Supervised Learning : Linear Regression and Logistic Regression

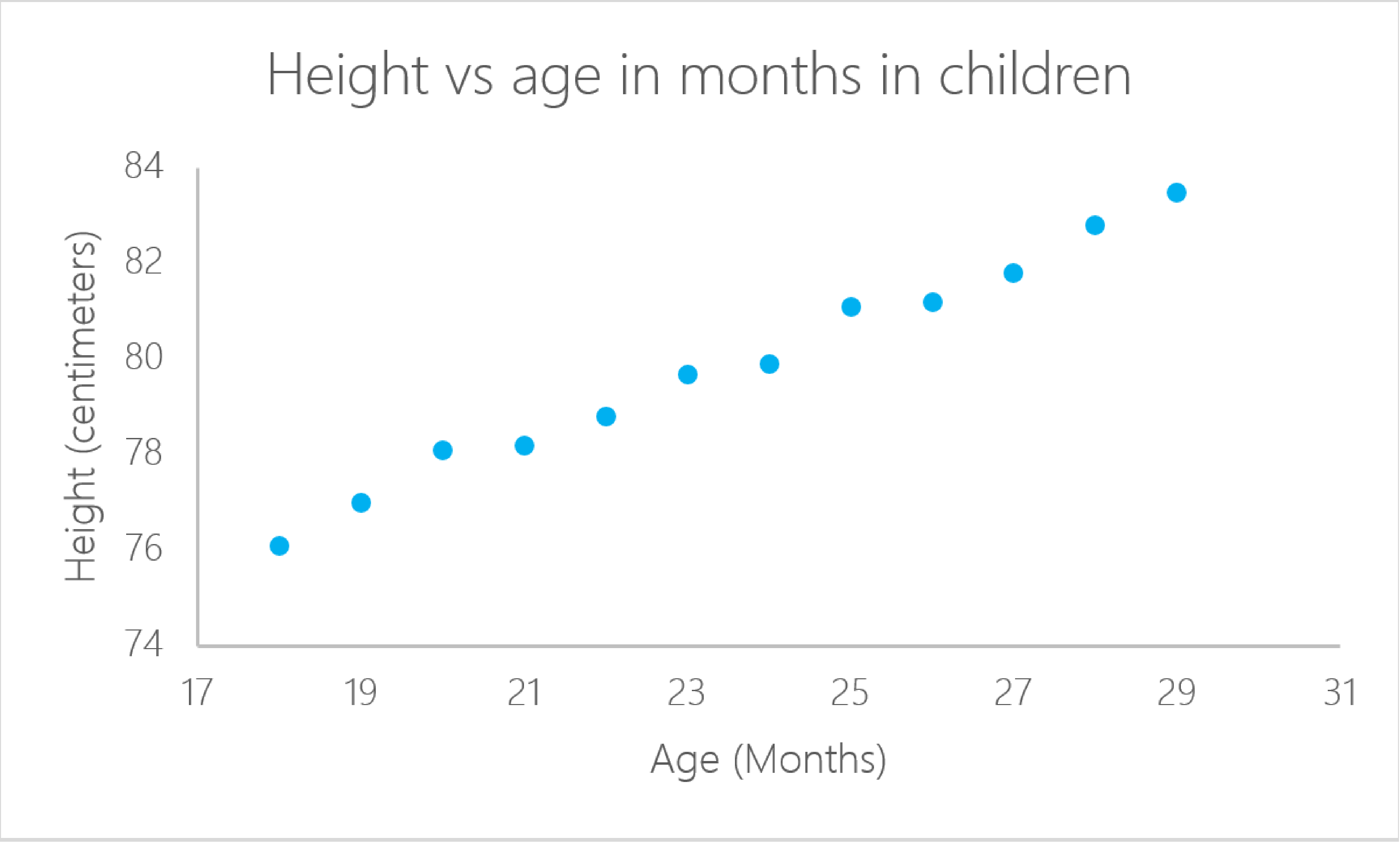
Dr. Srijith P K

Computer Science and Engineering

IIT Hyderabad

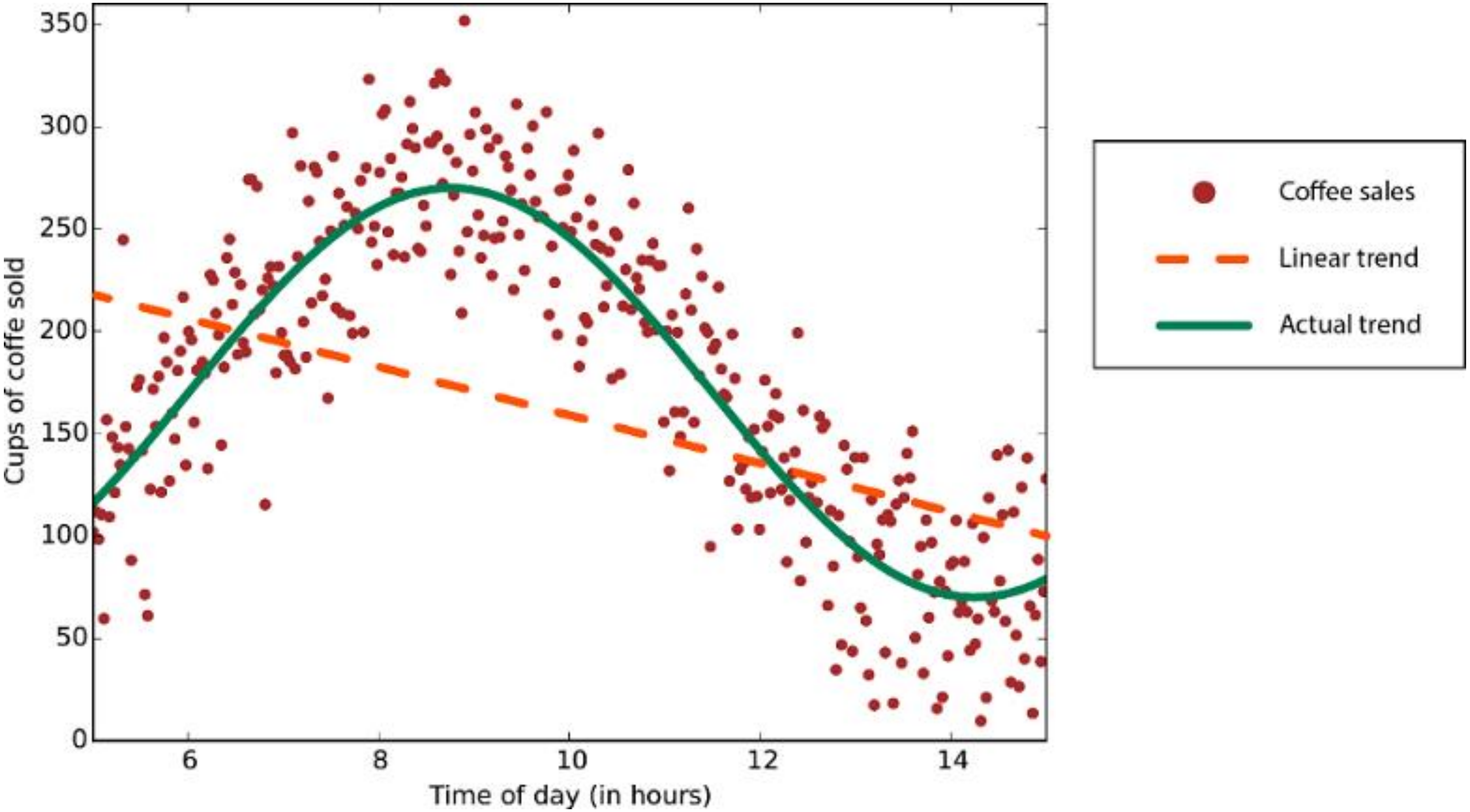
आई आई टी हैदराबाद
IIT Hyderabad

Supervised learning



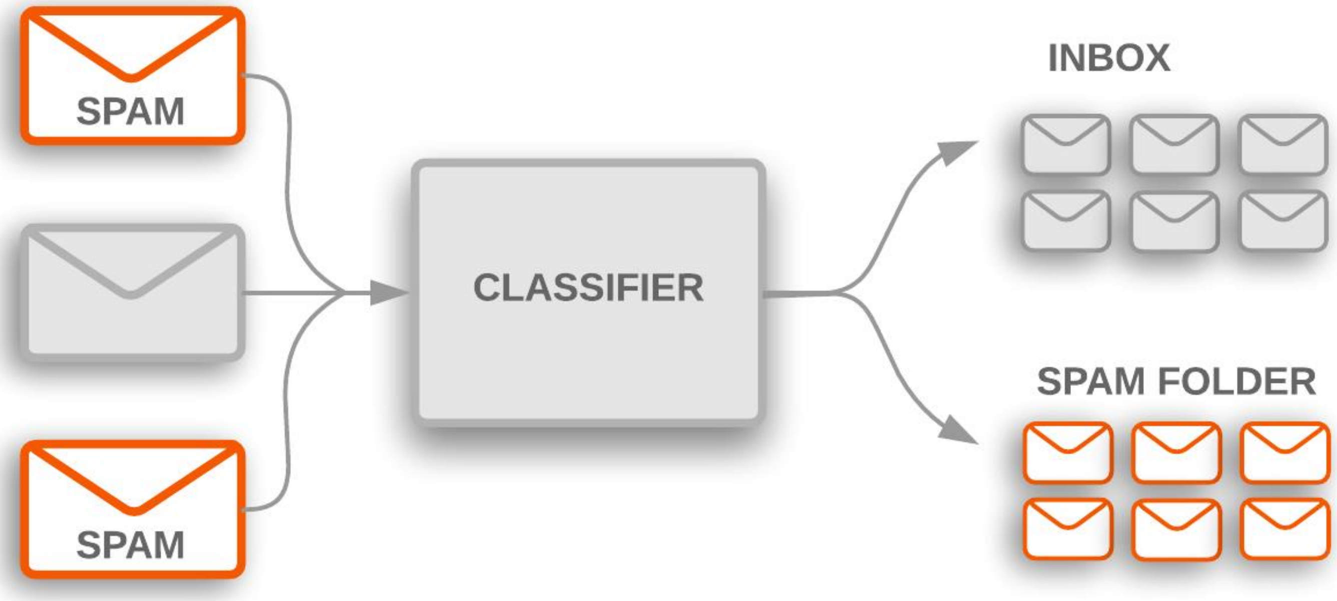
https://www.datacamp.com/community/tutorials/linear-regression-R

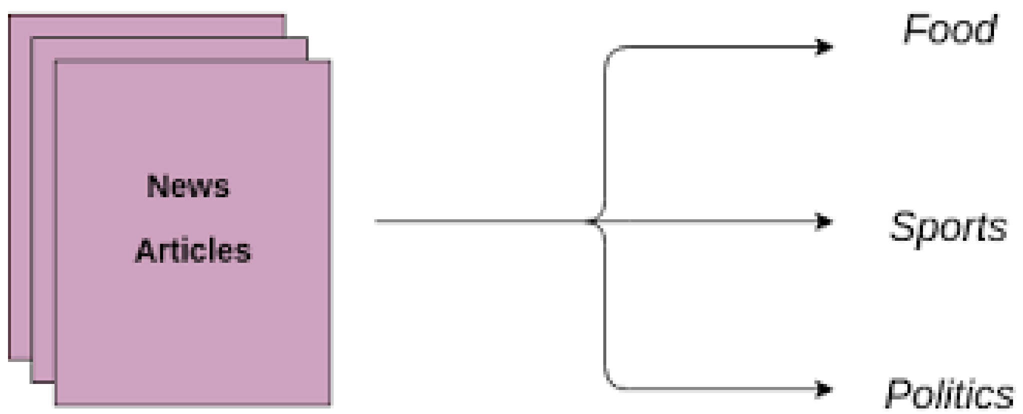
Supervised learning

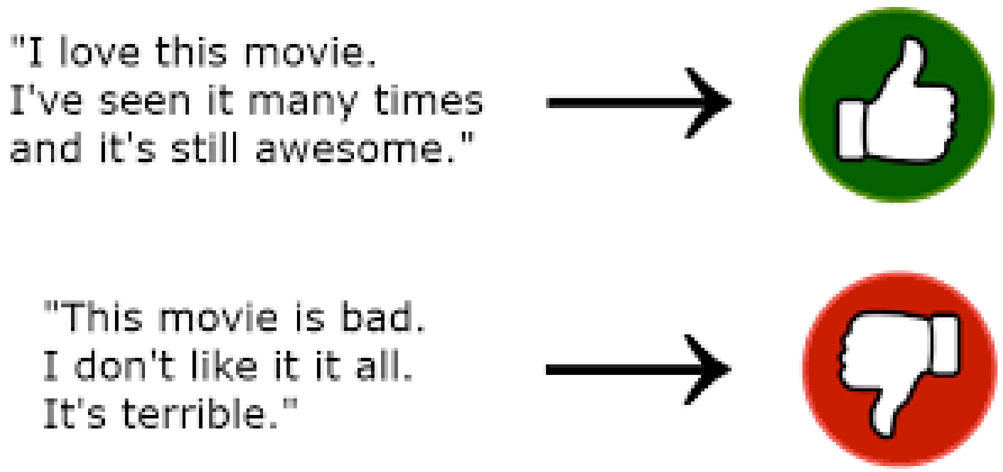


https://medium.com/@heyozramos/regression-vs-classification-86d73c281c5e

Supervised Learning






"I love this movie.
I've seen it many times
and it's still awesome."

"This movie is bad.
I don't like it it all.
It's terrible."

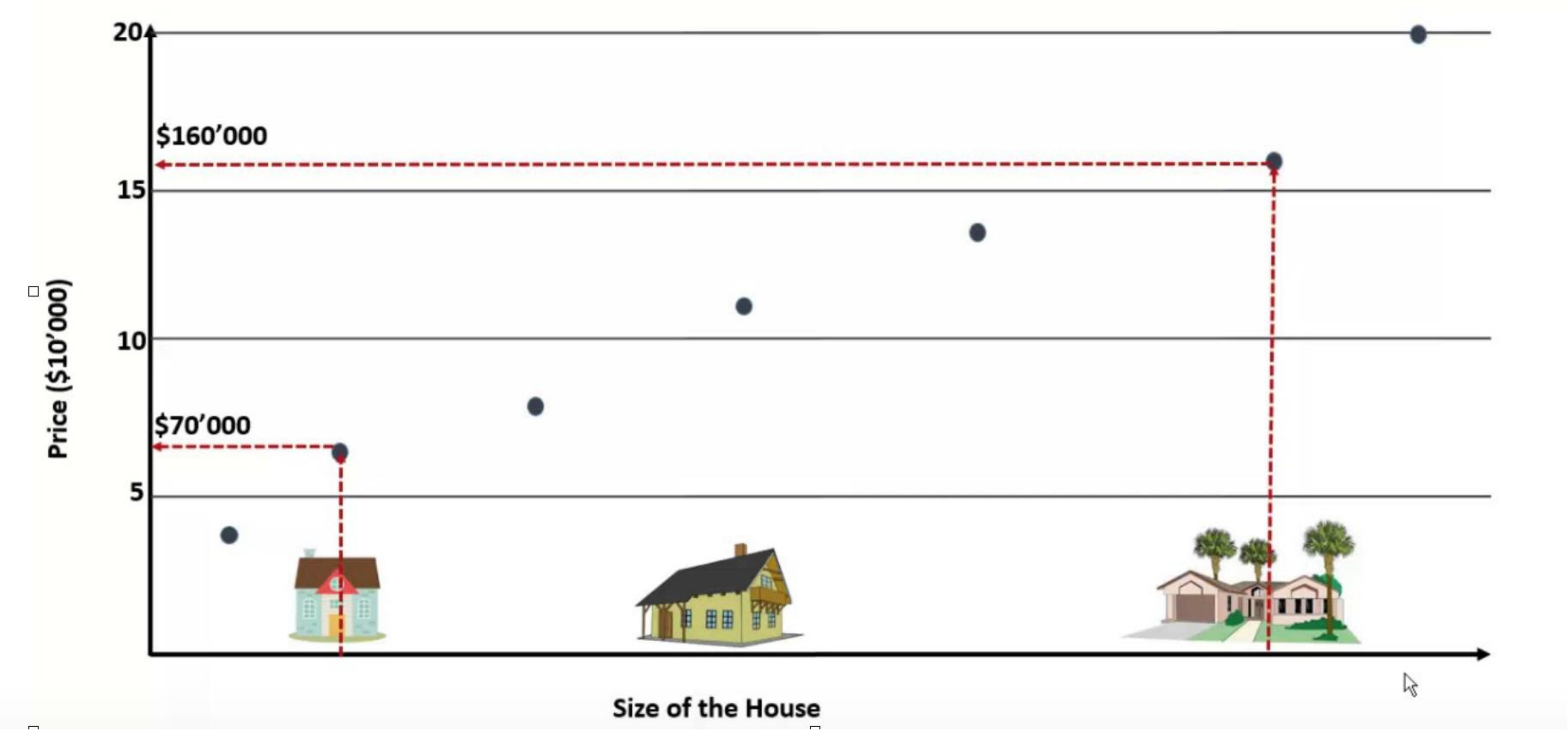
Outline

- Supervised Learning
- Regression
- Classification

- Linear Regression
- Logistic Regression
- Poisson Regression

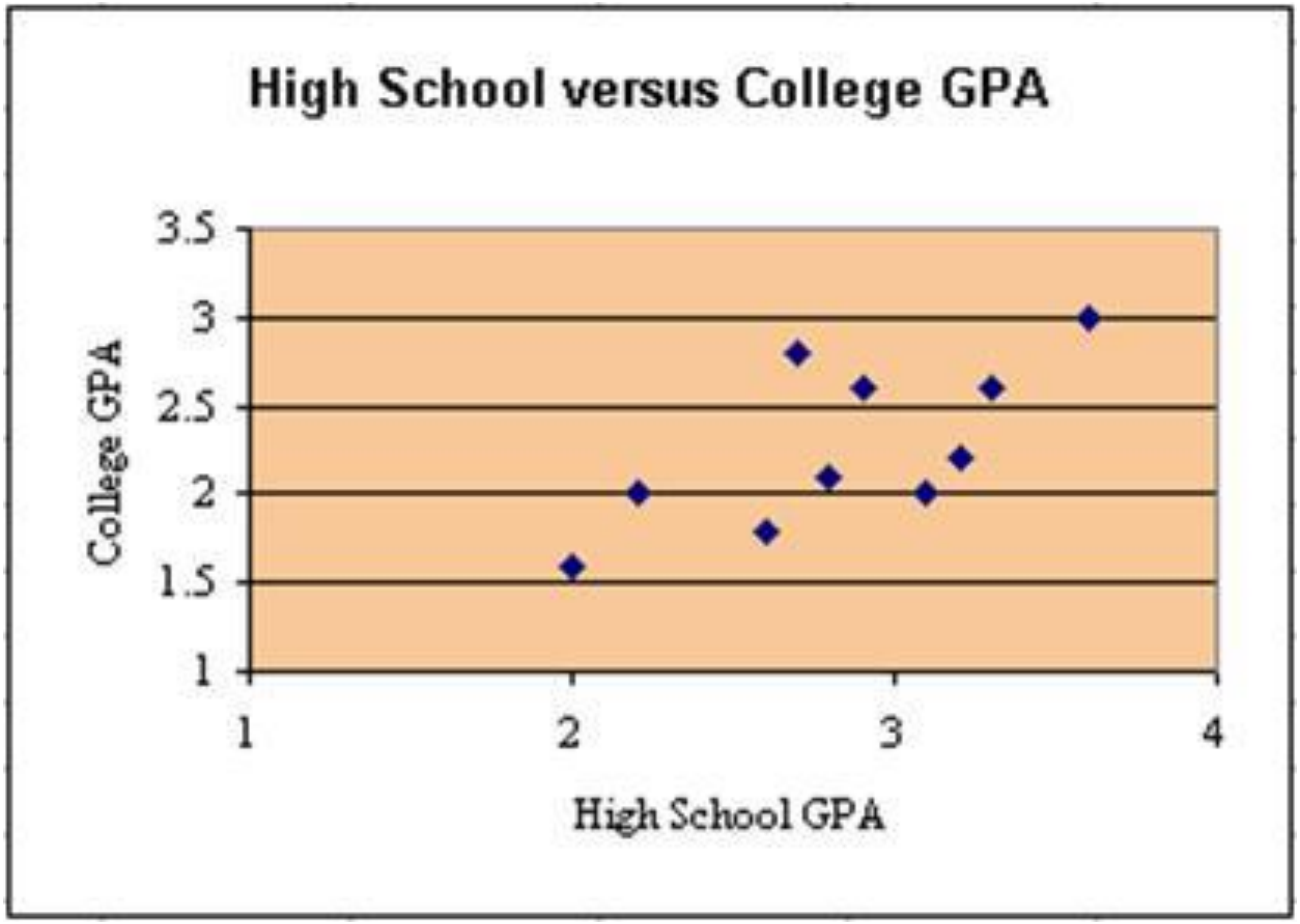
Supervised learning : Regression

Estimating Price of a house

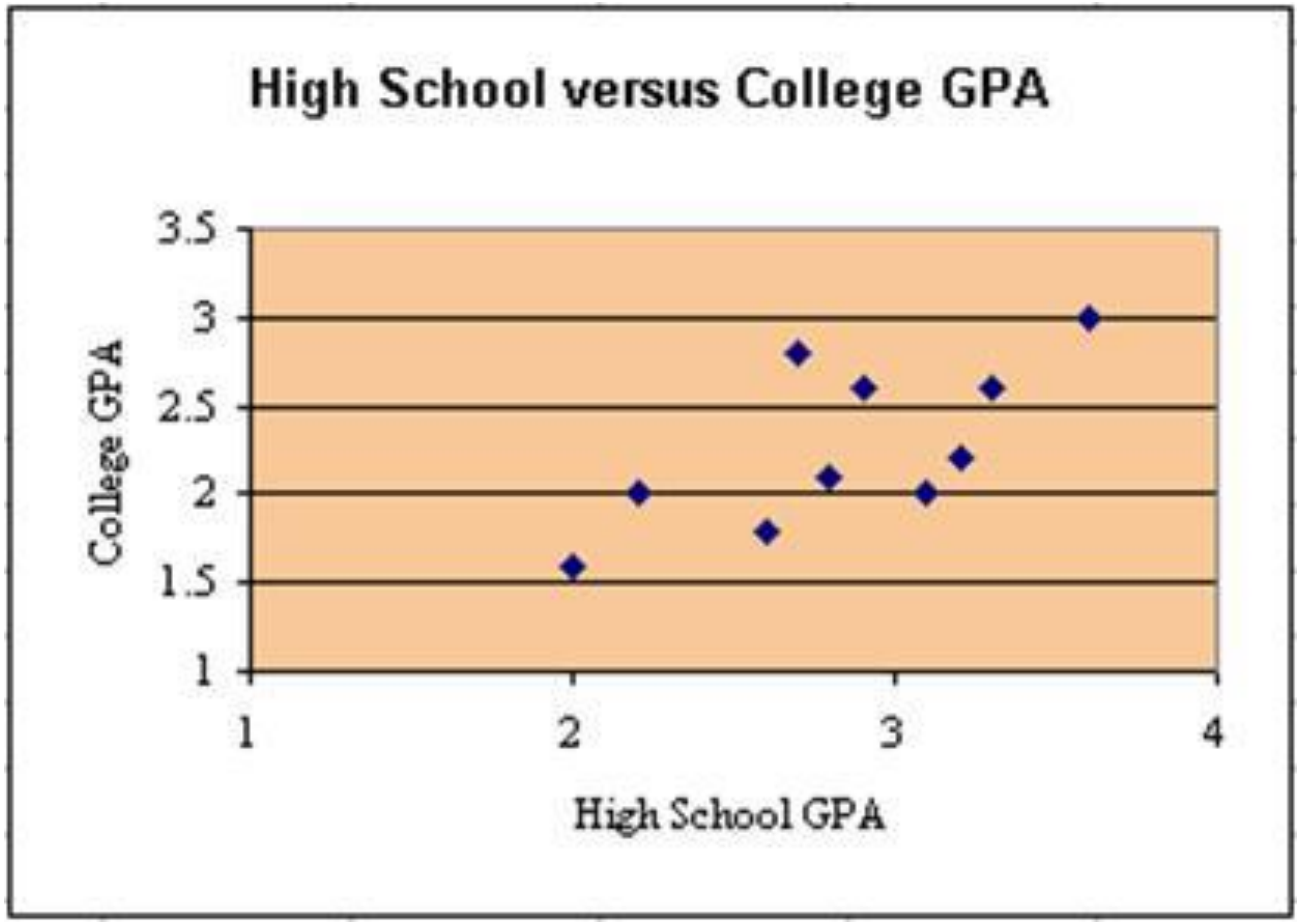


https://www.youtube.com/watch?v=dnKETo-gWbc

Supervised learning : Regression



Supervised learning : Regression



Real valued targets (outputs)

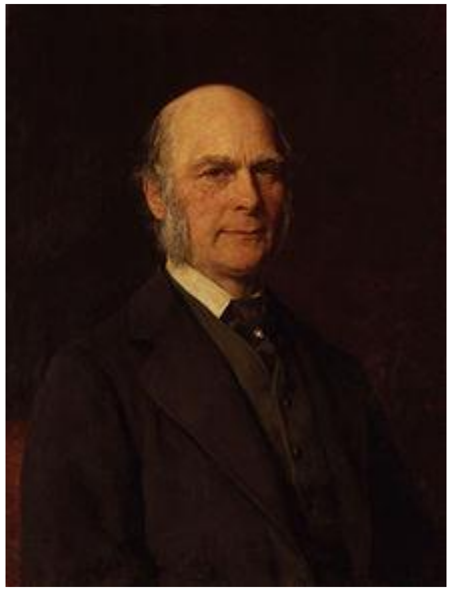
Generalization performance

Goal is to learn a function which maps inputs to outputs so that it will predict well on future data points

