  - The first term decreases. The “bias” decreases.
  - The second term increases (with large k). The “variance” increases.
- Infinite case: VC dimension, growth function, shatter function……

Homework

- Overfitting Example: Polynomial Curve Fitting

풍성한 마음이 {秋} 머무르는 계절
이번 풍요로운 추석 고마운 분들과
웃음 풍년 맞이하시기 바랍니다.
더도 덜도 말고 한가위만 같아라.

Q&A

lanyanyan@ict.ac.cn