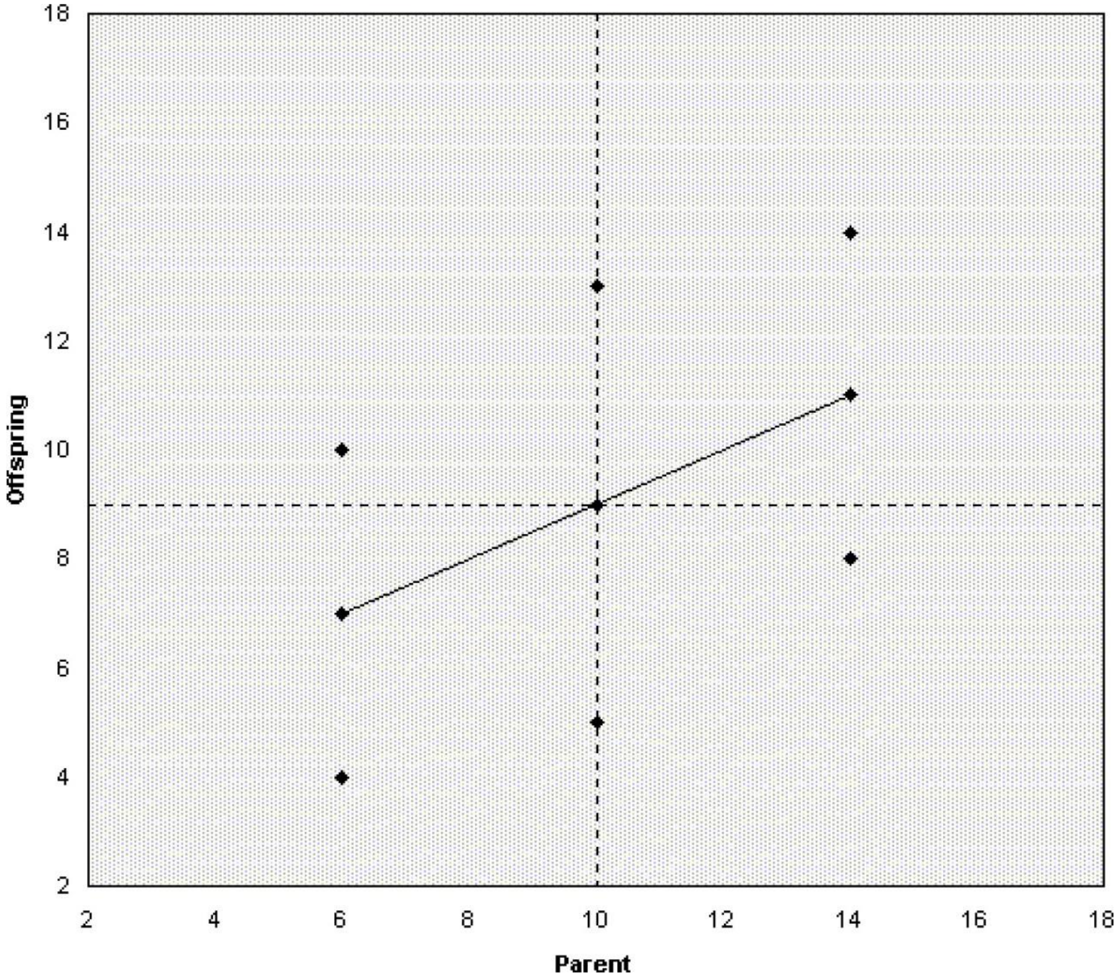
https://medium.com/@heyozramos/regression-vs-classification-86d73c281c5e

A historical look at regression

Genetics : parent sweet pea size on the *X*-axis and the offspring sweet pea size on the *Y*-axis (1900s)



Sir Francis Galton

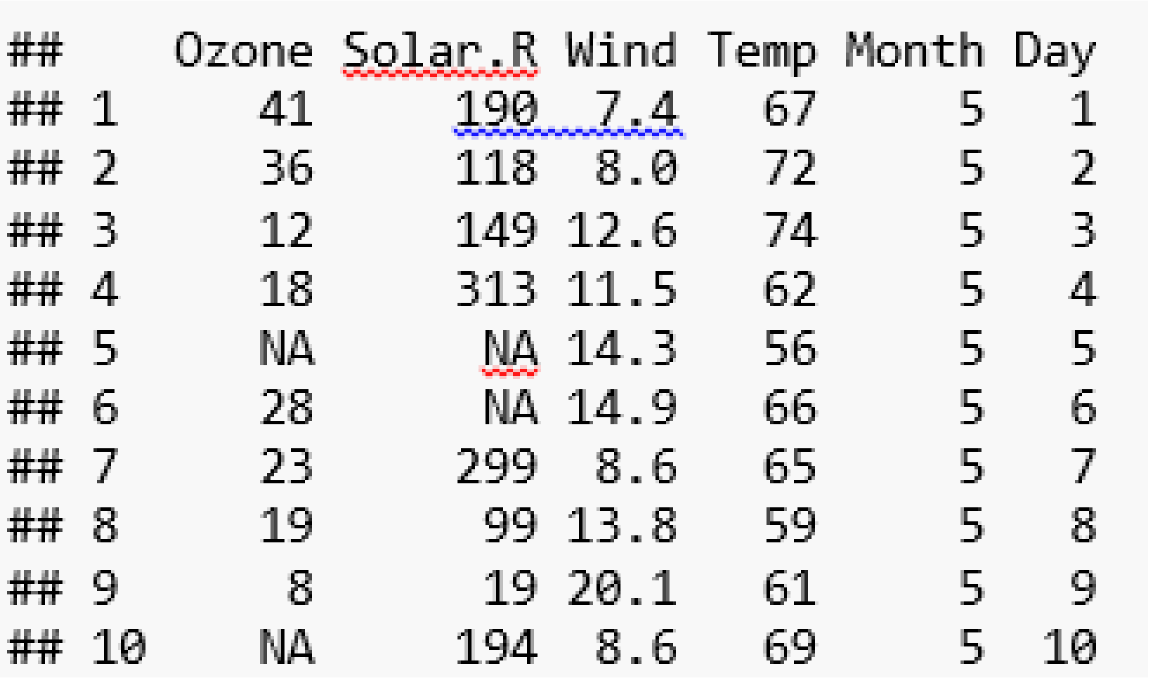


{(6, 4), (6, 7), (6, 10), (10, 5), (10, 9), (10, 13), (14, 8), (14, 11), (14, 14)}

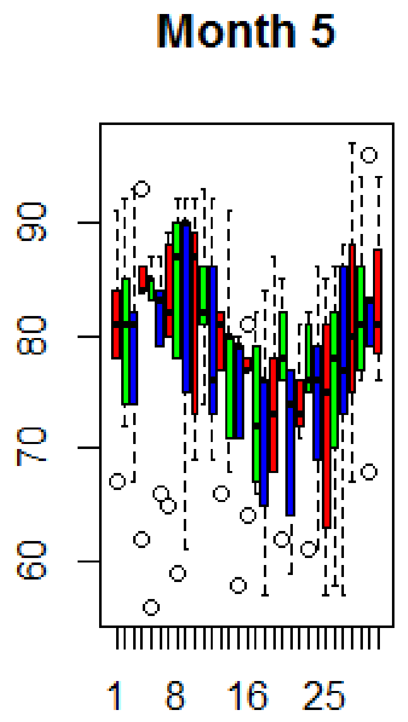
Airquality data.

- Data set has various air quality parameters in New York city.
- These are the parameters in the data set:
- Daily temperature from May to August
- Solar radiation data
- Ozone data
- Wind data
- Goal : predict the temperature for a particular month in New York using solar radiation, ozone and wind data.

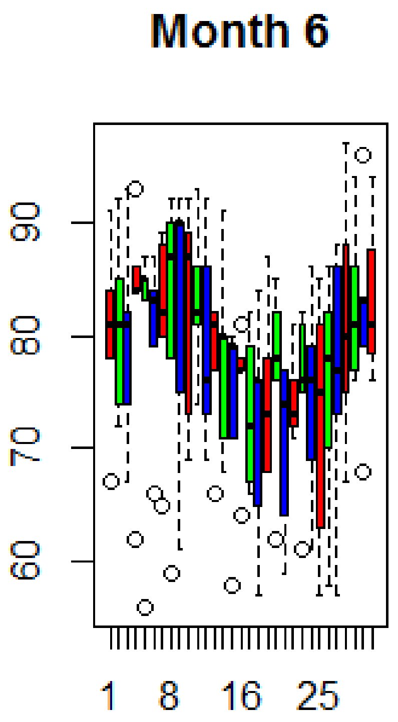
Airquality data



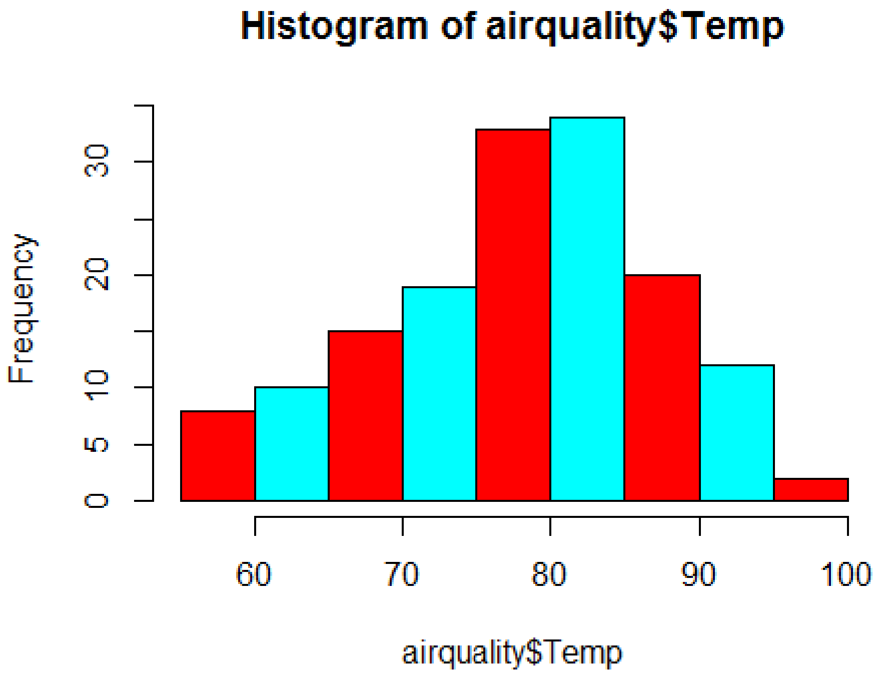
```
##    Ozone Solar.R Wind Temp Month Day
## 1     41     190  7.4   67     5   1
## 2     36     118  8.0   72     5   2
## 3     12     149 12.6   74     5   3
## 4     18     313 11.5   62     5   4
## 5     NA      NA 14.3   56     5   5
## 6     28      NA 14.9   66     5   6
## 7     23     299  8.6   65     5   7
## 8     19      99 13.8   59     5   8
## 9      8      19 20.1   61     5   9
## 10    NA     194  8.6   69     5  10
```



Month 5



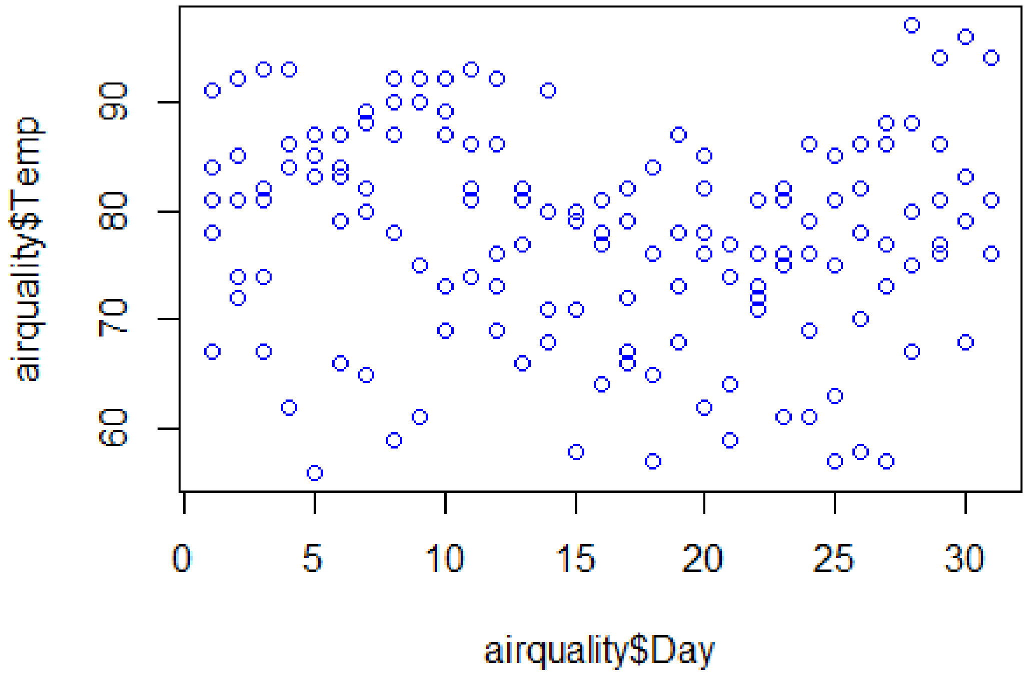
Month 6

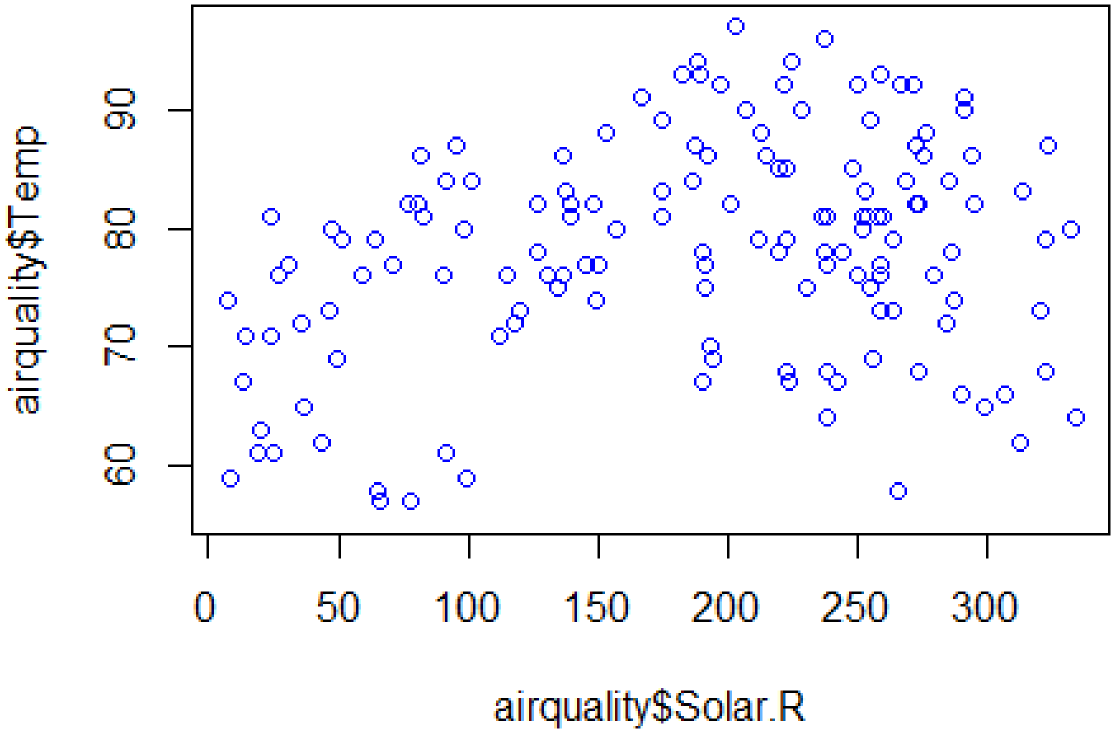


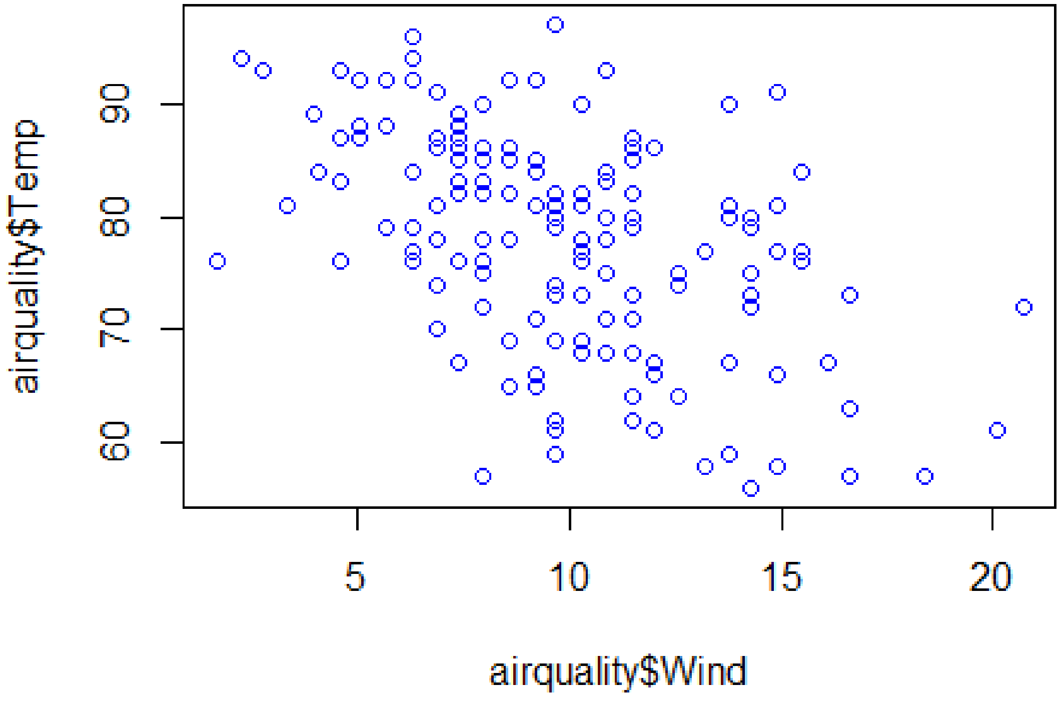
Histogram of airquality$Temp

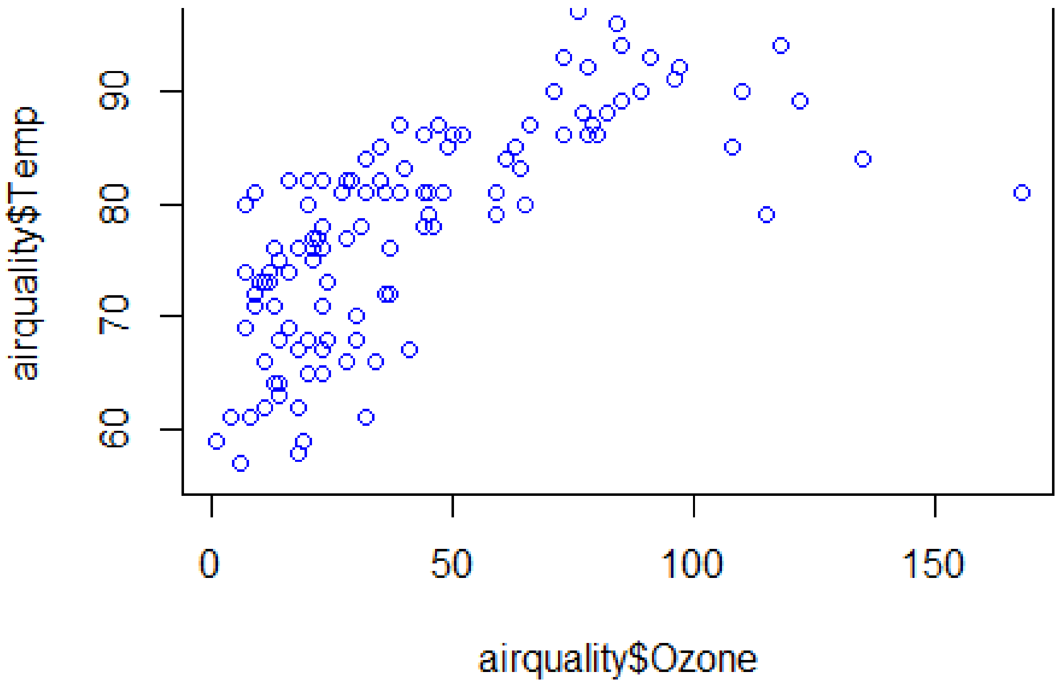
https://www.edvancer.in/step-step-guide-to-execute-linear-regression-r/

Airquality data







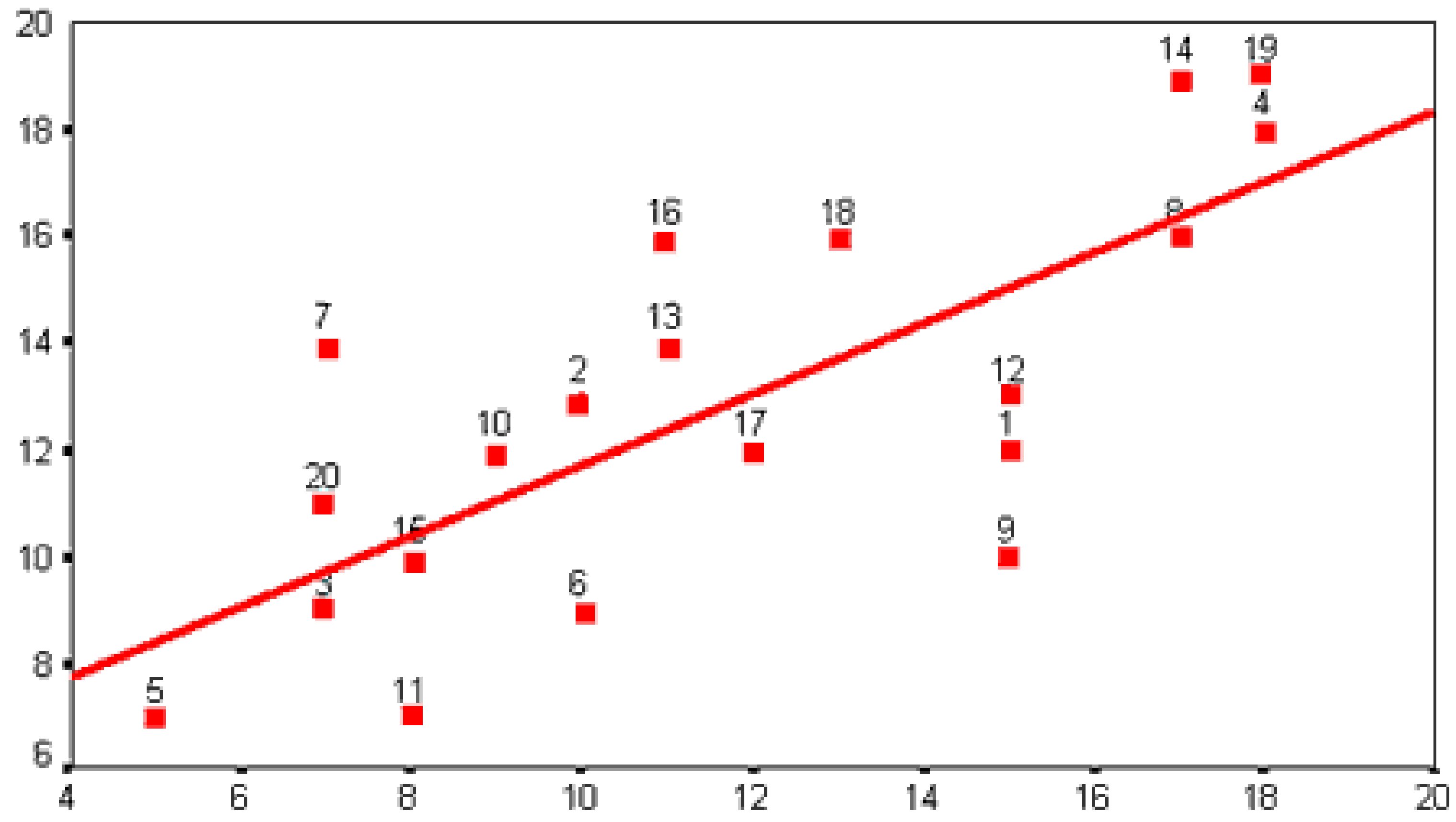


Linear regression

- Temp=w1.Solar.R +w2.Ozone + w3.Wind + error.
- Temperature of house depends on ozone, wind and solar radiations
- linear regression helps to discover relation between dependent and independent variables

Linear Regression

- Observations need not lie on a line
  - Observations are not generated by a linear line
  - Observations are noisy, due to measurement errors

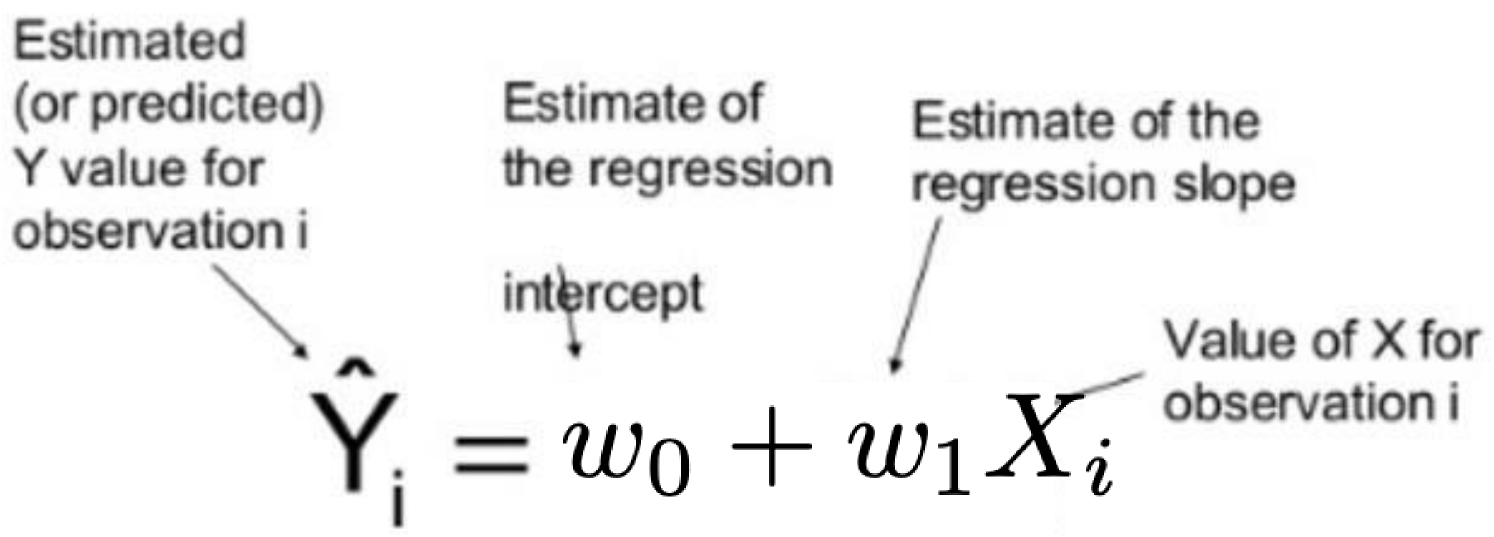


Linear regression

- Learn a function which maps input to output f : X -> Y

Regression Output is real and scalar, $y \in \mathbb{R}$

- Consider a Linear function



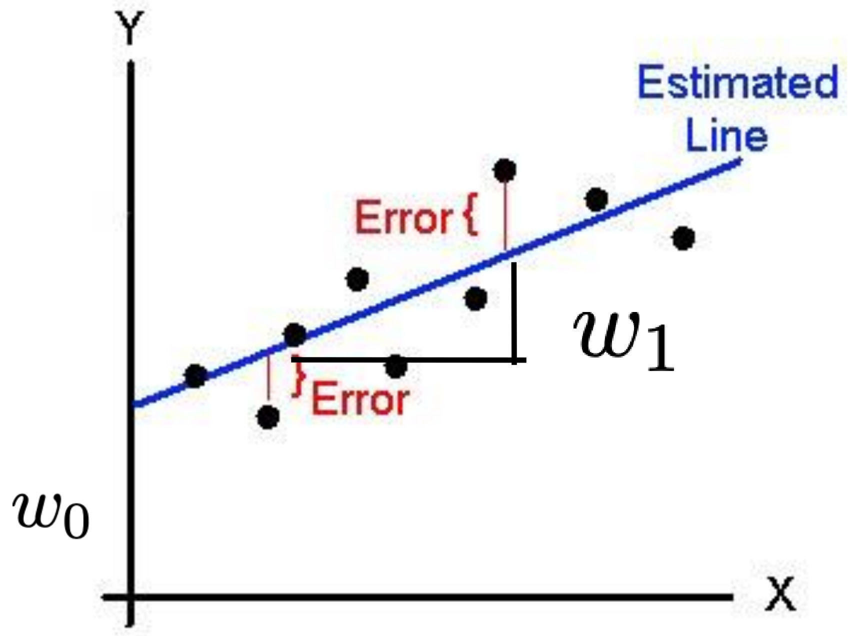
Estimated (or predicted) Y value for observation i

Estimate of the regression intercept

Estimate of the regression slope

Value of X for observation i

$$\hat{Y}_i = w_0 + w_1 X_i$$



Linear regression

- Learn a function which maps input to output f : X -> Y

Regression Output is real and scalar, $y \in \mathbb{R}$

- Consider a Linear function

Estimated (or predicted) Y value for observation i

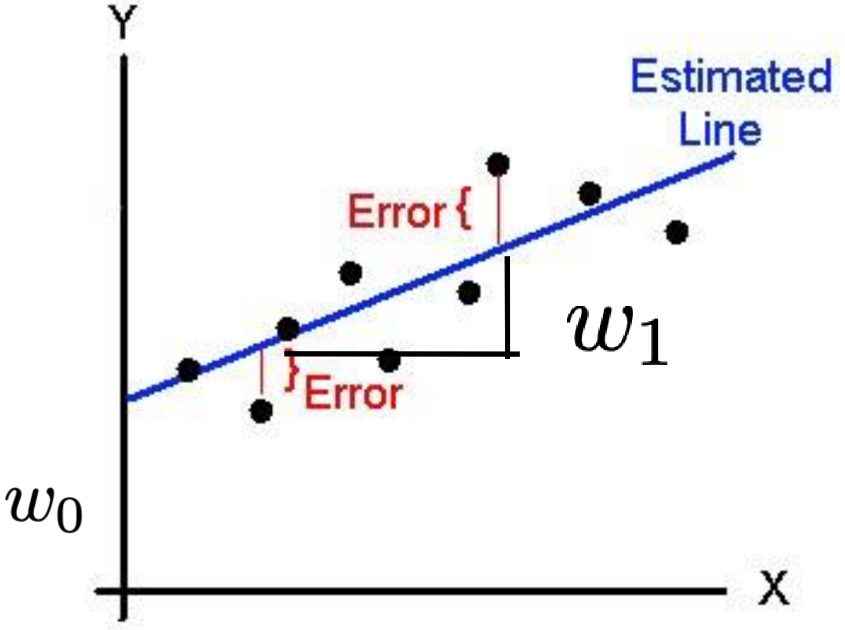
Estimate of the regression intercept

Estimate of the regression slope

Value of X for observation i

$$\hat{Y}_i = w_0 + w_1 X_i$$

$$= X_i^\top w$$



1 dim input $X_i = [1, X_i]^T$ $w = [w_0, w_1]^T$

D dim input $X_i = [1, X_{i1}, \ldots, X_{iD}]^T$ $w = [w_0, w_1, \ldots, w_D]^T$

Linear Regression - Learning Parameters

- Learn the function which passes through as many points as possible : Minimize the **Least Squares Error**

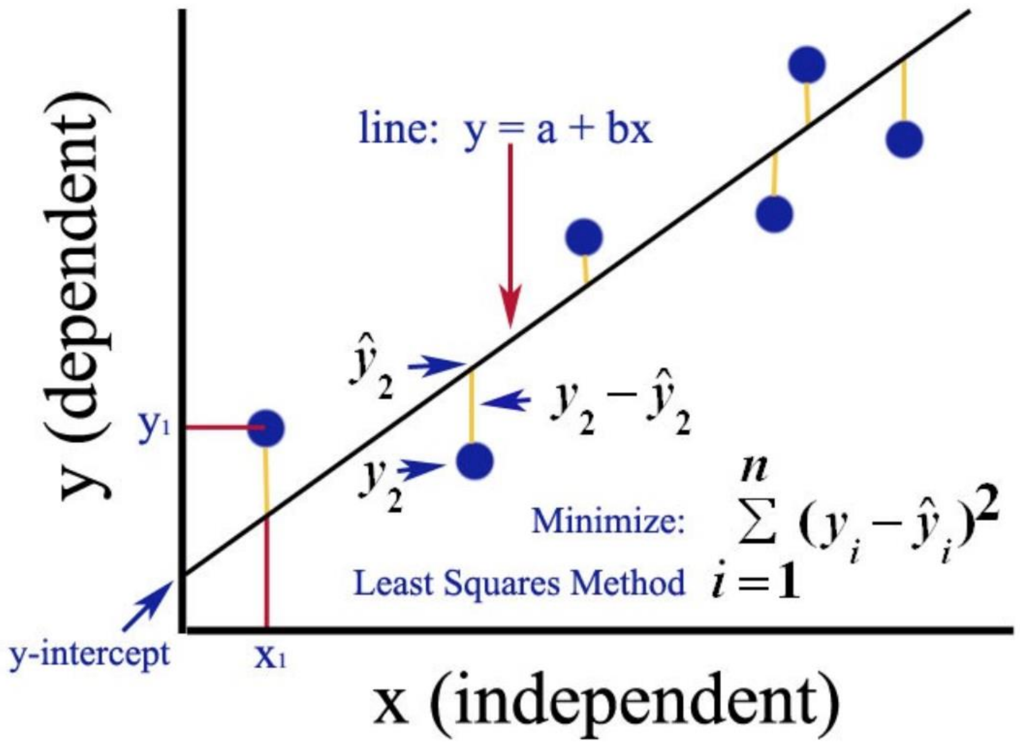
$$E(w) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$= \frac{1}{2}\sum_{i=1}^{N}(y_i - X_i^\top w)^2$$

$$\frac{1}{2}\|(y - X^\top w)\|^2$$

$X_i = [1, X_{i1}, \ldots, X_{iD}]^T$

Design matrix $X = \begin{pmatrix} X_{1,} X_{2,} \ldots X_N \end{pmatrix}_{(D+1) \times N}$



Linear Regression - Learning Parameters

- Learn the function which passes through as many points as possible : Minimize the **Least Squares Error**

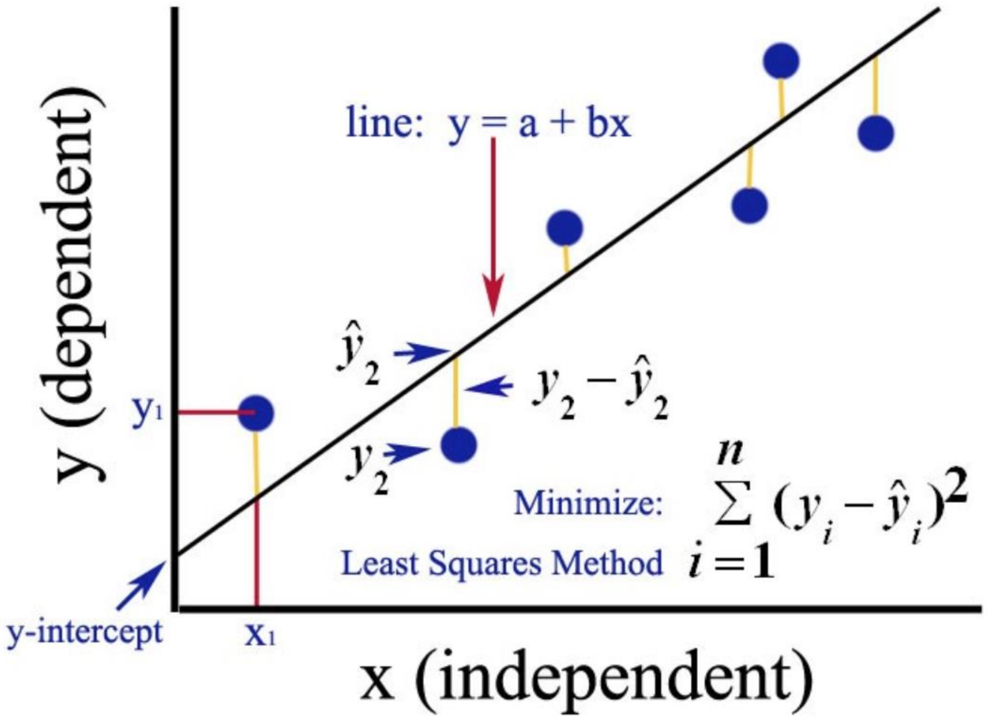
$$X_i = [1, X_{i1}, \ldots, X_{iD}]^T$$

Design matrix $X = \begin{pmatrix} X_{1,} X_{2,} \ldots X_N \\ \\ \end{pmatrix}_{(D+1) \times N}$

$$E(w) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$= \frac{1}{2}\sum_{i=1}^{N}(y_i - X_i^\top w)^2$$

$$\frac{1}{2}\|(y - X^\top w)\|^2$$



$$\nabla E(w) = Xy - XX^\top w = 0$$

$$\boxed{w_{ML} = (XX^\top)^{-1}Xy}$$

$$\frac{\partial}{\partial s}(x - As)^\top W(x - As) = -2(x - As)^\top WA$$

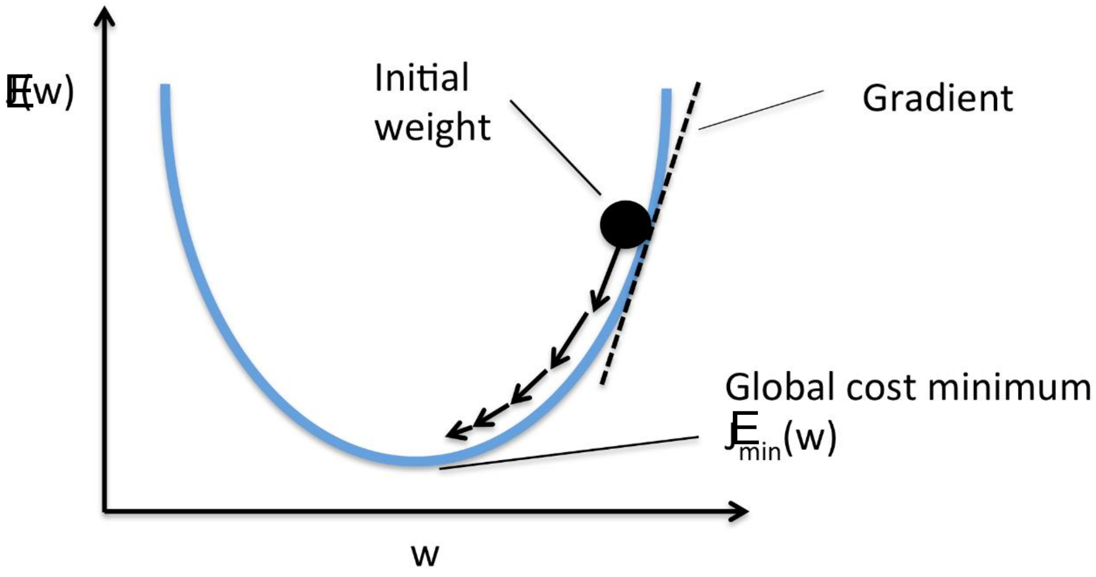
Linear Regression - Learning Parameters

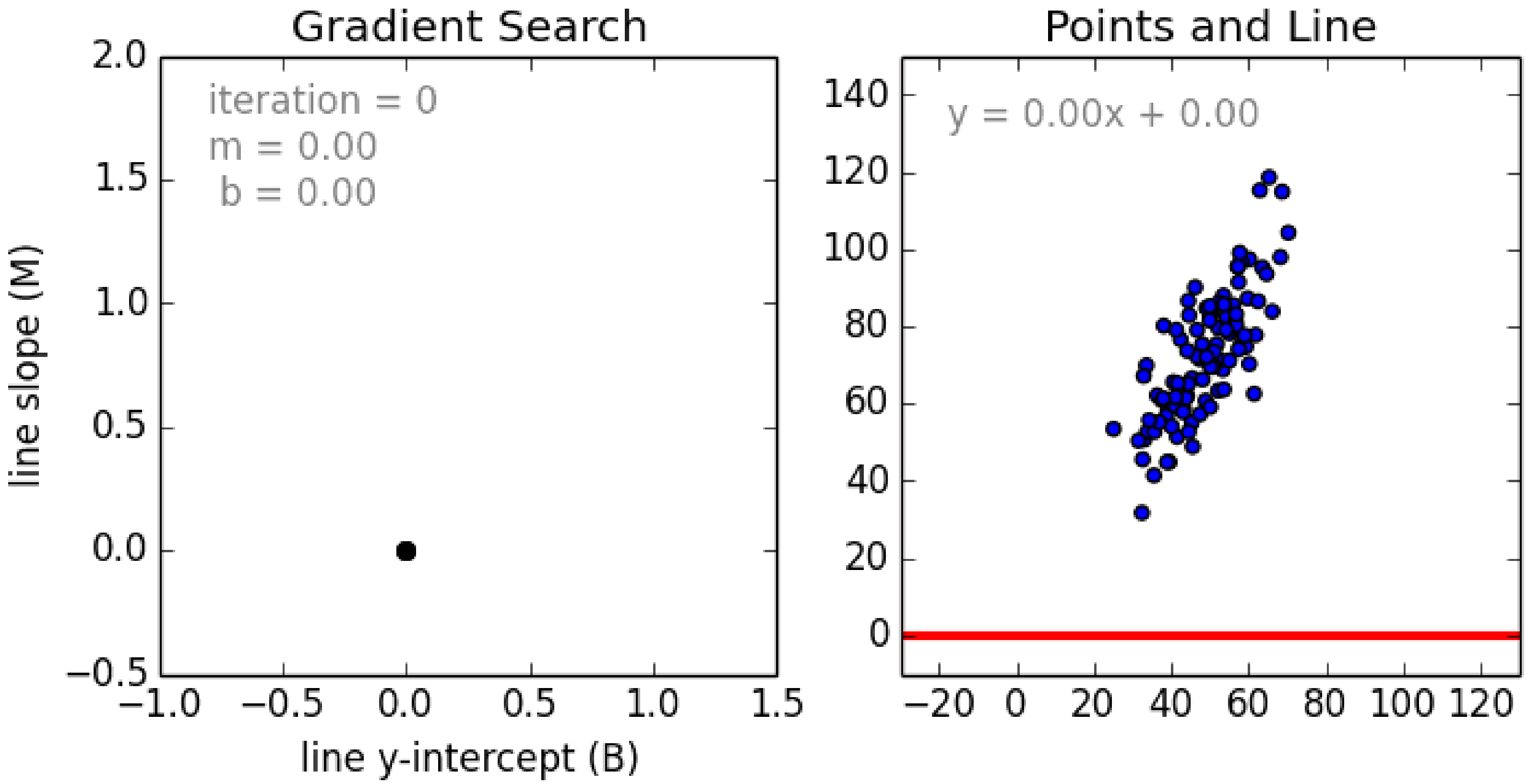
- Learn the function which passes through as many points as possible
  : Minimize the **least squares error** using **gradient descent**

$$\nabla E(w) = Xy - XX^{\top}w$$

$$Error_{(m,c)} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + c))^2$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n$$





Question

- write down three equations for the line **y = mx+c** to go through **y = 7** at **x = -1**, **y = 7** at **x = 1** and **y = 21** at **x = 2**. Find the least squares solution **(c,m)**?

- Implement In python least squares solution to linear regression
  - Analytical approach
  - Gradient descent approach

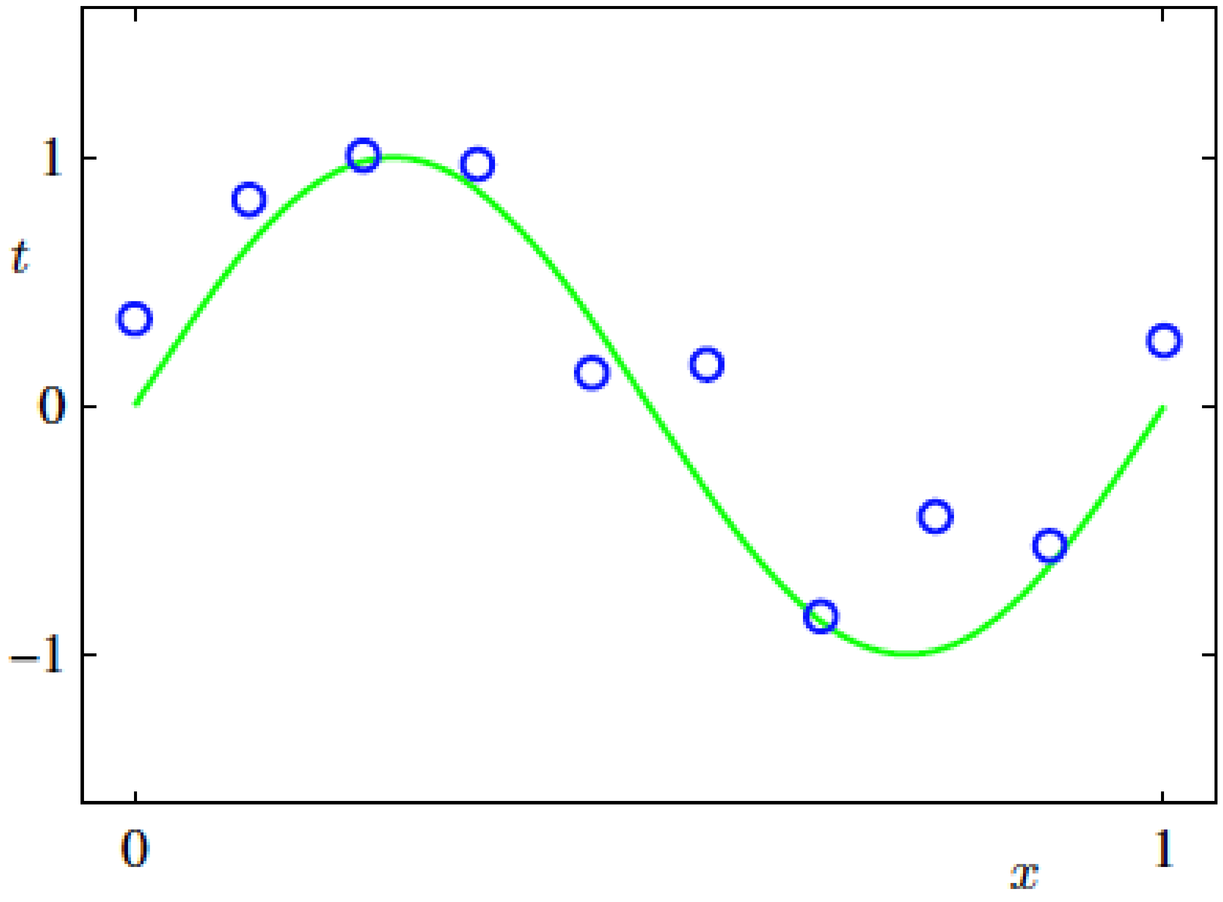
Question

- write down three equations for the line $y = mx+c$ to go through $y = 7$ at $x = -1$, $y = 7$ at $x = 1$ and $y = 21$ at $x = 2$. Find the least squares solution (c,m)?

- Answer : (9,4)

Non Linear Regression - curve fitting

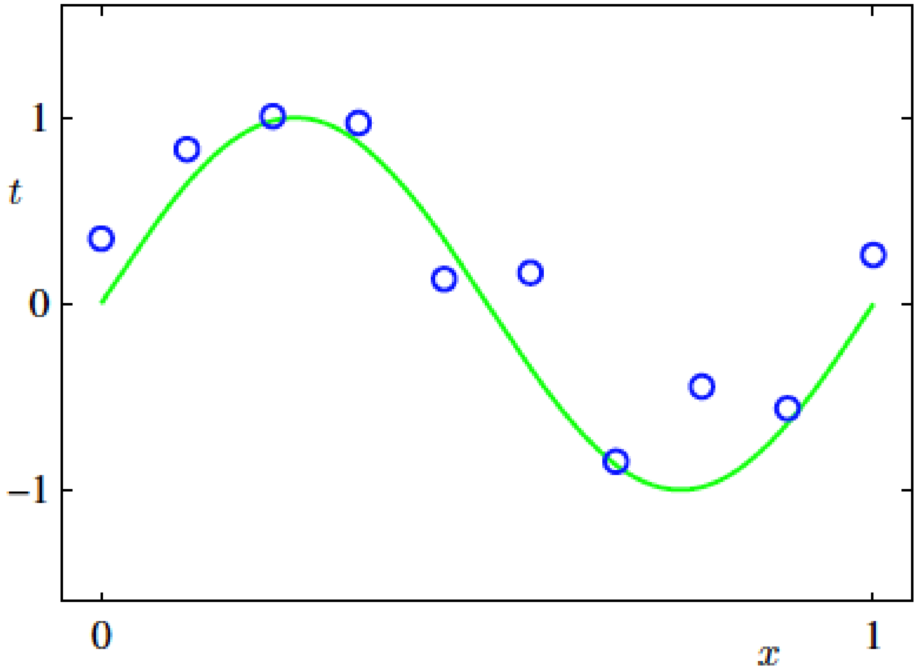
- Remember high school maths !
- Real-valued target variable t.
- Training set comprising N observations



Regression - curve fitting

- M is the order of the polynomial, y(x,w) is a nonlinear function of x, it is a linear function of the coefficients w.
- Functions, such as the polynomial, which are linear in the unknown parameters have important properties and are called **linear models**

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

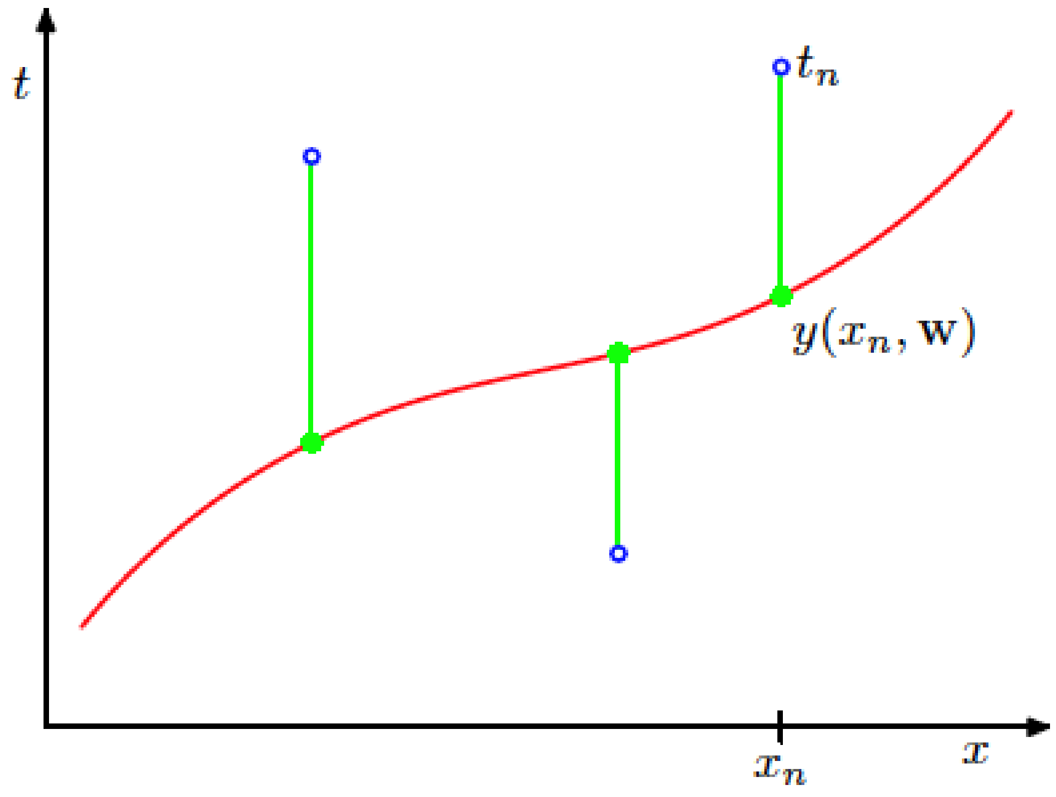


Regression - curve fitting

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

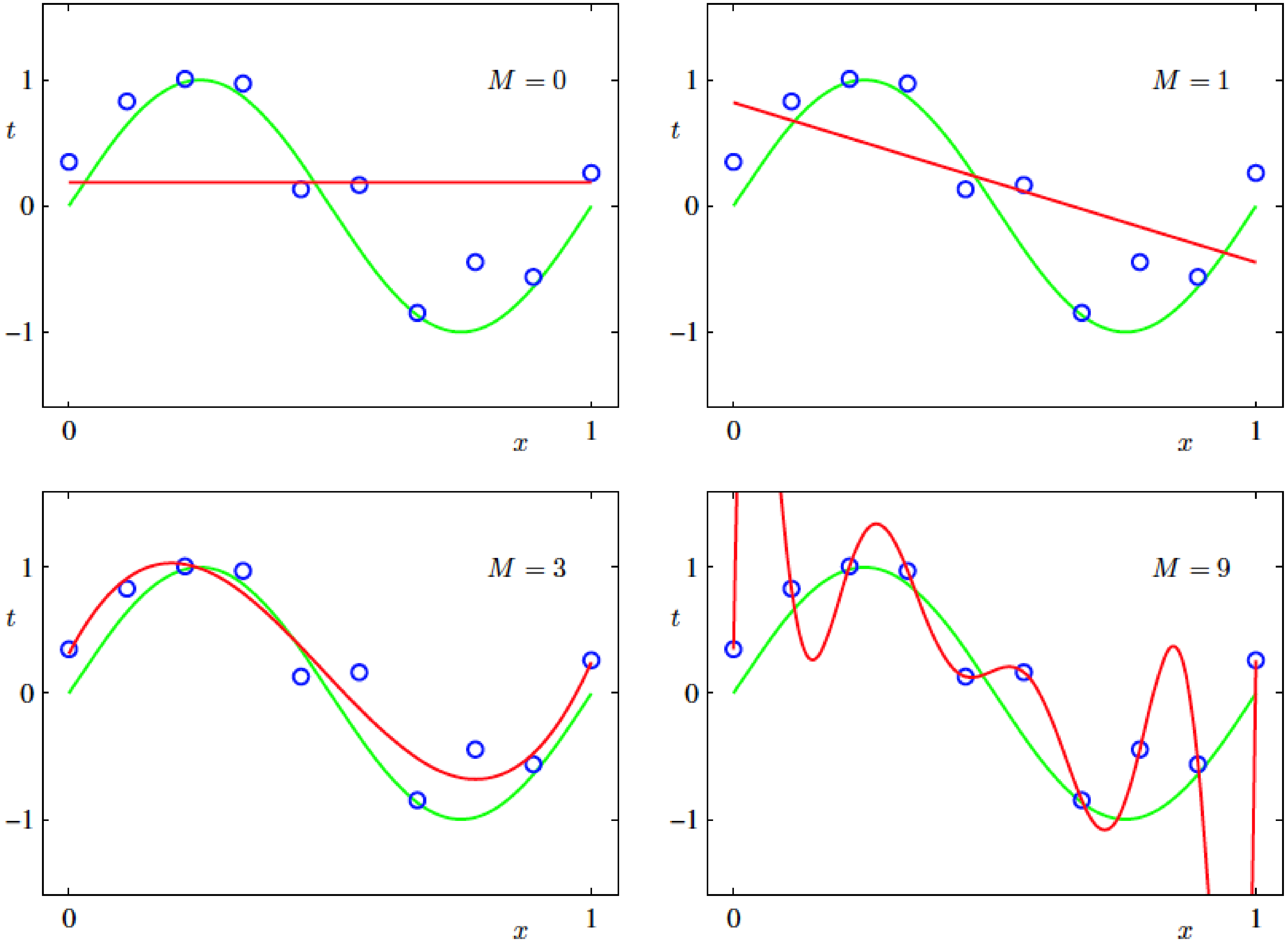
- Coefficients will be determined by fitting the polynomial to the training data. This can be done by minimizing an error function

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



Regression - curve fitting

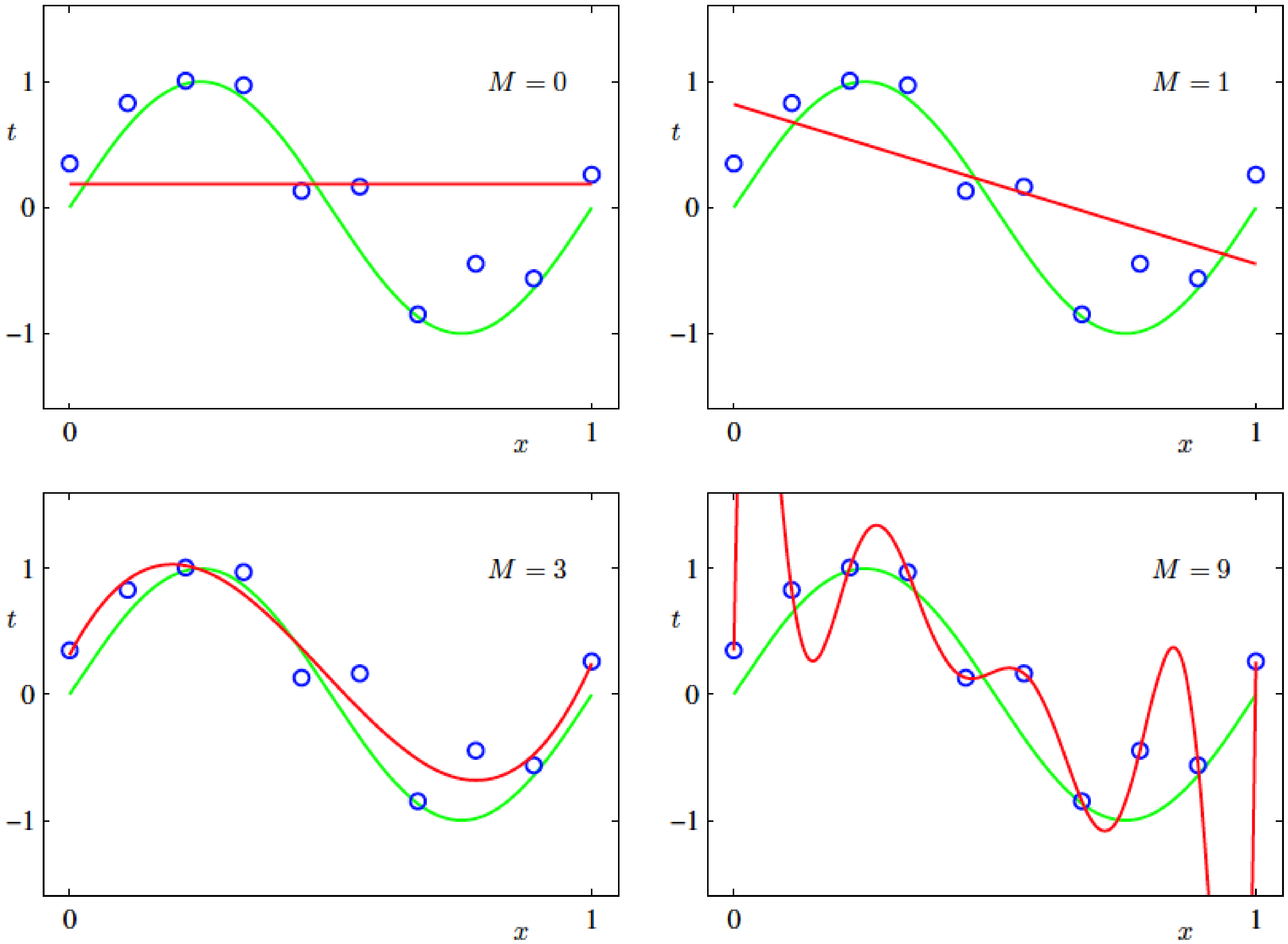
- Model selection (choosing M) : higher order polynomial (M = 9), provide excellent fit to the training data but gives a very poor representation of the function



Regression - curve fitting

- Model selection (choosing M) : hi omial (M = 9), provide excellent fit to the training es a very poor representation of the function

Overfitting



model that is too flexible with respect to the number of data

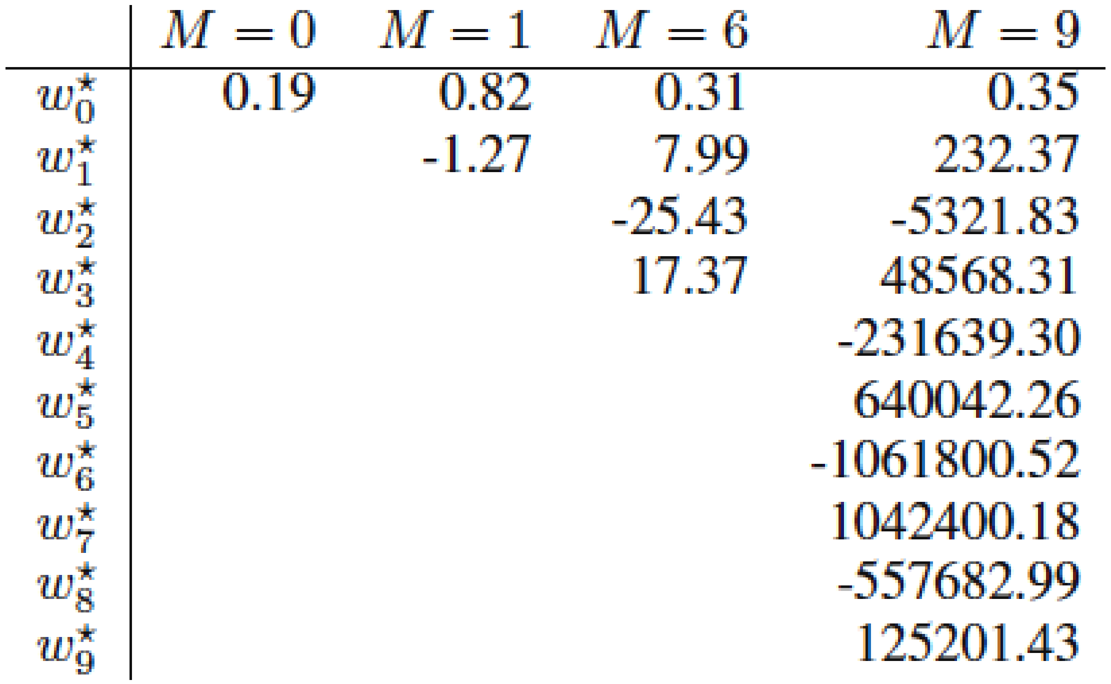
Regression - curve fitting

- Generalization performance : root mean square error on test data

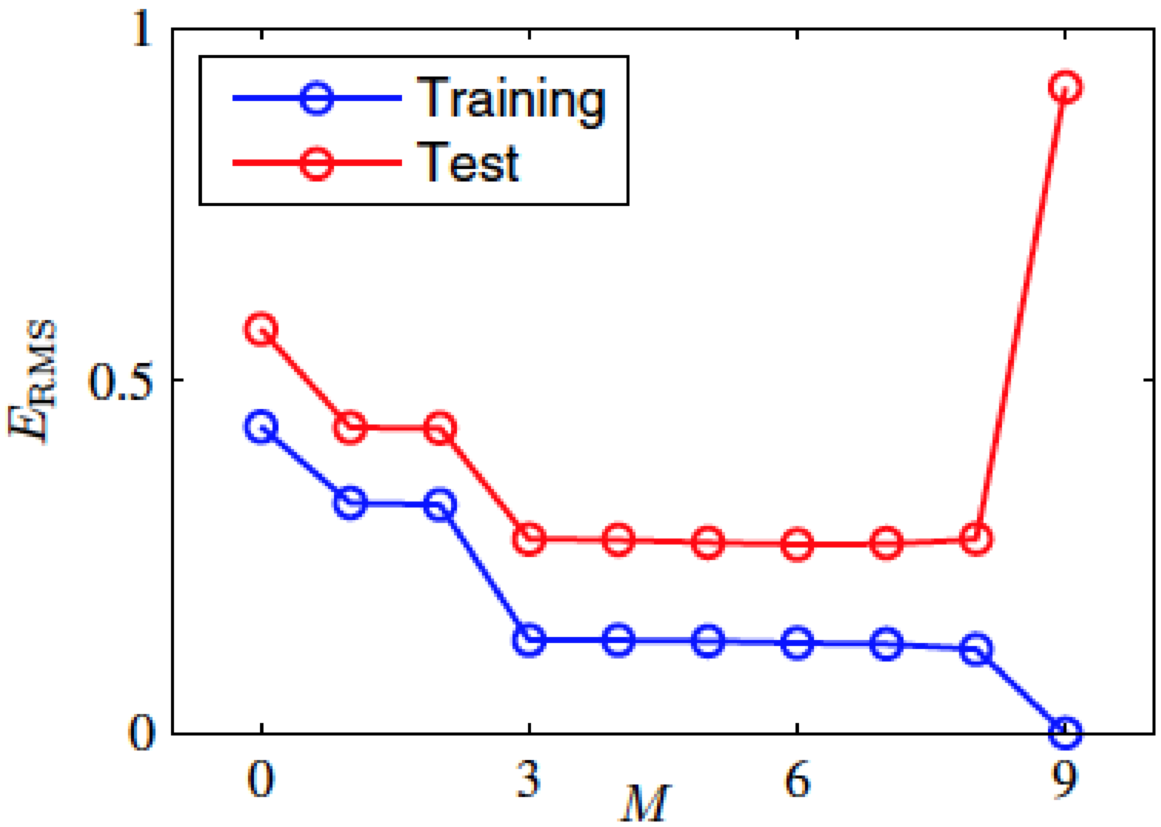
- Weights coefficients for M=9 is extremely large !

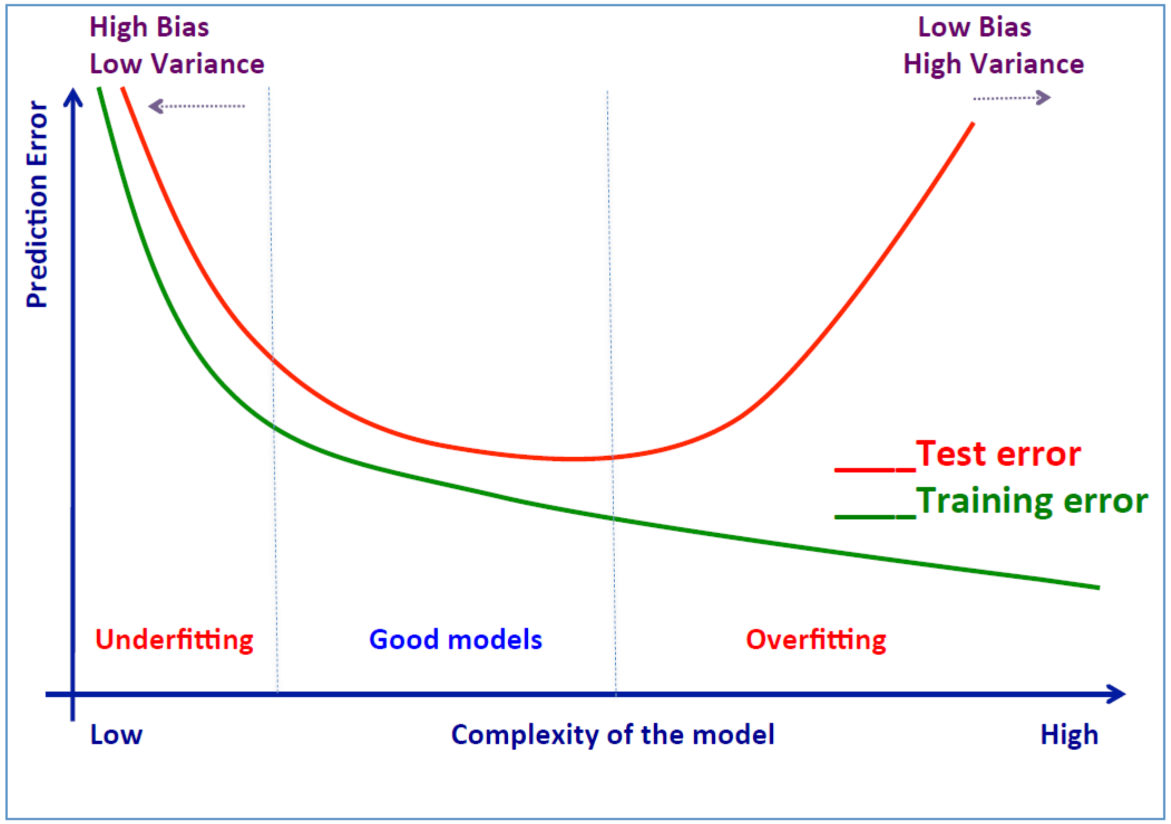
$$E_{\mathrm{RMS}} = \sqrt{2E(\mathbf{w}^\star)/N}$$

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



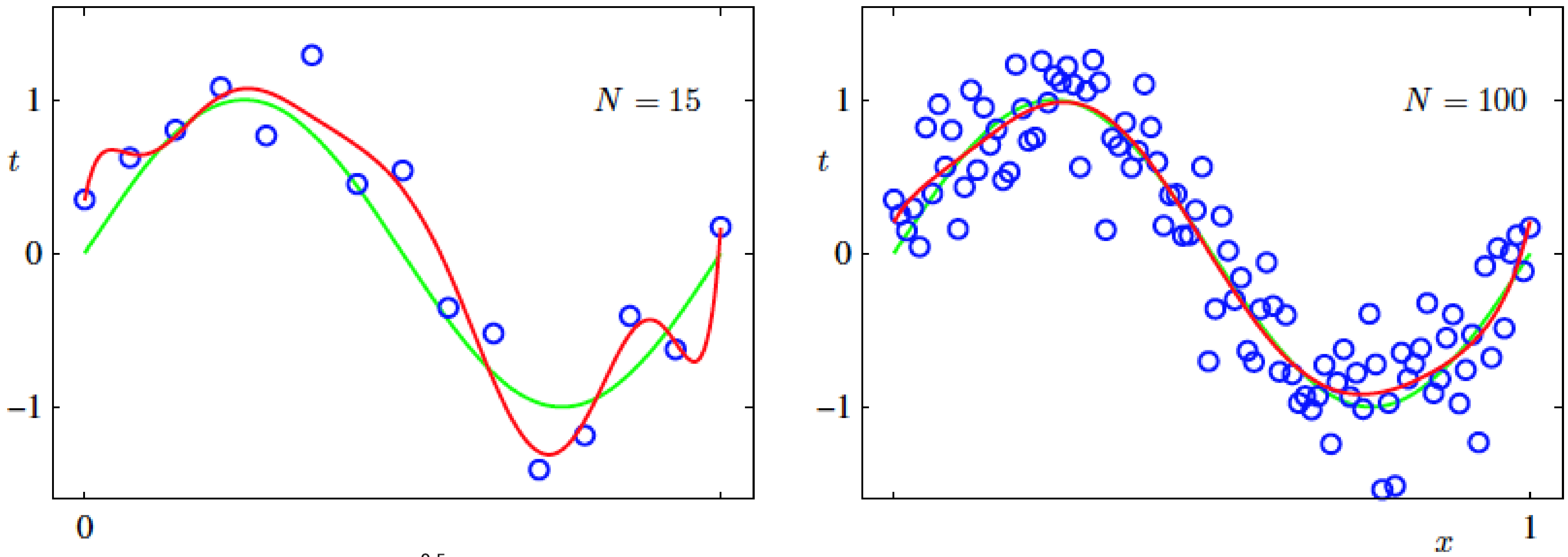
	$M=0$	$M=1$	$M=6$	$M=9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43

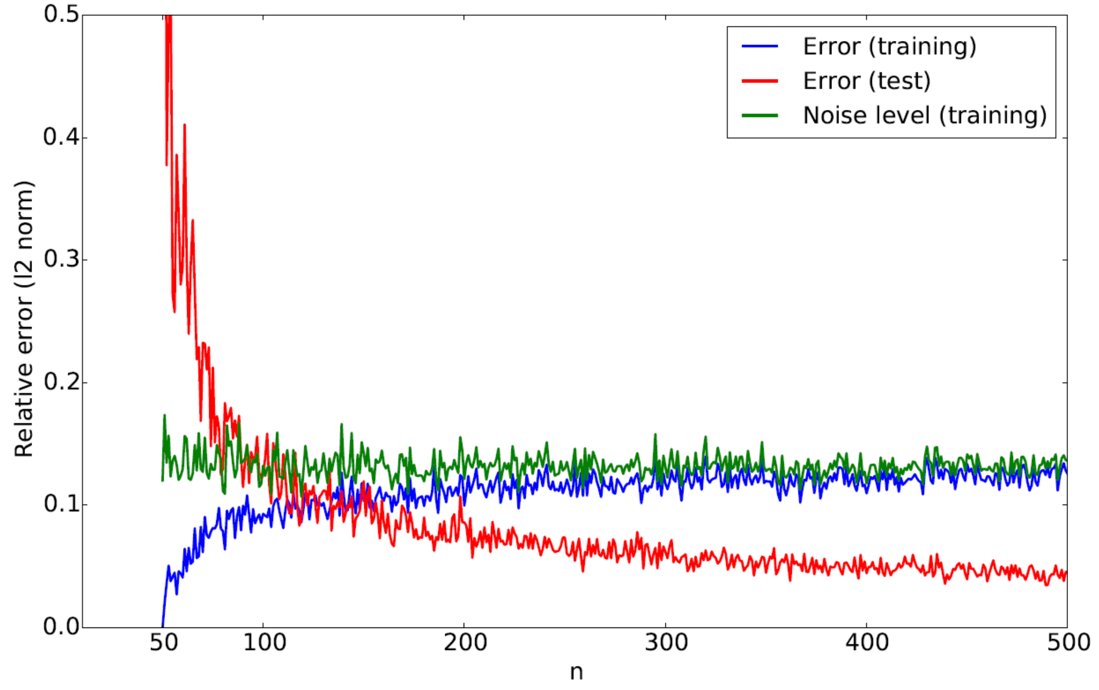




Regression - curve fitting

- Given model complexity, the over-fitting problem become less severe as the size of the data set increases.





Curve fitting - regularization

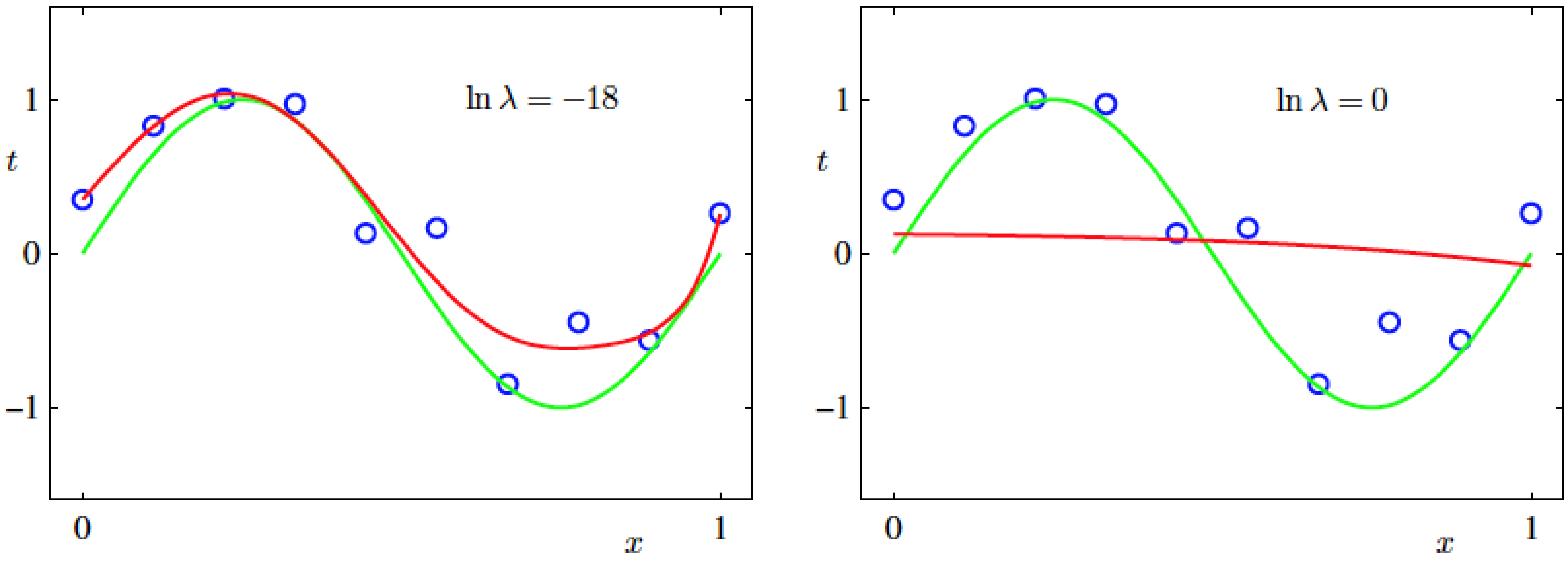
- Add a penalty term to the error function (1.2) in order to discourage the coefficients from reaching large values

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

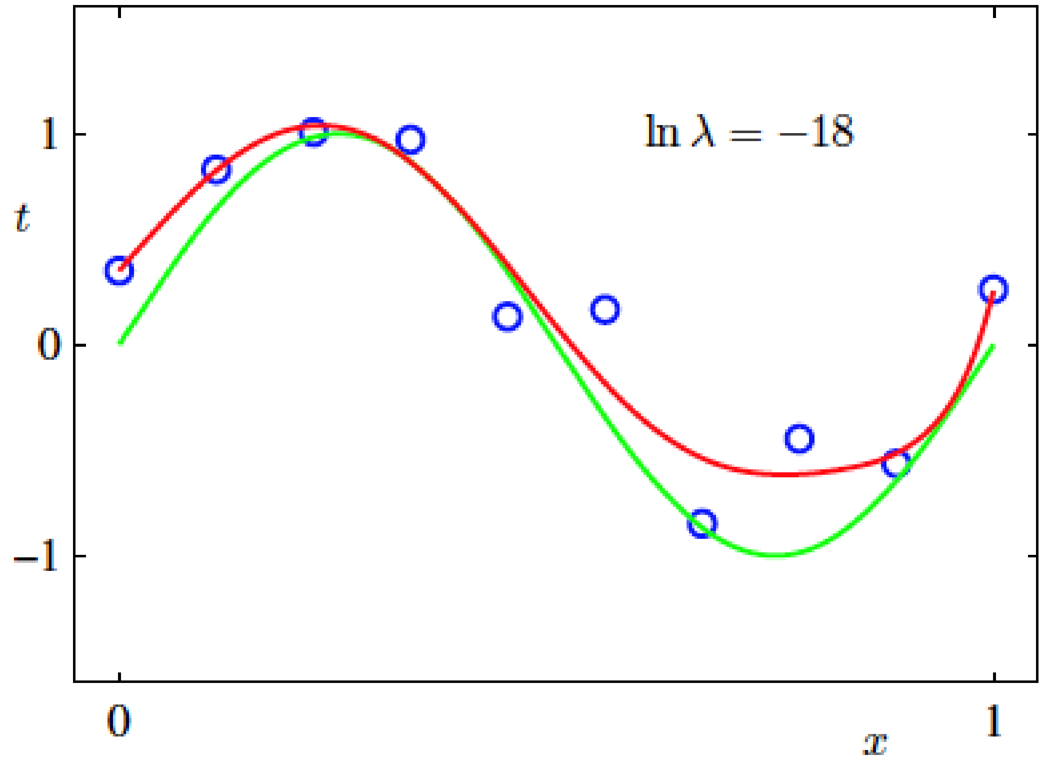
- Ridge regression : L2 norm

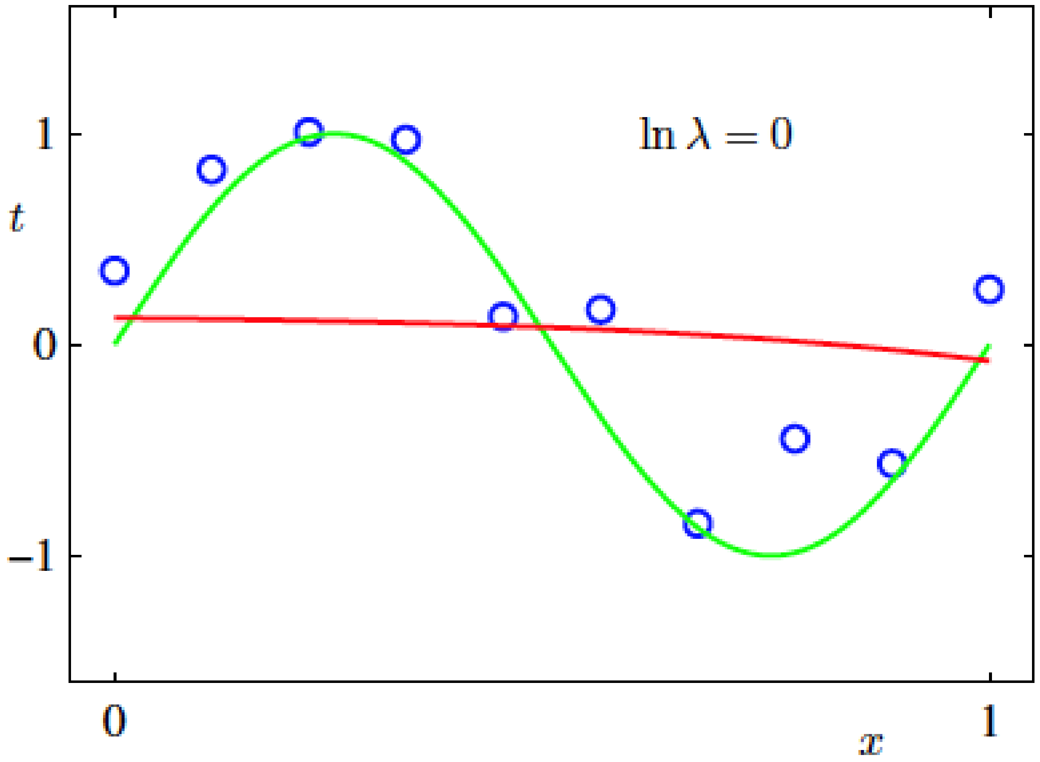
Regularization constant

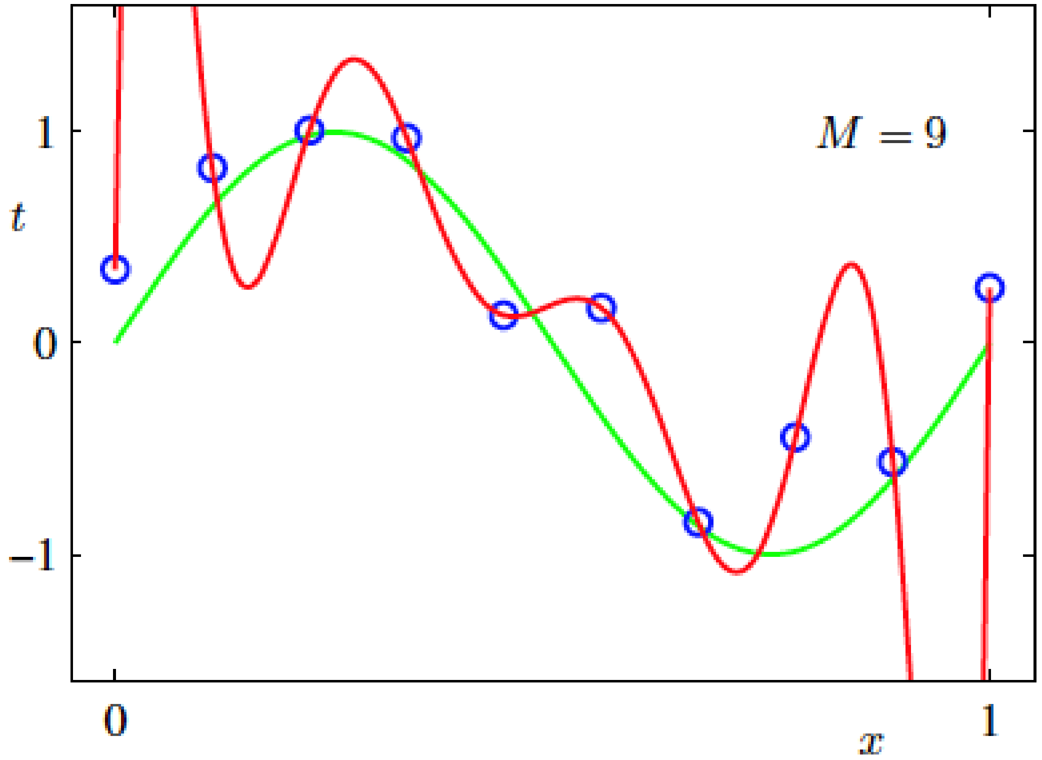
$$\|\mathbf{w}\|^2 \equiv \mathbf{w}^{\mathrm{T}}\mathbf{w} = w_0^2 + w_1^2 + \ldots + w_M^2$$

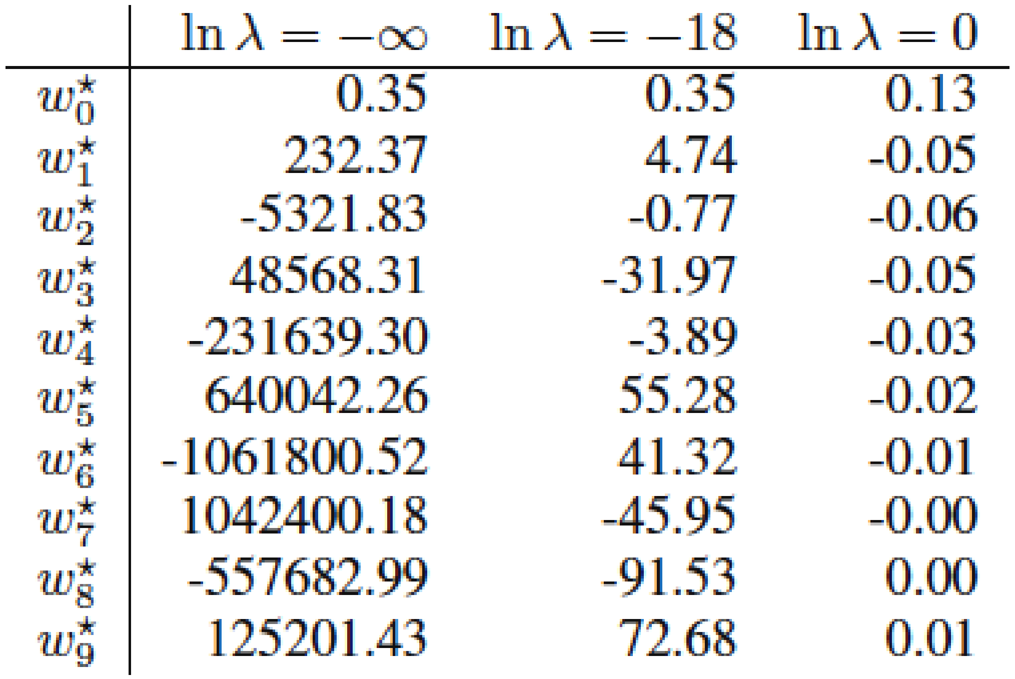


Curve fitting - regularization









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

Regularized Least Squares

- Parameter shrinkage, weight decay

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

- **Ridge regression** q=2

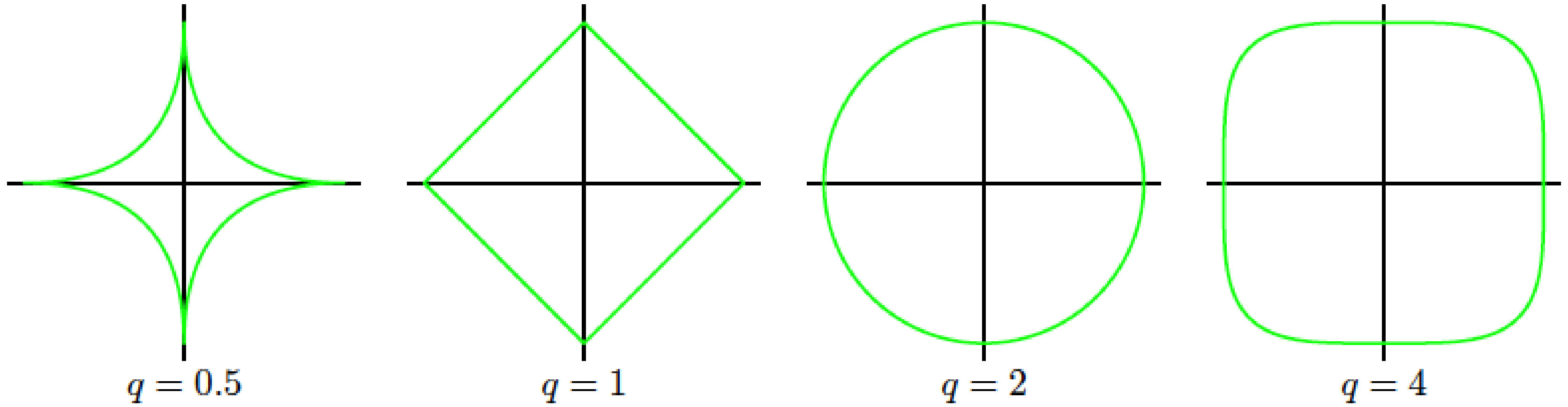
$$\frac{\lambda}{2}\sum_{j=1}^{M} w_j^2$$

- **Lasso regression** q=1, if λ is sufficiently large, some of the coefficients are driven to zero

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|$$

- Elastic net regularization

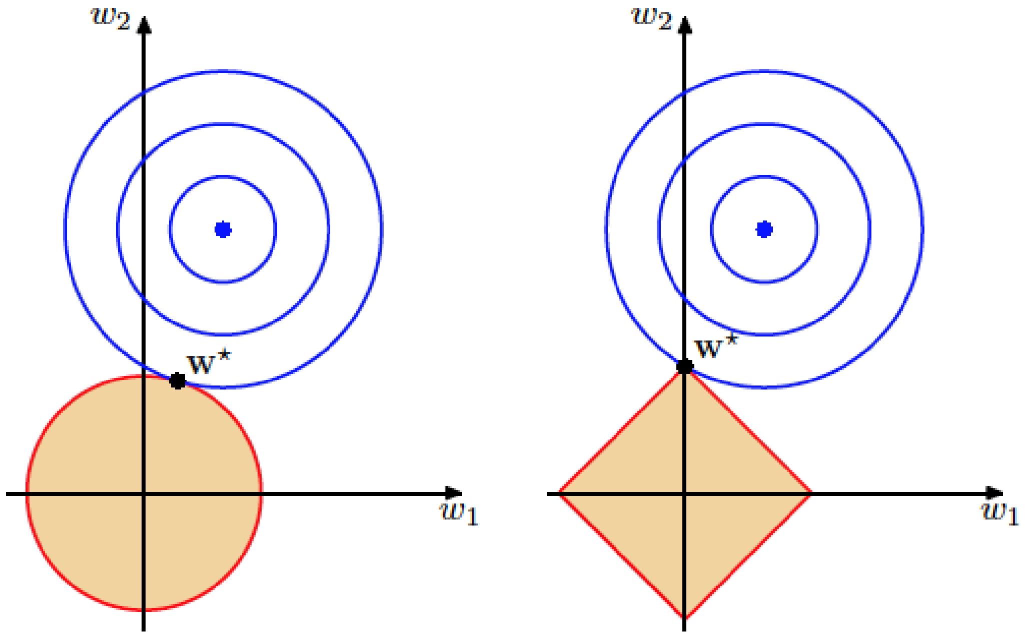
$$\frac{1}{n}\|Y - \mathrm{X}\,w\|_2^2 + \lambda_1\sum_{j=1}^{d}|w_j| + \lambda_2\sum_{j=1}^{d}|w_j|^2$$



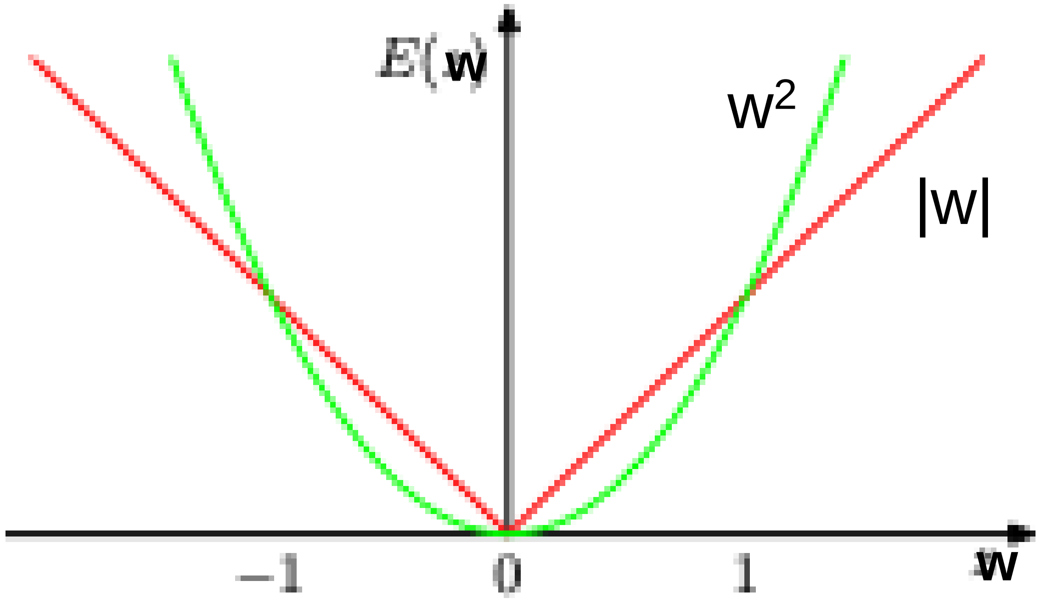
Regularized Least Squares

- Parameter shrinkage, weight decay
- **Ridge regression** q=2 $\frac{\lambda}{2}\sum_{j=1}^{M} w_j^2$
- **Lasso regression** q=1, if λ is sufficiently large, some of the coefficients are driven to zero

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 \quad + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|$$

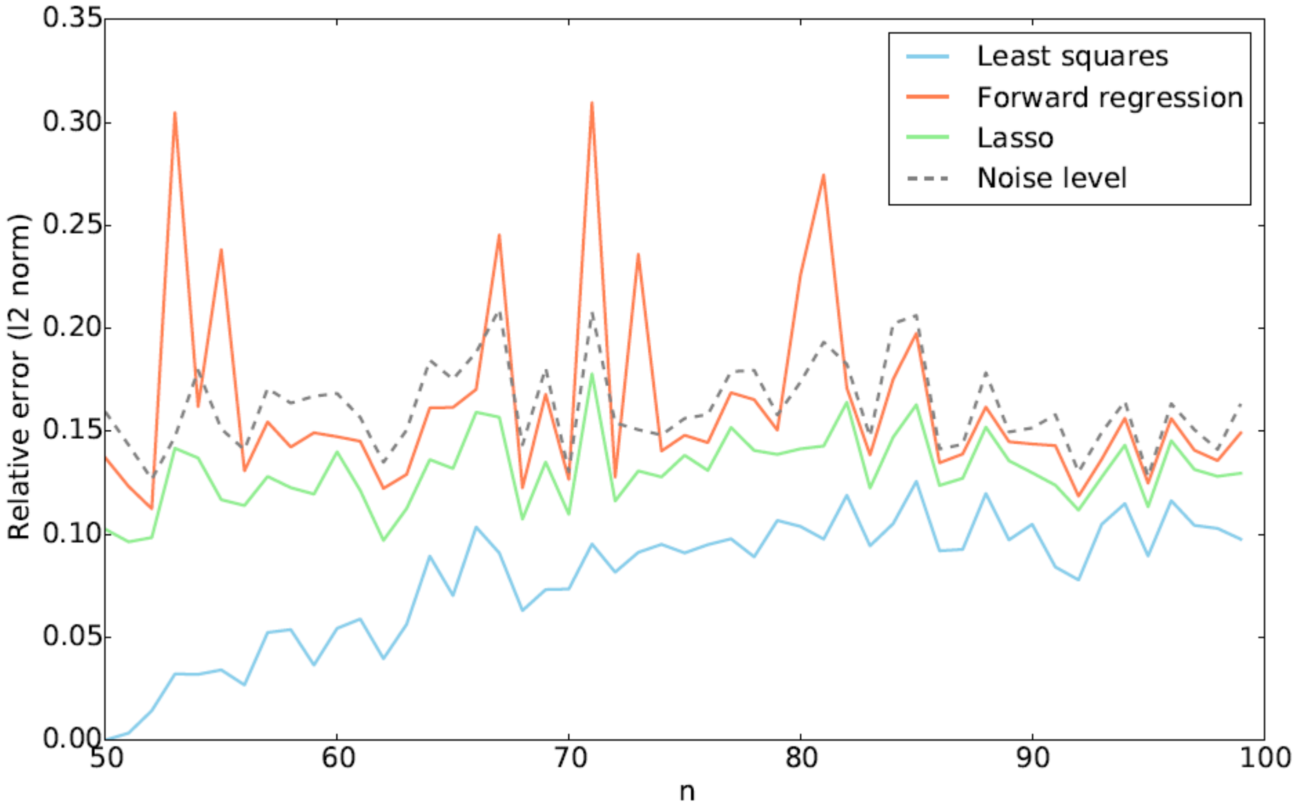


$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

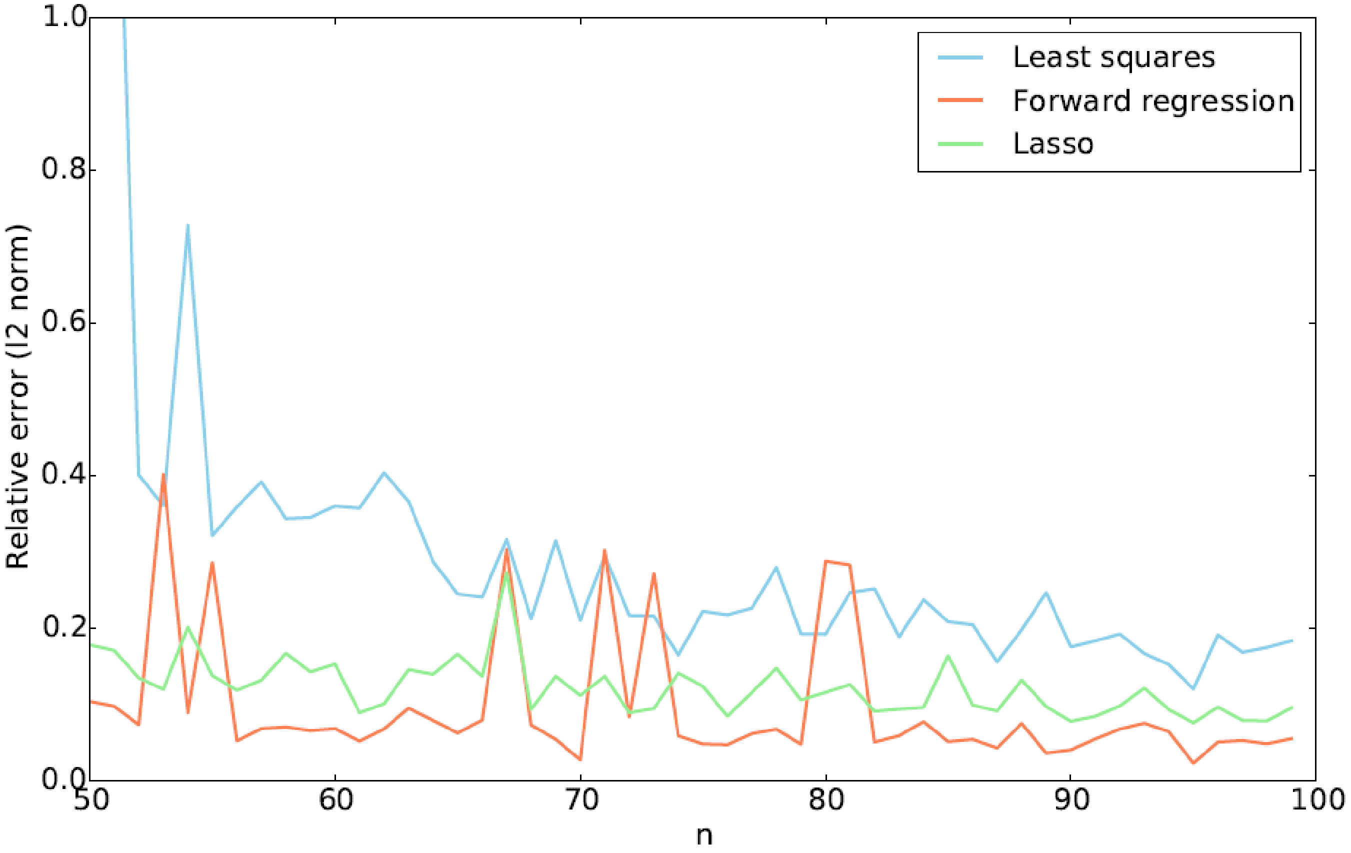


Least Absolute Shrinkage and Selection Operator (LASSO)

Training



Test



Ridge Regression

- Regularized Least Squares

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}.$$

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}_{N \times M}$$

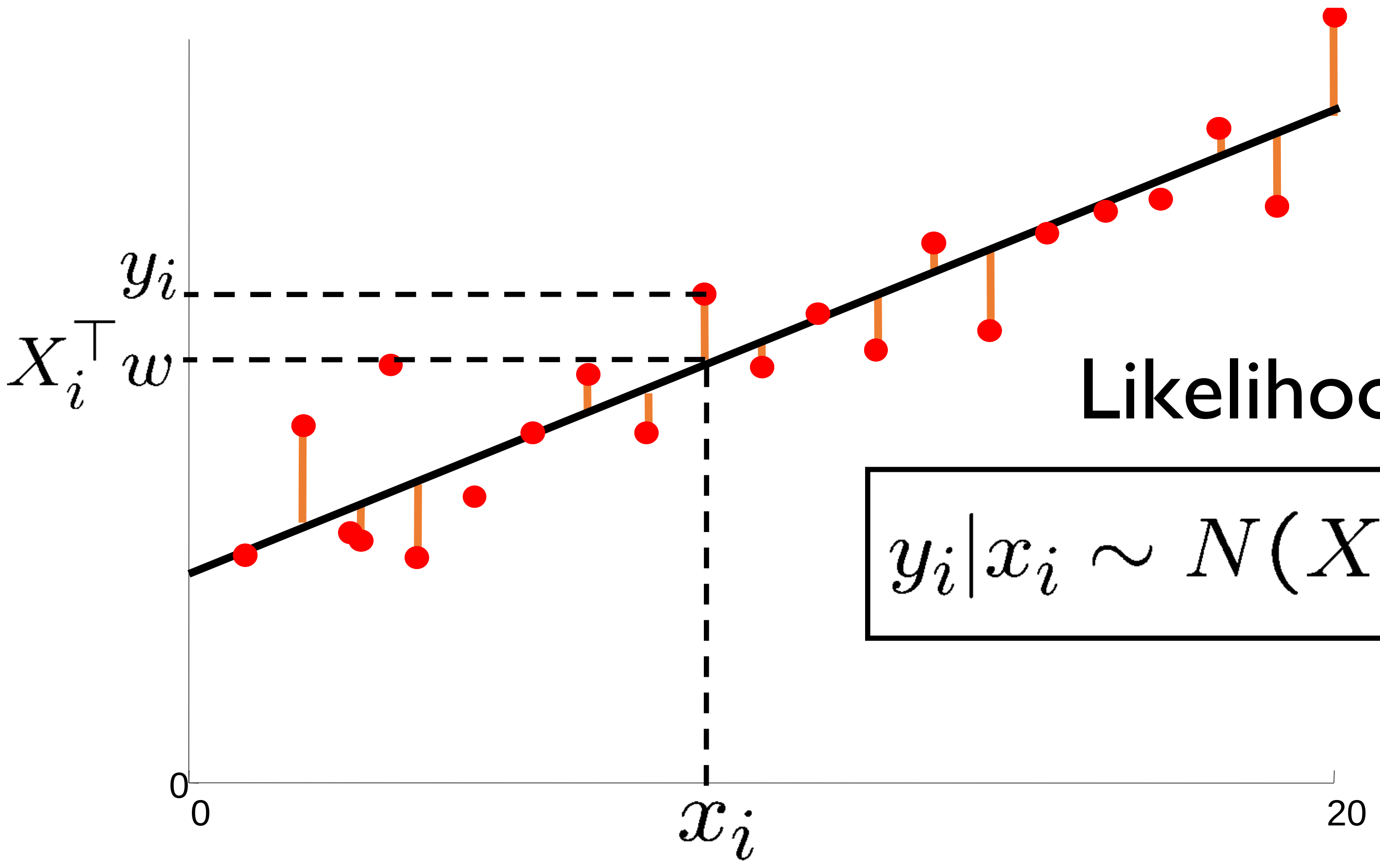
- Show that the regularized least squares solution is

$$\mathbf{w} = \left(\lambda\mathbf{I} + \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}.$$

Stable and unique solution

Probabilistic Interpretation : Least Squares = Maximum likelihood estimation

$$y_i \sim w \cdot x_i + N(0, \sigma^2) = N(w \cdot x_i, \sigma^2)$$



Likelihood

$$\boxed{y_i | x_i \sim N(X_i^\top w, \sigma^2)}$$

$$L = \prod_i \exp -\frac{1}{2\sigma^2}(X_i^\top w - y_i)^2 = \exp -\frac{1}{2\sigma^2}\sum_i (X_i^\top w - y_i)^2$$

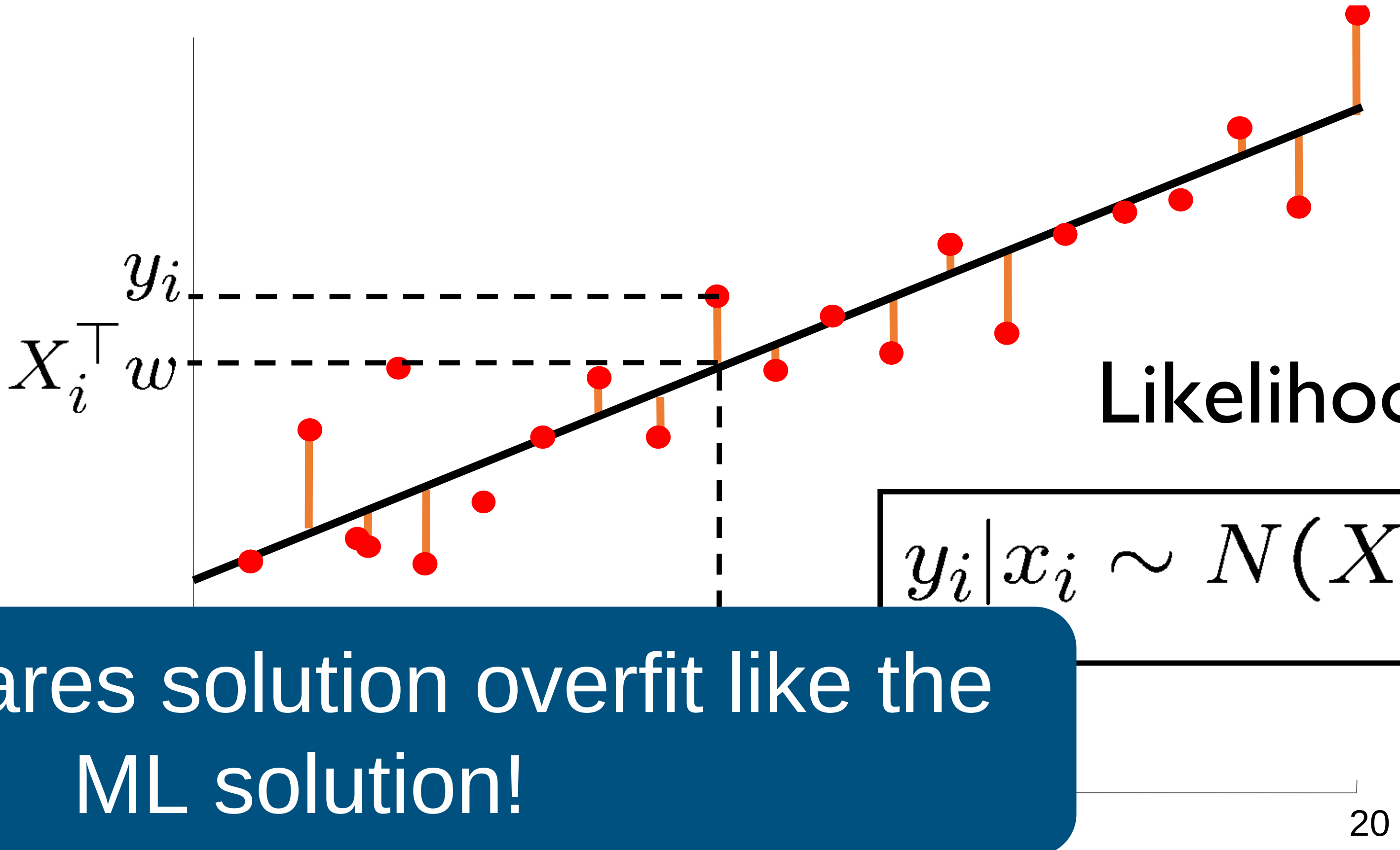
$$\underset{w}{\text{argmax}}\, L = \underset{w}{\text{argmin}}\, E$$

- Similarly Regularized least squares is same as maximum aposteriori estimate assuming p(w) to be a Gaussian : argmax p(yi |w, xi) p(w)

Slide Credits : Vineeth Balasubramanian, Applied Machine Learning, IITH

Probabilistic Interpretation : Least Squares = Maximum likelihood estimation

$$y_i \sim w \cdot x_i + N(0, \sigma^2) = N(w \cdot x_i, \sigma^2)$$



Likelihood

$$\boxed{y_i | x_i \sim N(X_i^\top w, \sigma^2)}$$

Least squares solution overfit like the ML solution!

$$L = \prod_i \exp -\frac{1}{2\sigma^2}(X_i^\top w - y_i)^2 = \exp -\frac{1}{2\sigma^2}\sum_i (X_i^\top w - y_i)^2$$

$$\underset{w}{\text{argmax}}\, L = \underset{w}{\text{argmin}}\, E$$

- Similarly Regularized least squares is same as maximum aposteriori estimate assuming p(w) to be a Gaussian : argmax p(yi |w, xi) p(w)

Slide Credits : Vineeth Balasubramanian, Applied Machine Learning, IITH

Regularized least squares regression
= Maximum Aposteriori Estimate

- Ridge Regression $$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$
- Compute Maximum aposteriori (MAP) estimate
- Prior over parameters $$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp\left\{-\frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}\right\}$$
- Posterior $$p(\mathbf{w}|\mathbf{x}, \mathbf{t}, \alpha, \beta) \propto p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta)p(\mathbf{w}|\alpha).$$
- MAP estimate $$\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\alpha}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}.$$

Unique Solution

$$\mathbf{w} = \left(\lambda\mathbf{I} + \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}. \qquad \lambda = \alpha/\beta.$$

Model Selection

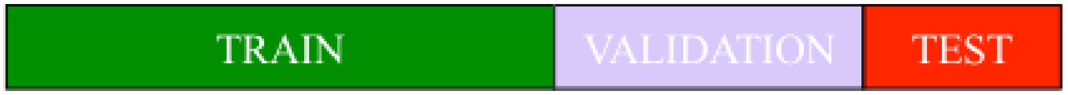
- limiting the number of basis functions in order to avoid over-fitting has the side effect of limiting the flexibility of the model to capture interesting and important trends in the data.
- Regularization allows complex models to be trained on data sets of limited size without severe over-fitting, essentially by limiting the effective model complexity. However, the problem of determining the optimal model complexity is then shifted from one of finding the appropriate number of basis functions to one of determining a suitable value of the regularization coefficient λ.

- What happens when you minimizes the regularized error function with respect to weight vector w and regularization coefficient λ ?

Regularized Least Squares : Cross Validation

How to choose λ ?
Use validation data!

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$



1. Training set is a set of examples used for learning a model parameters (e.g., weight vector w in linear regression

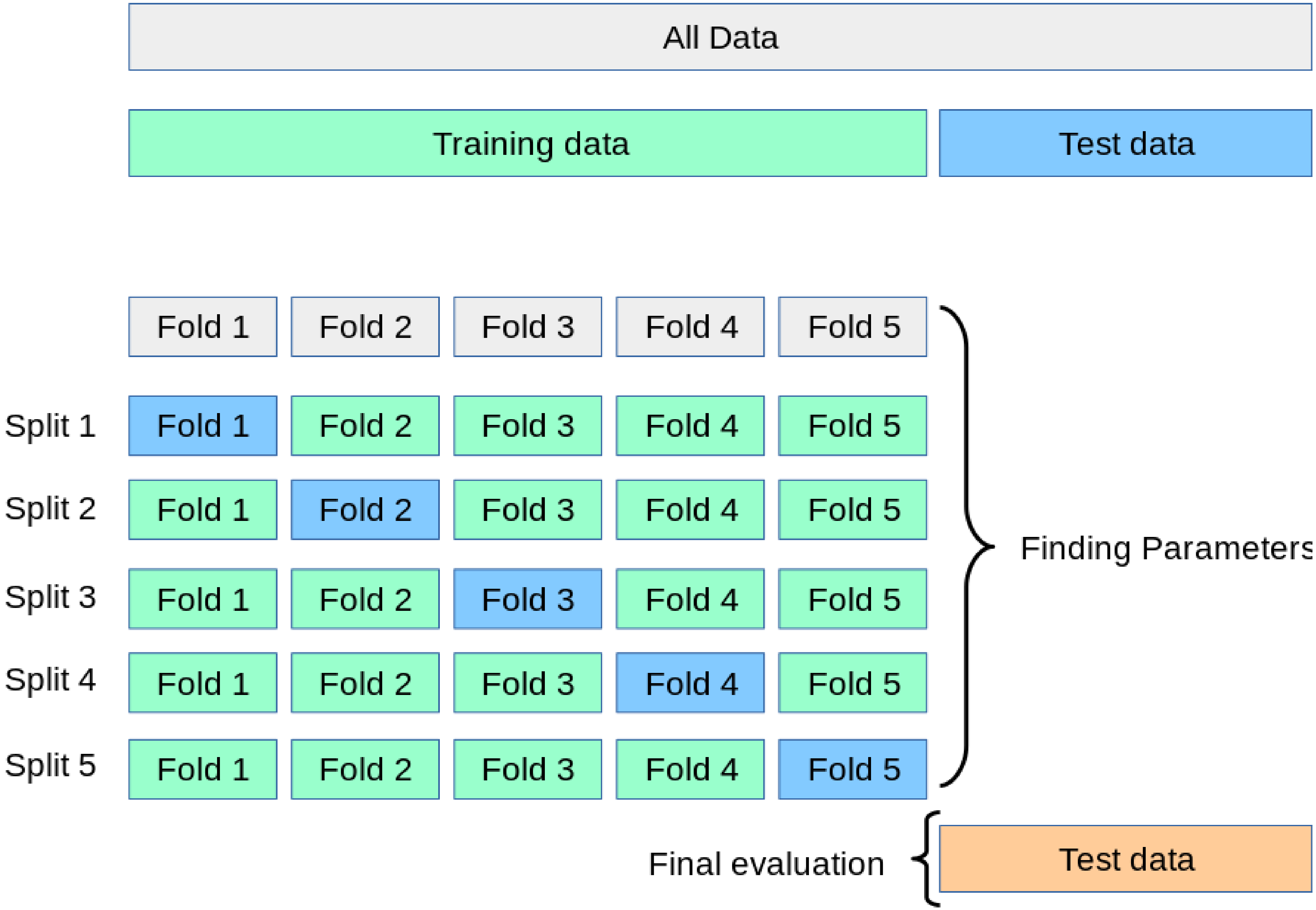
2. Validation set is a set of examples that cannot be used for learning the model parameter but can help tune model hyper-parameters e.g Regularization constant in LR. Validation helps control overfitting.

3. Test set is used to assess the performance of the final model and provide an estimation of the test error.

Example: Split the data randomly into 60% for training, 20% for validation and 20% for testing.

Note: Dont use the test set to further tune the parameters or revise the model.

Cross Validation



All Data

Training data

Test data

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

Finding Parameters

Final evaluation

Test data

https://scikit-learn.org/stable/modules/cross_validation.html

Linear Regression

- Parameter Estimation
  - Maximum Likelihood
  - Maximum Aposteriori
- Hyper-parameter Estimation
  - Cross-validation
- Prediction

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}).$$

Bayesian Decision Theory

- decision theory that, when combined with probability theory, allows us to make optimal decisions
- Suppose we have an input vector x together with a corresponding vector t of target variables, and our goal is to predict t given a new value for x.
- Determination of p(x, t) from a set of training data is an example of inference
- Decision stage consists of choosing a specific estimate y(x) of the value of t for each input x, we incur a loss L(t, y(x)).

$$\mathbb{E}[L] = \iint L(t, y(\mathbf{x}))p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t. \qquad \mathbb{E}[L] = \iint \{y(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t.$$

$$\frac{\delta \mathbb{E}[L]}{\delta y(\mathbf{x})} = 2\int \{y(\mathbf{x}) - t\}p(\mathbf{x}, t)\,\mathrm{d}t = 0. \qquad y(\mathbf{x}) = \frac{\displaystyle\int tp(\mathbf{x}, t)\,\mathrm{d}t}{p(\mathbf{x})} = \int tp(t|\mathbf{x})\,\mathrm{d}t = \mathbb{E}_t[t|\mathbf{x}]$$

$$\mathbf{y}(\mathbf{x}) = \mathbb{E}_t[\mathbf{t}|\mathbf{x}].$$

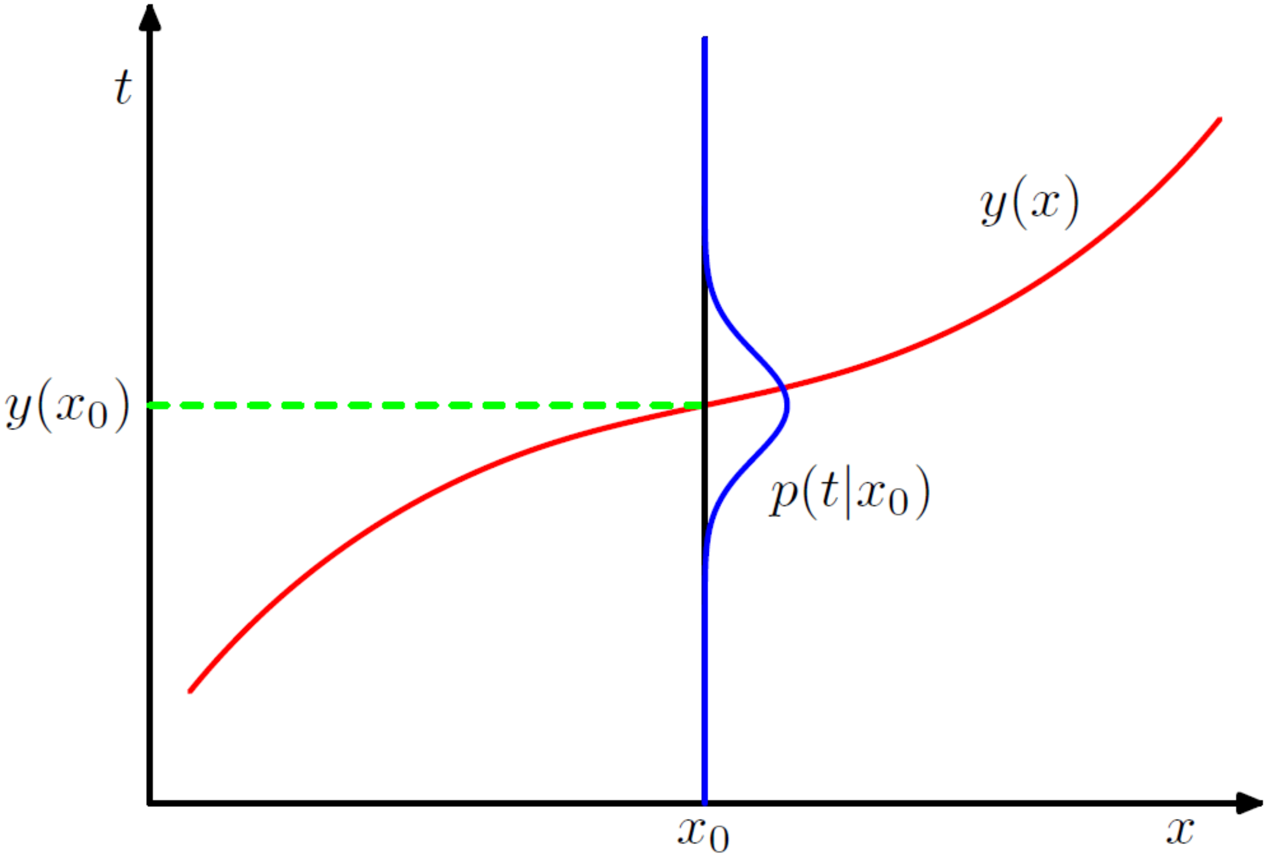
Bayesian Decision Theory

- If we assume a squared loss function, then the optimal prediction, for a new value of x, will be given by the conditional mean of the target variable.

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}).$$

$$\mathbb{E}[t|\mathbf{x}] = \int tp(t|\mathbf{x})\,\mathrm{d}t = y(\mathbf{x}, \mathbf{w}).$$



Bias Variance Decomposition

Given the actual conditional distribution *p*(*t*/**x**), **and** p(**x,t**) = p(t|x) p(x). For squared loss function, optimal prediction is given by the Bayesian estimate which in this case is the conditional expectation, which we denote by *h*(**x**)

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x})\,\mathrm{d}t.$$

Expected squared difference between *y*(**x**;*D*) and the regression function *h*(**x**)

$$\begin{aligned}
&\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2\right] \\
&\quad = \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right]}_{\text{variance}}.
\end{aligned}$$

Bias- Variance Decomposition

- Bias Error
  - Bias are the simplifying assumptions made by a model to make the target function easier to learn.
  - **Low Bias:** Suggests less assumptions about the form of the target function.
  - **High-Bias**: Suggests more assumptions about the form of the target function.

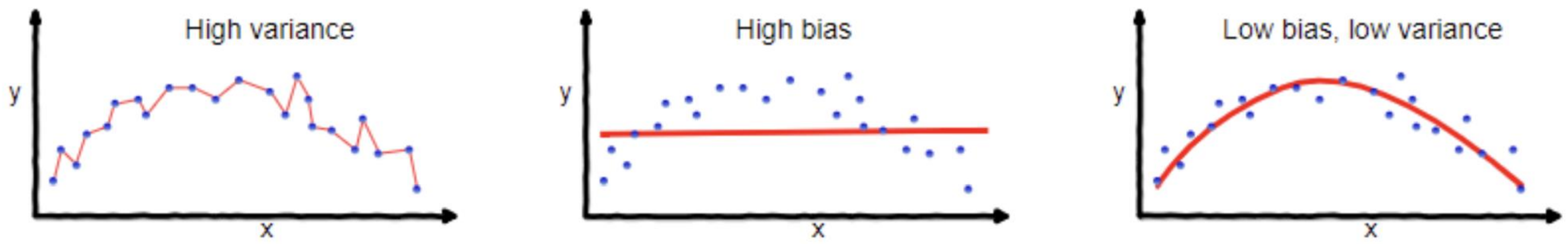
$$
\begin{aligned}
&\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D})-h(\mathbf{x})\}^2\right] \\
&\quad = \underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]-h(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D})-\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right]}_{\text{variance}}.
\end{aligned}
$$

There is a trade-off between bias and variance, with very flexible models having low bias and high variance, and relatively rigid models having high bias and low variance.

Bias- Variance Decomposition

- **Variance Error**
  - Variance is the amount that the estimate of the target function will change if different training data was used.
  - **Low Variance**: Suggests small changes to the estimate of the target function with changes to the training dataset.
  - **High Variance**: Suggests large changes to the estimate of the target function with changes to the training dataset.

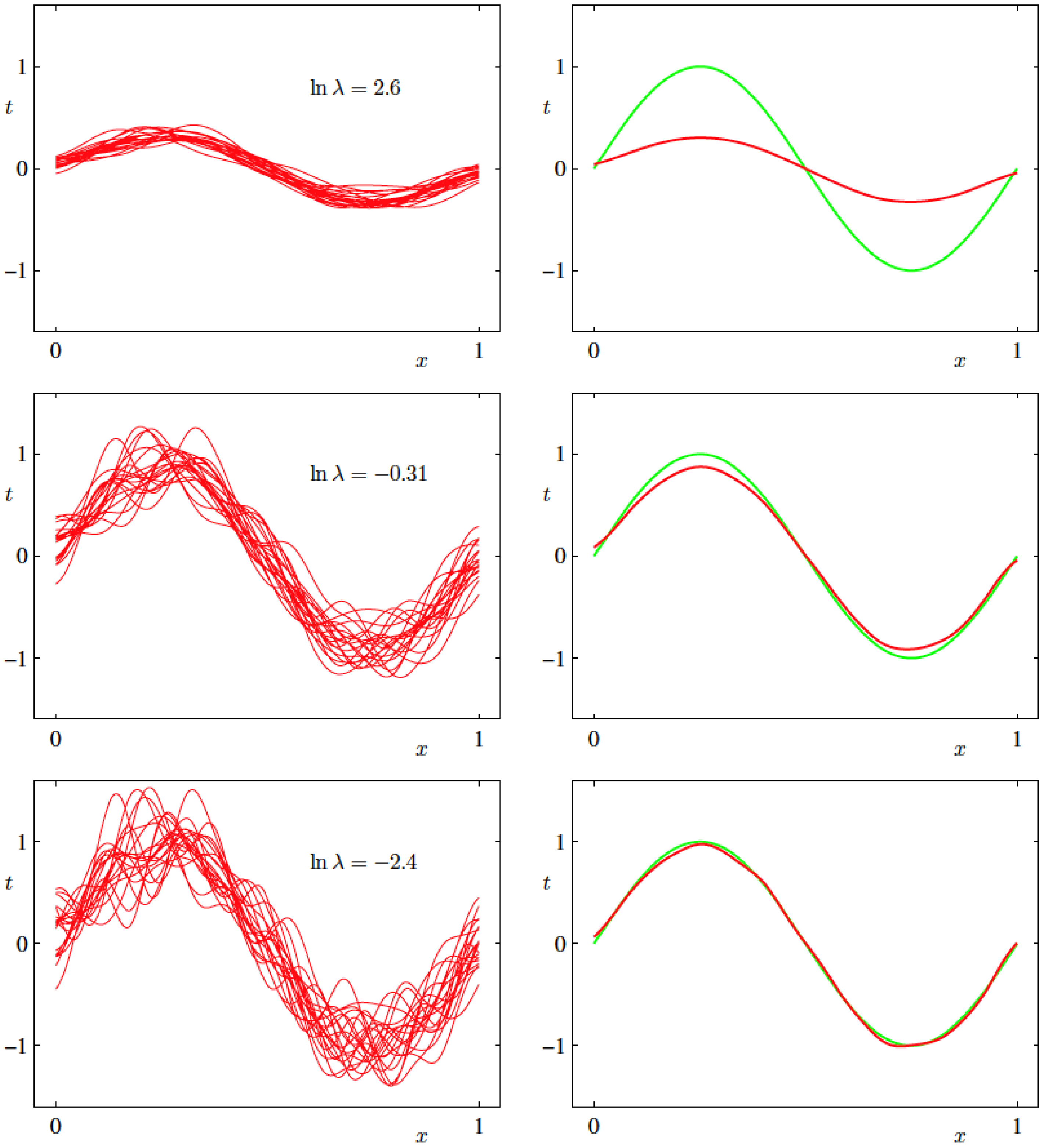
$$\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D})-h(\mathbf{x})\}^2\right]$$
$$=\underbrace{\{\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]-h(\mathbf{x})\}^2}_{(\text{bias})^2}+\underbrace{\mathbb{E}_{\mathcal{D}}\left[\{y(\mathbf{x};\mathcal{D})-\mathbb{E}_{\mathcal{D}}[y(\mathbf{x};\mathcal{D})]\}^2\right]}_{\text{variance}}.$$



overfitting **underfitting** **Good balance**

•**Linear** machine learning algorithms often have a high bias but a low variance.
•**Nonlinear** machine learning algorithms often have a low bias but a high variance.

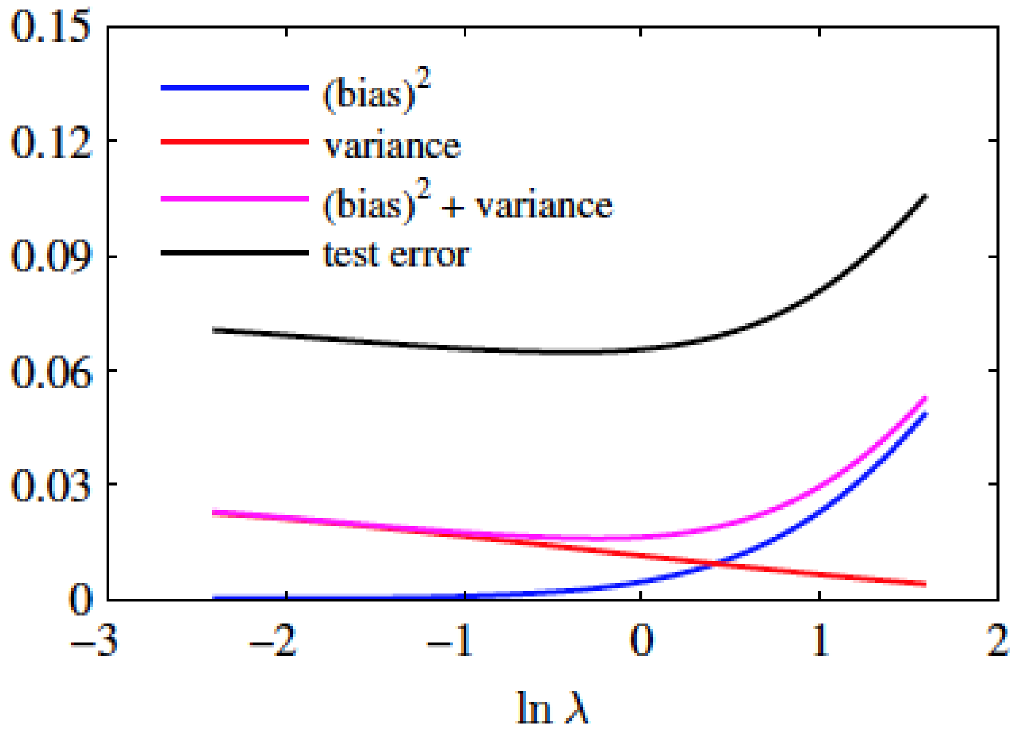
Bias Variance Decomposition



$$h(x) = \sin(2\pi x). \qquad \overline{y}(x) = \frac{1}{L}\sum_{l=1}^{L} y^{(l)}(x)$$

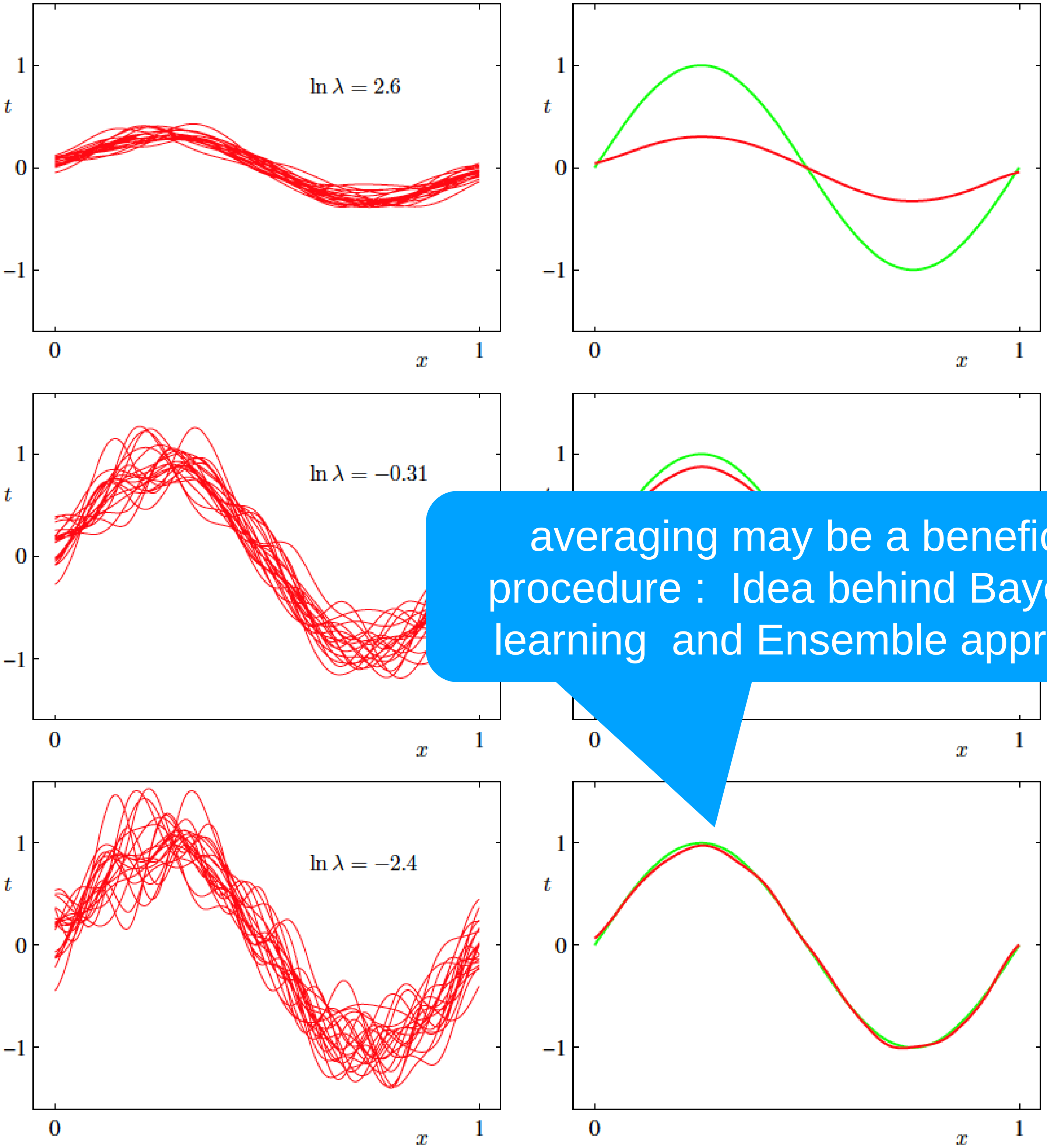
$$(\text{bias})^2 = \frac{1}{N}\sum_{n=1}^{N}\left\{\overline{y}(x_n) - h(x_n)\right\}^2$$

$$\text{variance} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{L}\sum_{l=1}^{L}\left\{y^{(l)}(x_n) - \overline{y}(x_n)\right\}^2$$



L = 100 data sets, each having *N* = 25 data points,

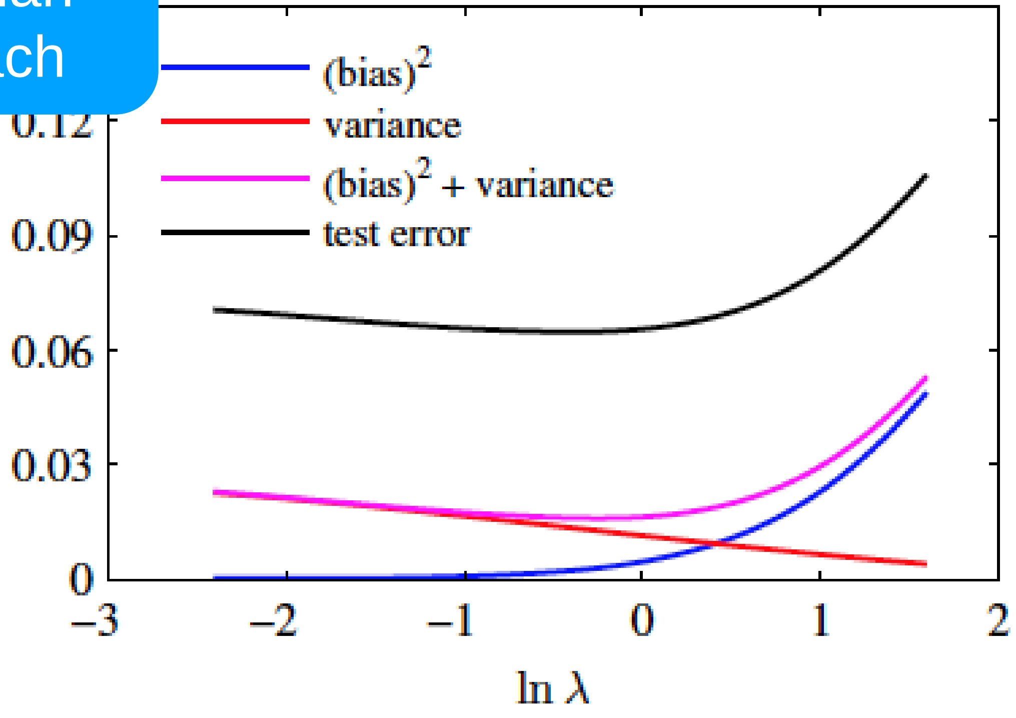
Bias Variance Decomposition



$$h(x) = \sin(2\pi x).$$

L = 100 data sets,
each having *N* = 25
data points,

averaging may be a beneficial procedure : Idea behind Bayesian learning and Ensemble approach



Bias Variance Tradeoff

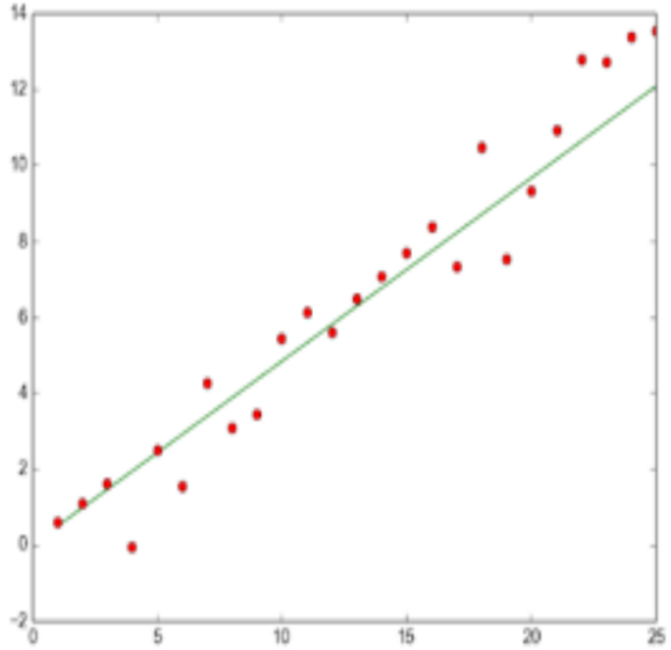
- Dimensionality reduction and feature selection can decrease variance by simplifying models.
- Similarly, a larger training set tends to decrease variance.
- Adding features (predictors) tends to decrease bias, at the expense of introducing additional variance.
- linear and Generalized linear models can be regularized to decrease their variance at the cost of increasing their bias

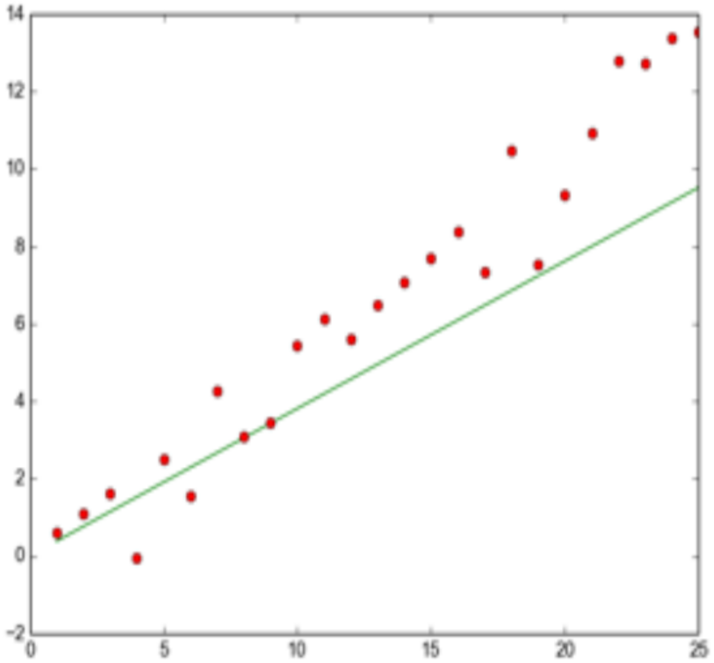
Bayesian Linear Regression

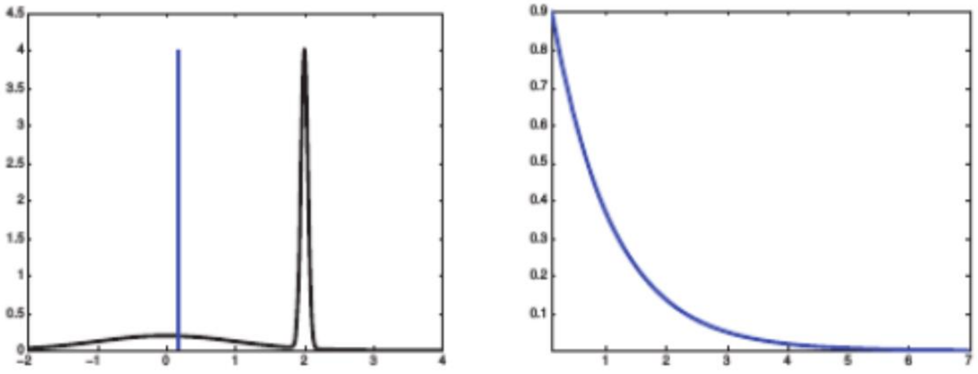
$$Prediction : p(y_*|x_*) = \int p(y_*|x_*, w)p(w|D)dw$$

$$p(w|D) \propto p(D|w) \quad p(w)$$

Posterior $\propto$ Likelihood Prior

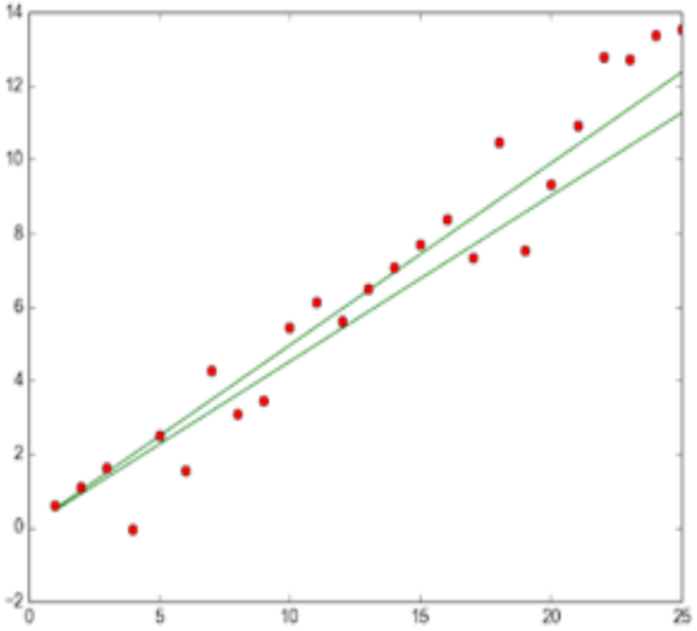


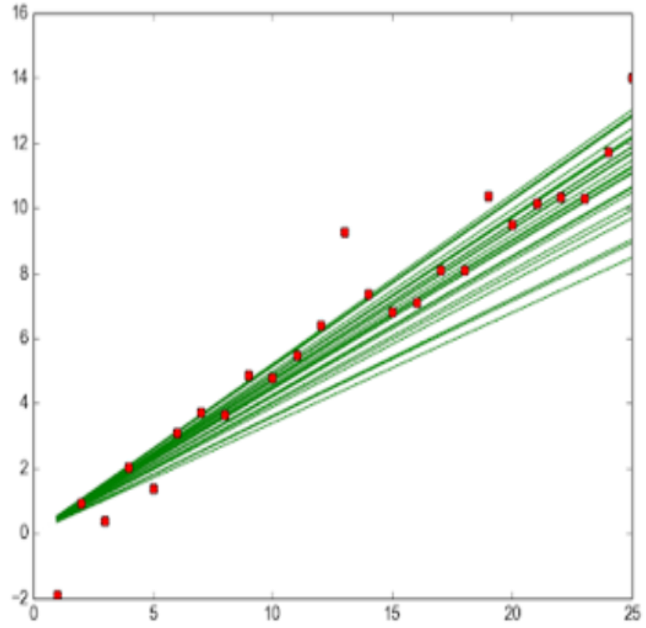




(a)

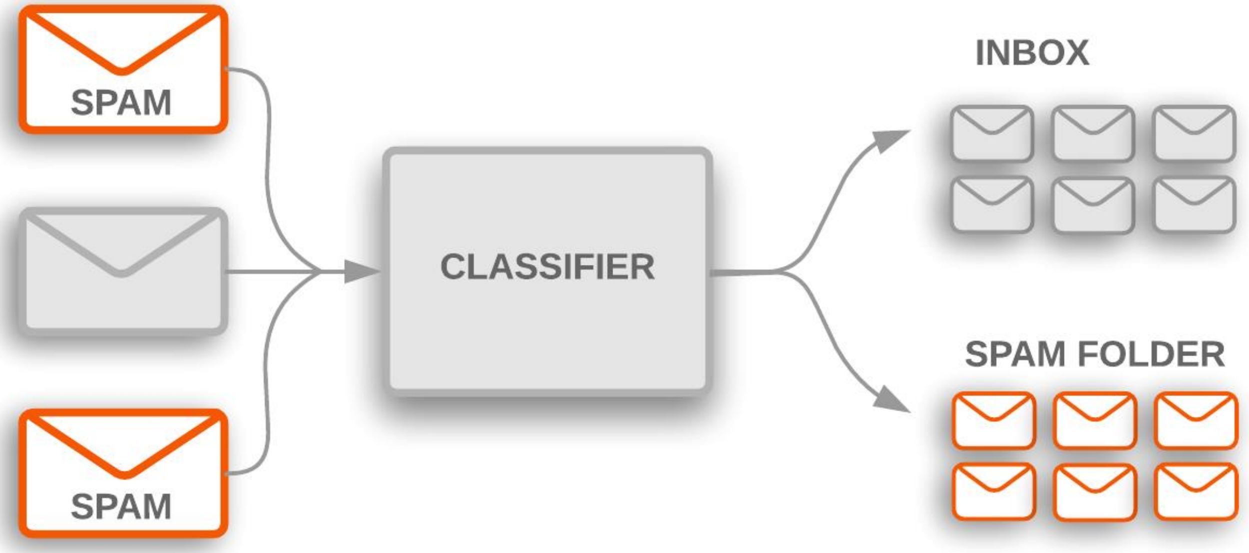
(b)



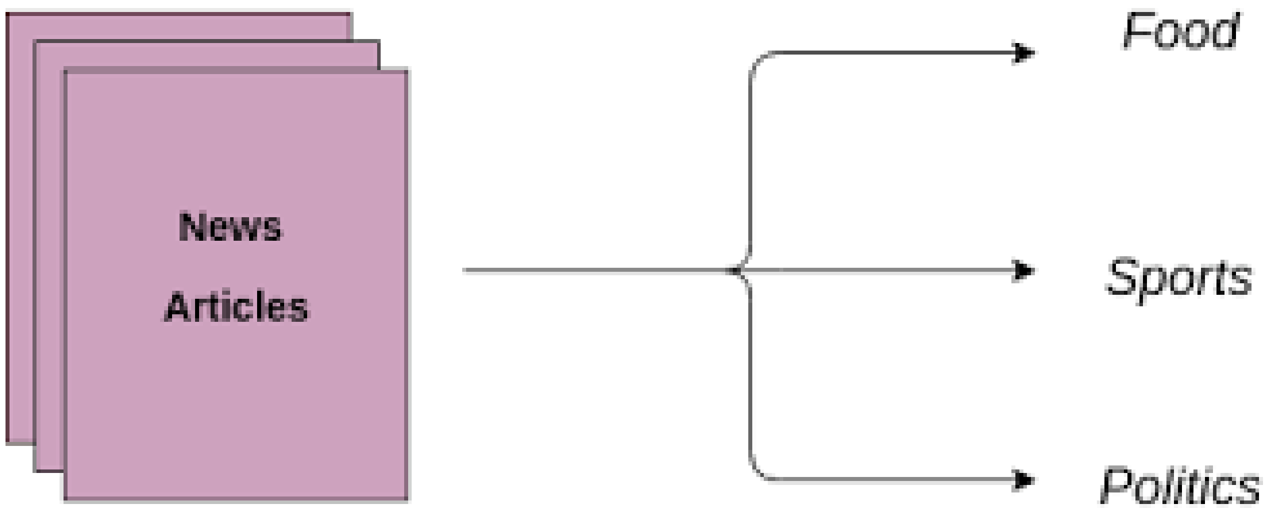


Supervised Learning : Classification

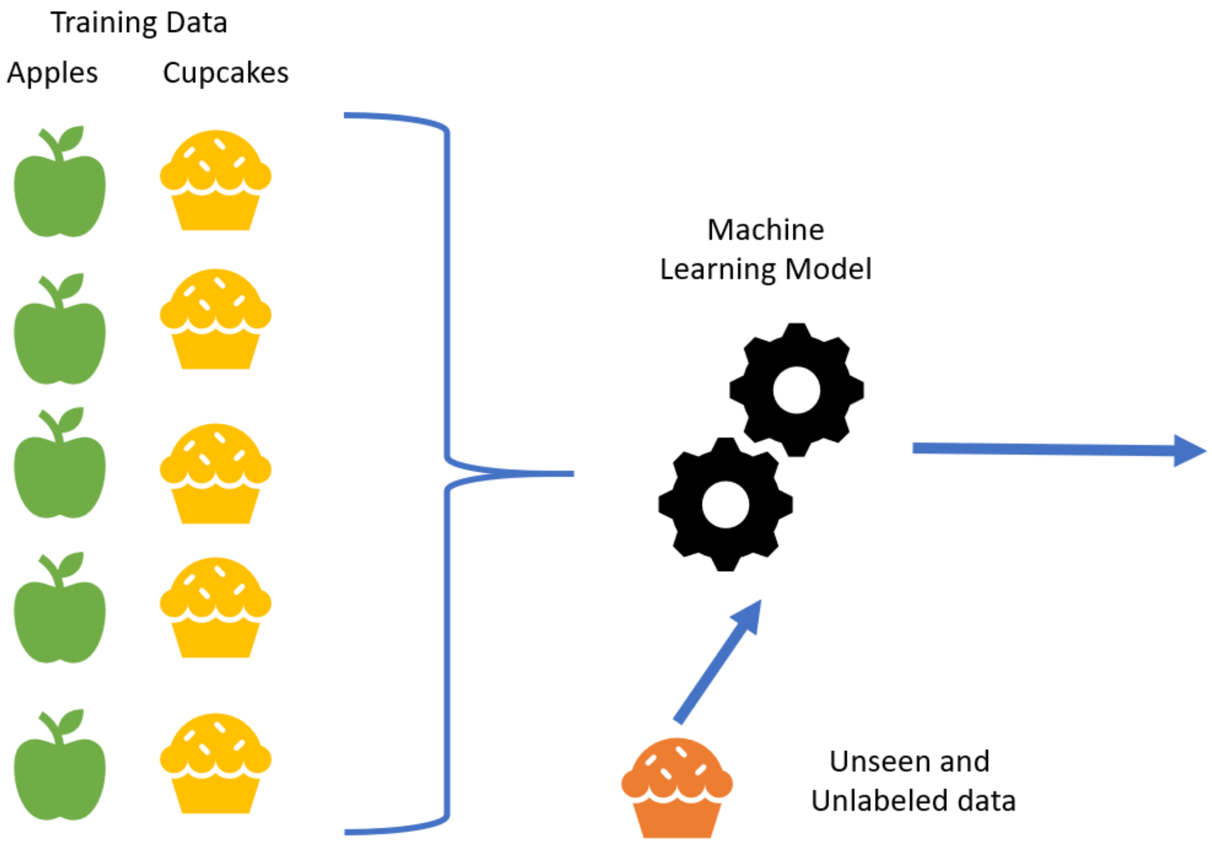
- Binary classification : $y = \{0,1\}$
- Multiclass classification : $y = \{1,2,\ldots K\}$



p(y|x)?



Classification : Linear Models



Training Data

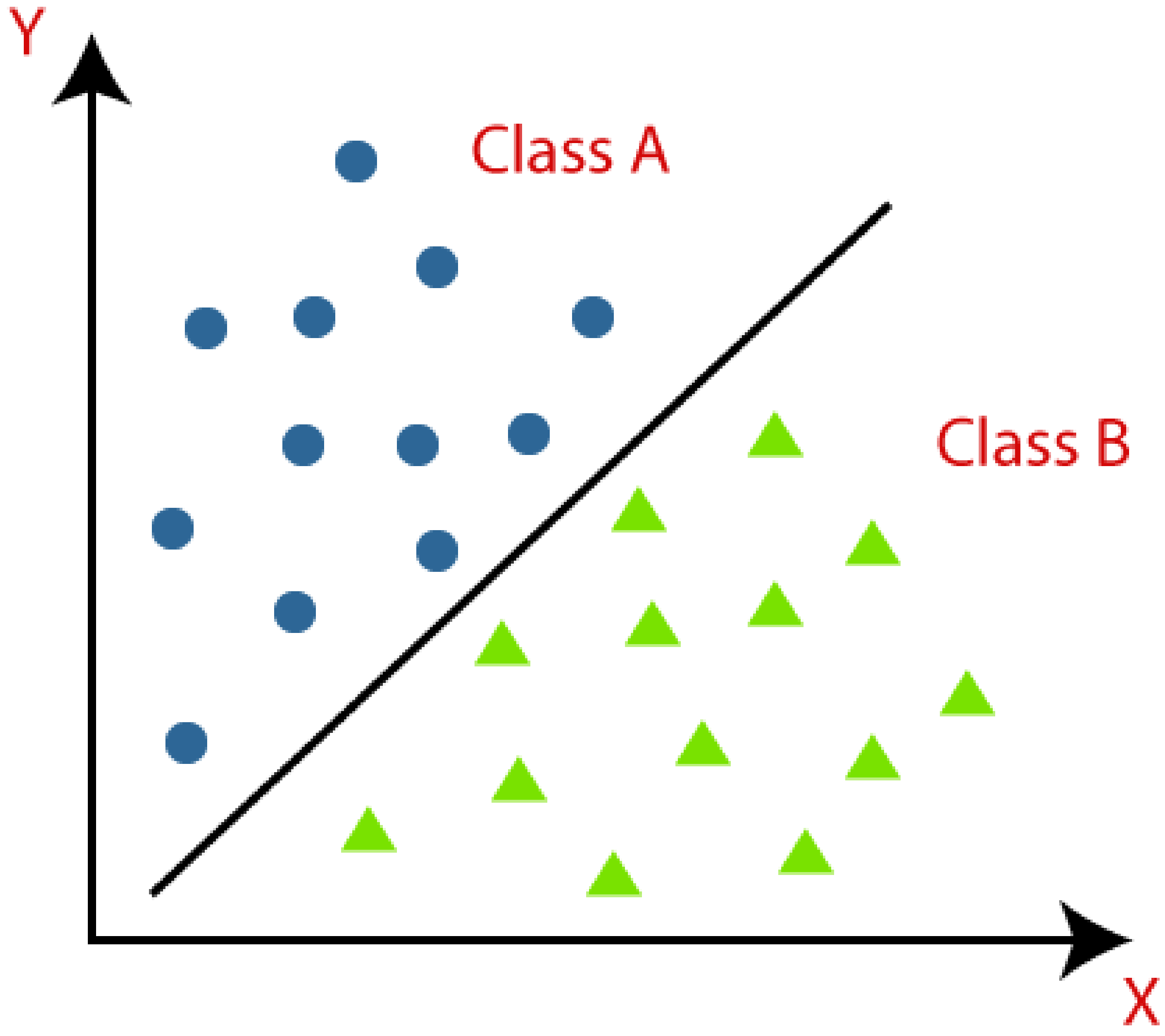
Apples

Cupcakes

Machine Learning Model

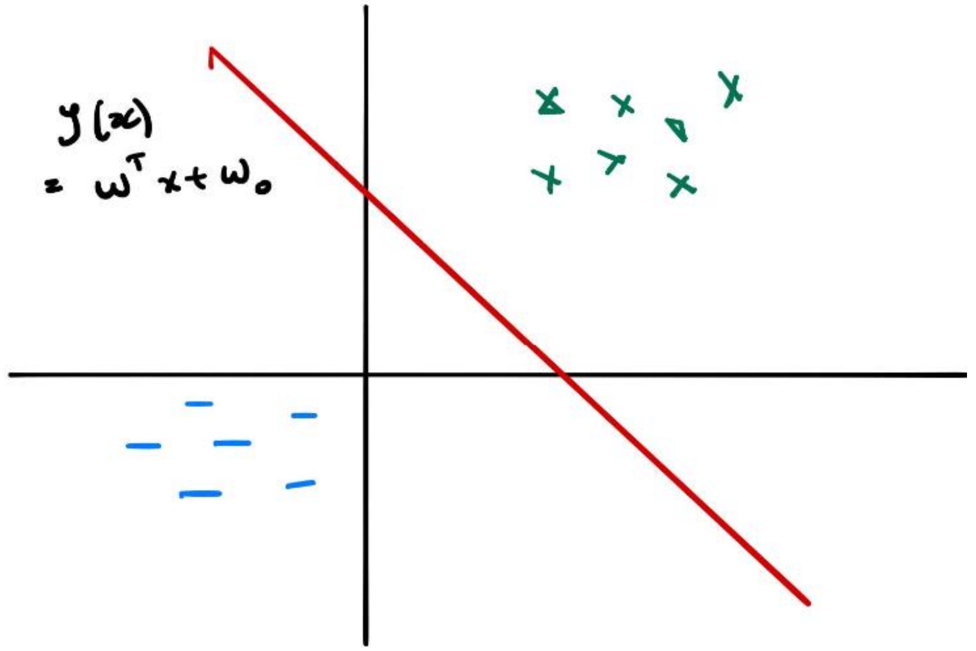
Unseen and Unlabeled data

Class : Cupcake



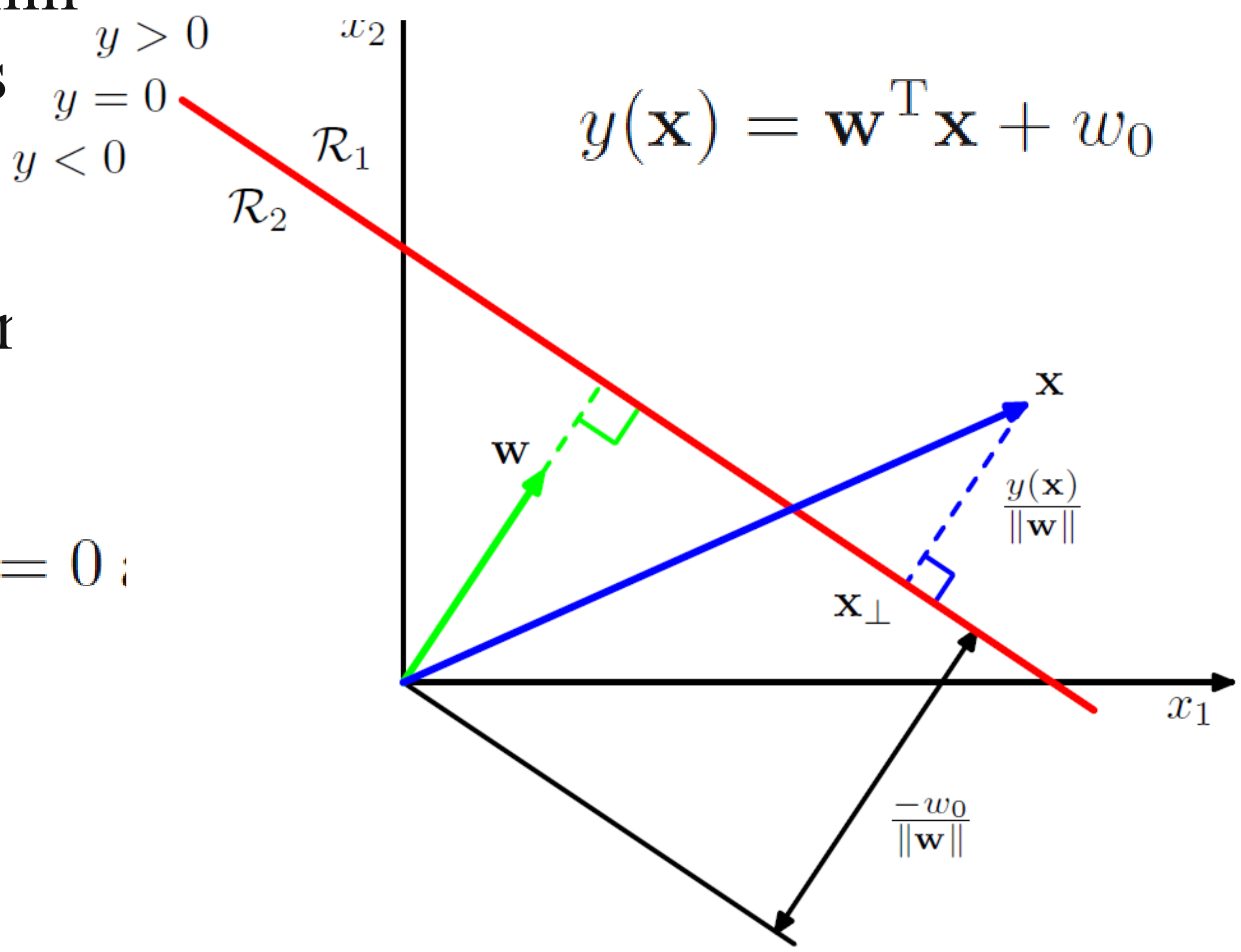
Discriminant function

input vector **x** is assigned to class *C*1 if *y*(**x**) > 0 and to class *C*2 otherwise



decision boundary is therefore defined by the relation *y*(**x**) = 0

w is orthogonal to every vector lying within the decision surface, and so **w** determines the orientation of the decision surface.



two points **X**A and **X**B both of which lie o decision surface.

$$y(\mathbf{x}_A) = y(\mathbf{x}_B) = 0, \text{ we have } \mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) = 0$$

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}.$$

$$r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}.$$

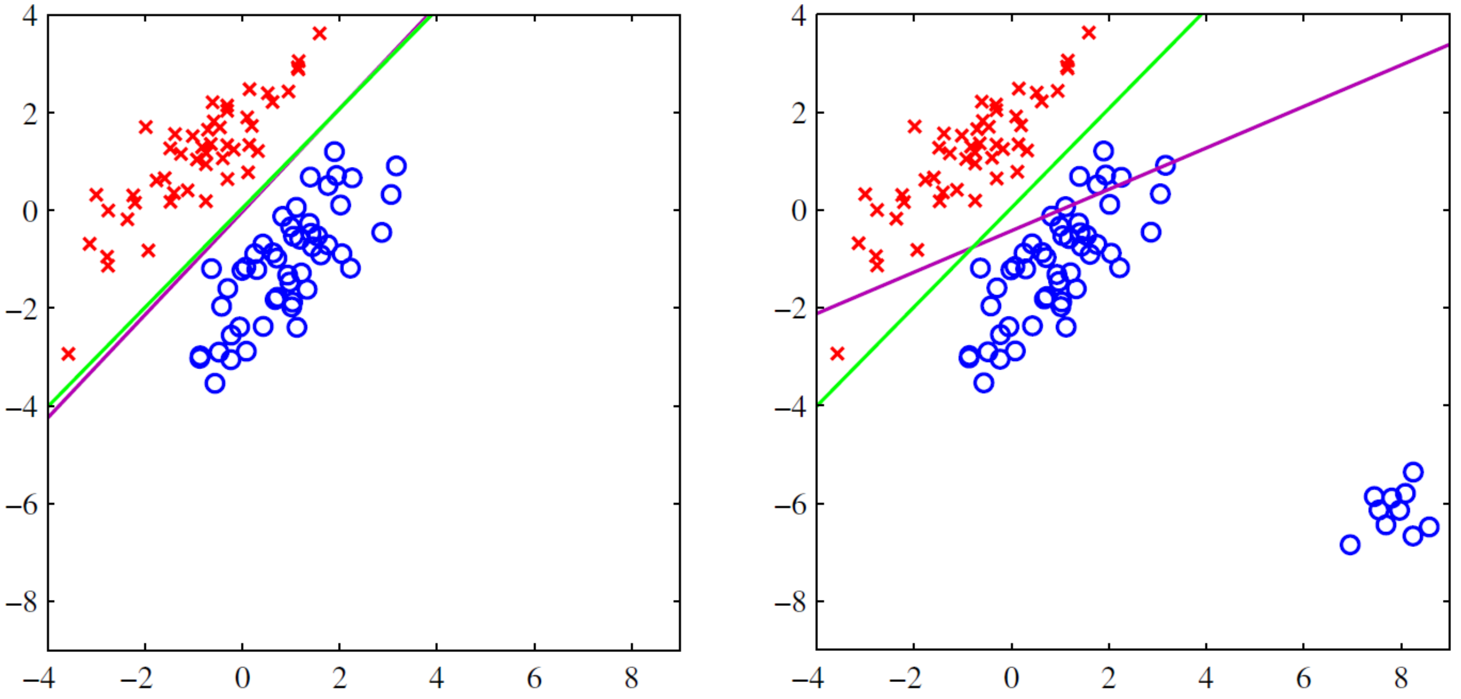
$$y(\mathbf{x}_\perp) = \mathbf{w}^T\mathbf{x}_\perp + w_0 = 0,$$

Least squares classification

- Loss function $$E_D(\widetilde{\mathbf{W}}) = \frac{1}{2}\mathrm{Tr}\left\{(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})^{\mathrm{T}}(\widetilde{\mathbf{X}}\widetilde{\mathbf{W}} - \mathbf{T})\right\}.$$

- Solution $$\widetilde{\mathbf{W}} = (\widetilde{\mathbf{X}}^{\mathrm{T}}\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^{\mathrm{T}}\mathbf{T} = \widetilde{\mathbf{X}}^{\dagger}\mathbf{T}$$

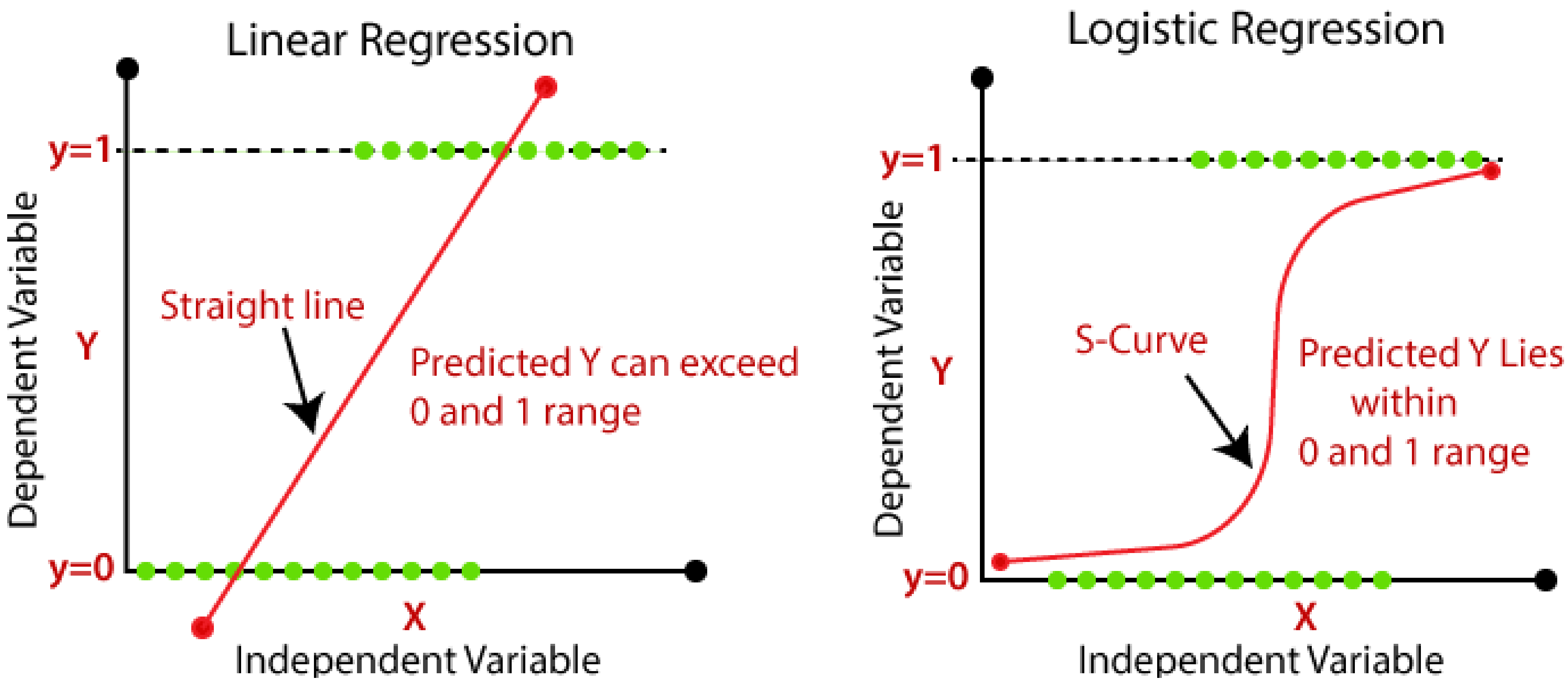
- least-squares solutions lack robustness to outliers



Least squares corresponds to maximum likelihood under the assumption of a **Gaussian conditional** distribution, whereas binary target vectors clearly have a distribution that is far from Gaussian.

Linear regression to Logistic regression

- Y takes value 0 or 1



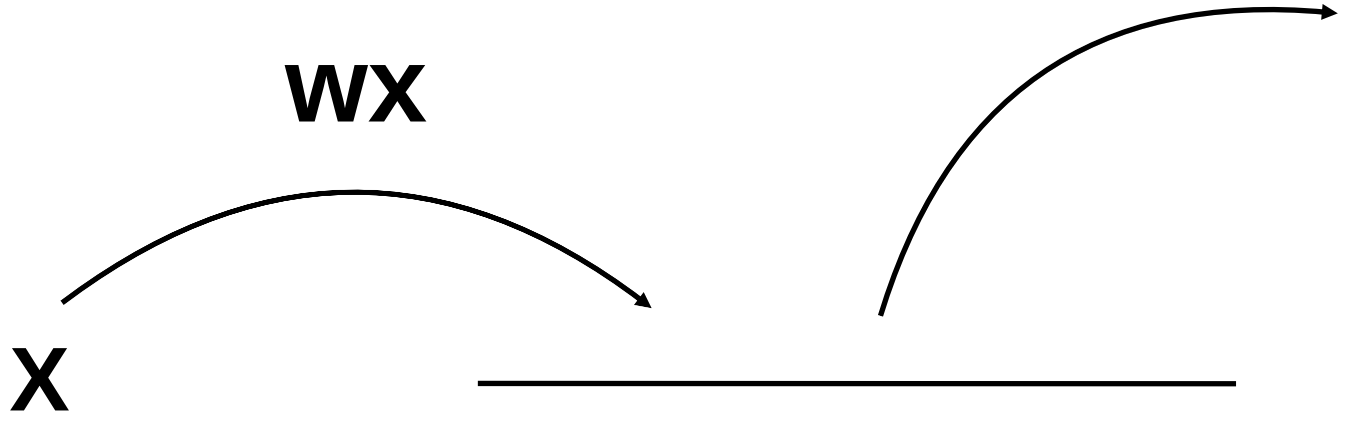
Logistic Regression

- A discriminative approach which directly models p(y|x)
- To predict an outcome variable that is categorical from one or more categorical or continuous predictor variables.

- Let X be the data instance, and Y be the class label {0,1} : Model P(Y|X) directly using a Sigmoid function:

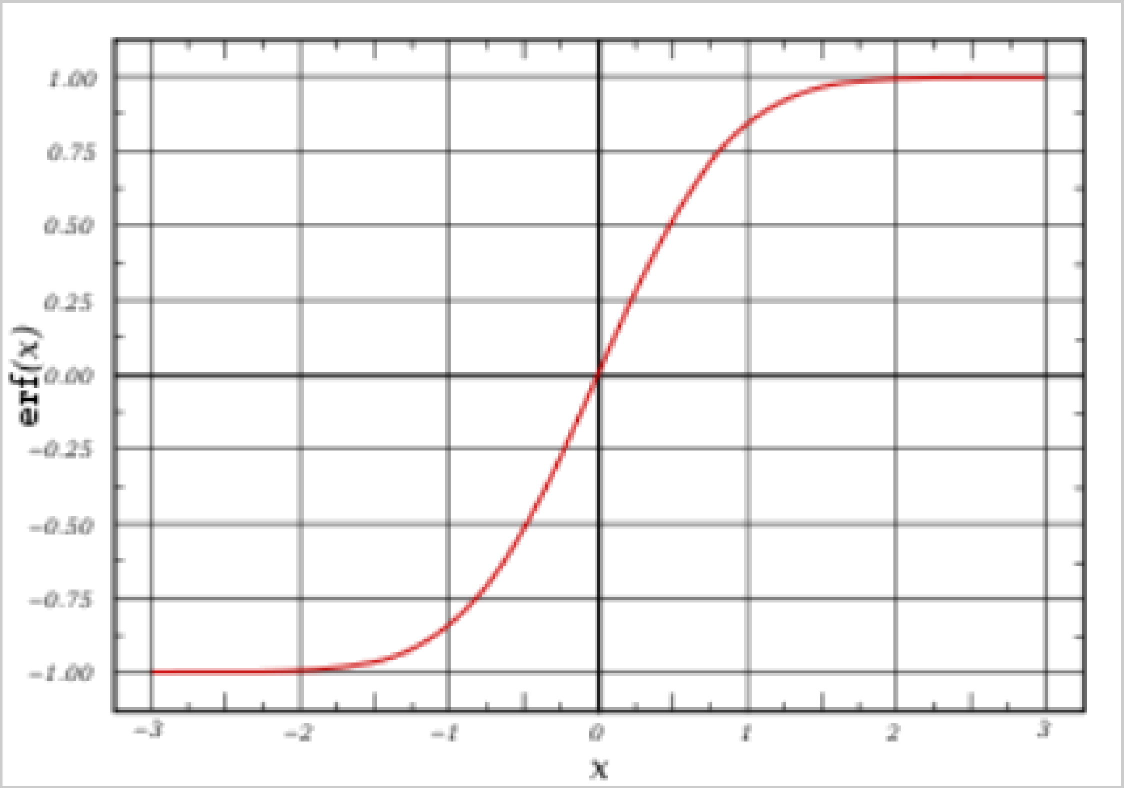
Logistic Sigmoid : $P(Y=1\,|\,\mathbf{X}) = \dfrac{1}{1+e^{-\mathbf{wx}}}$

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$



WX

X



Logistic Regression

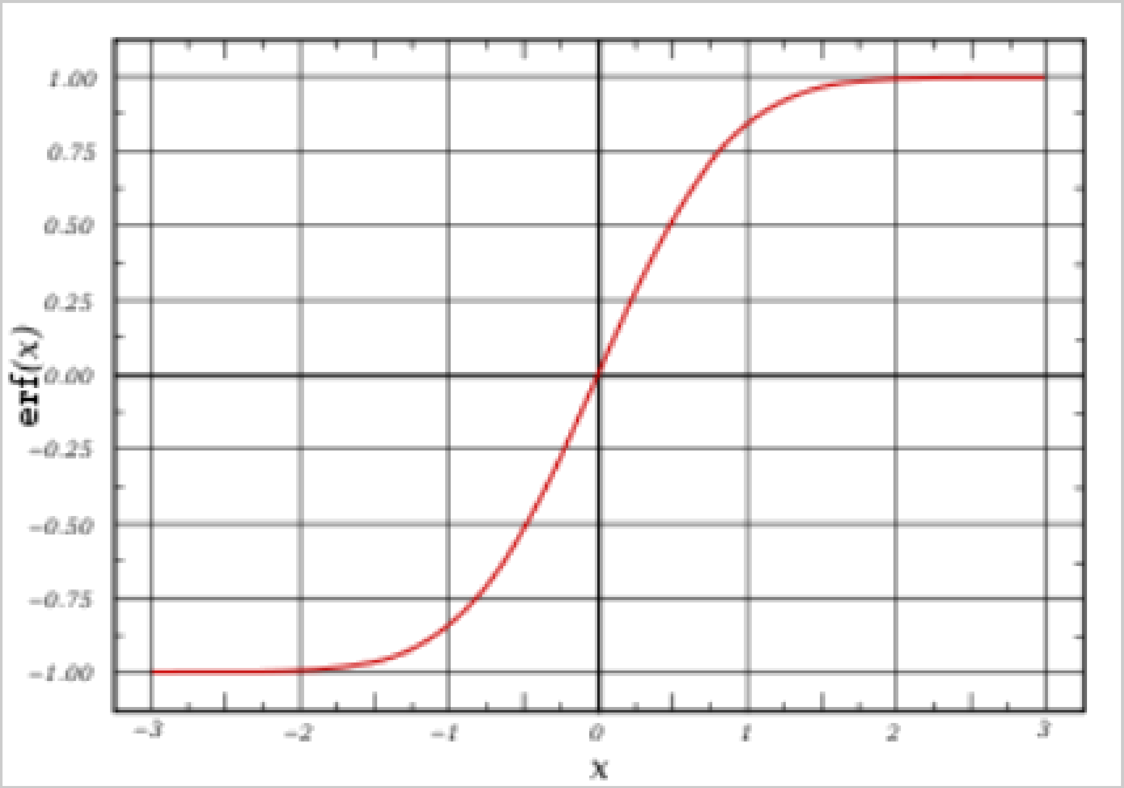
- To predict an outcome variable that is categorical from one or more categorical or continuous predictor variables.

- Let X be the data instance, and Y be the class label:
  Model P(Y|X) directly using a Sigmoid function: Logistic function

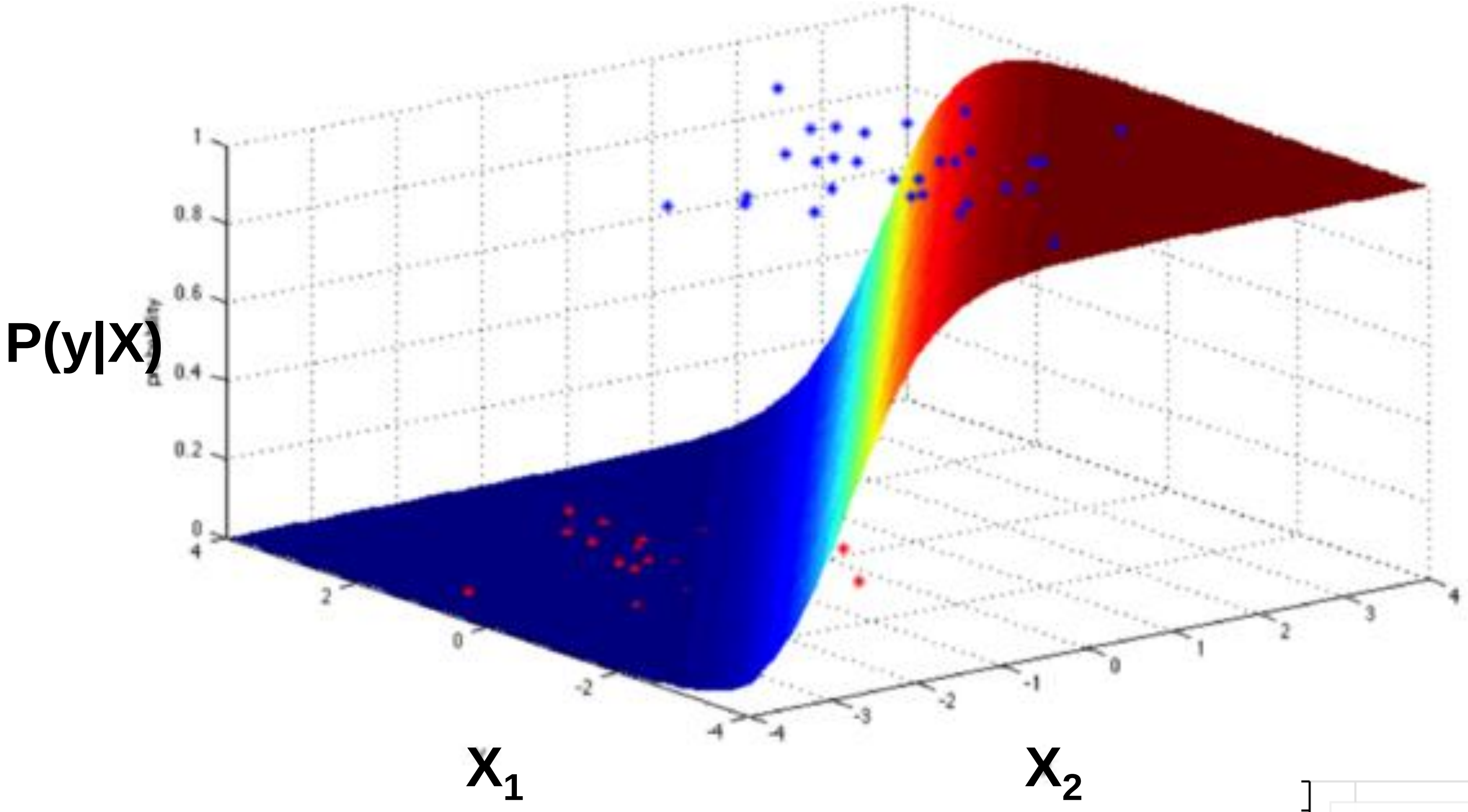
$$P(Y=1|\mathbf{X}) = \frac{1}{1+e^{-\mathbf{wx}}}$$

[HW] Find derivative of s(w) = p(y=1|X)!

$$\Phi(a) = \int_{-\infty}^{a} \mathcal{N}(\theta|0,1)\,\mathrm{d}\theta$$



Logistic Regression

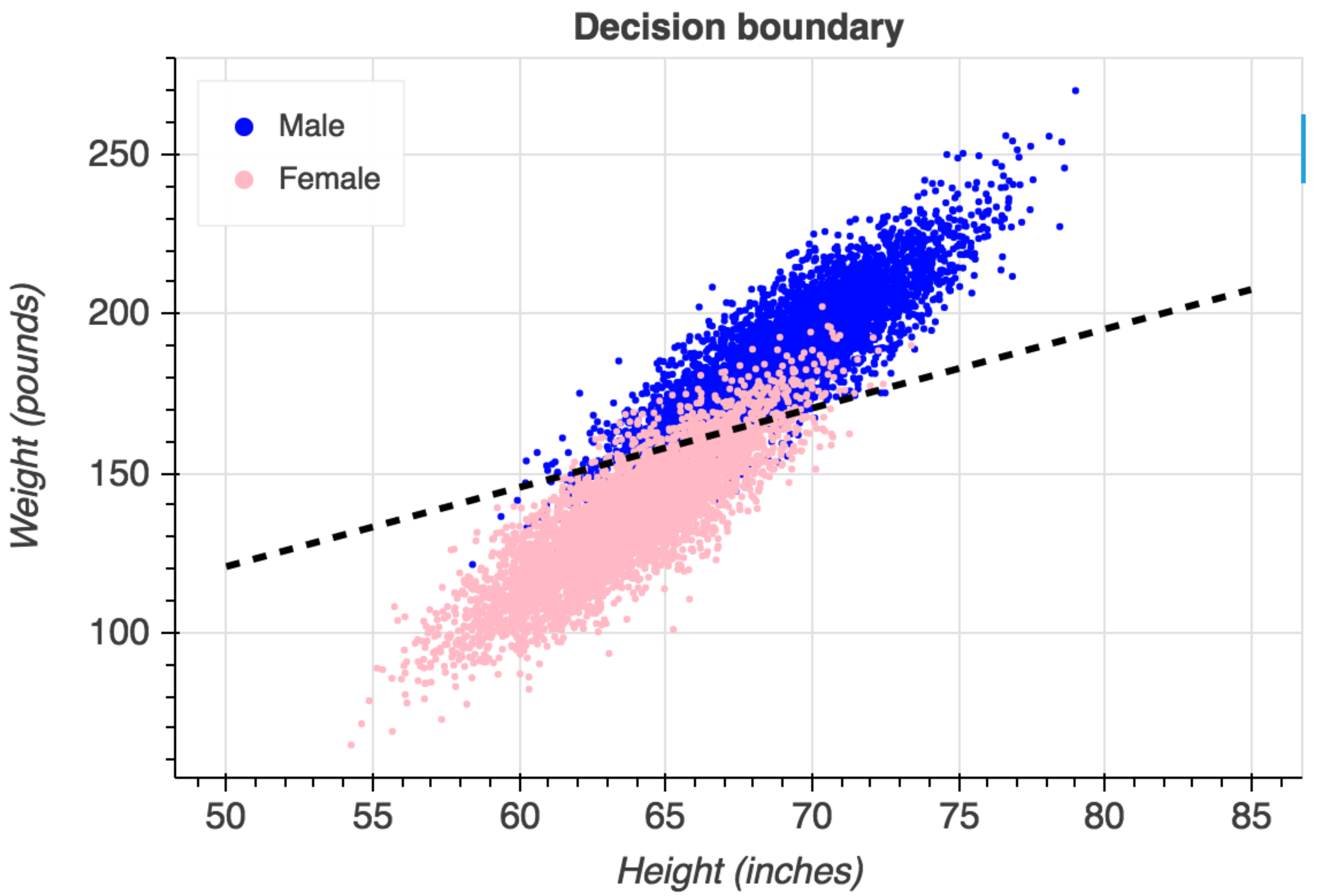


Discriminant Functions

$$P(Y=1\,|\,\mathbf{X}) = \frac{1}{1+e^{-\mathbf{wx}}}$$

Decision surfaces are linear functions of x

Decision surfaces correspond to *y*(**x**) = constant, so that **w**T**x** + *w*0 = constant



Logistic Regression

- In logistic regression, we learn the conditional distribution P(y|x)
- Let $p_y(x;w)$ be our estimate of P(y|x), where w is a vector of adjustable parameters.
- Assume there are two classes, y = 0 and y = 1 and

$$p_1(\mathbf{x};\mathbf{w}) = \frac{1}{1+e^{-\mathbf{wx}}} \qquad p_0(\mathbf{x};\mathbf{w}) = 1 - \frac{1}{1+e^{-\mathbf{wx}}} \quad = \frac{\mathbf{1}}{\mathbf{1+e^{wx}}}$$

- This is equivalent to

$$\log\frac{p_1(\mathbf{x};\mathbf{w})}{p_0(\mathbf{x};\mathbf{w})} = \mathbf{wx}$$

- That is, the log odds of class 1 is a linear function of x
- Q: How to find **W**?

- Alternate representation of p(y|x) : $\mathbf{p_y(x;w) = \frac{1}{1+e^{-ywx}}; y = \{-1, 1\}}$

Logistic Regression

- Conditional data likelihood - Probability of observed Y values in the training data, conditioned on corresponding X values.
- We choose parameters w that satisfy

$$\mathbf{w} = \arg\max_{\mathbf{w}} \prod_{l} P(y^{l} \mid \mathbf{x}^{l}, \mathbf{w})$$

- where
  - **w** = <w_0,w_1 ,…,w_n> is the vector of parameters to be estimated,
  - y^l denotes the observed value of Y in the *l* th training example, and
  - $\mathbf{x}^l$ denotes the observed value of **X** in the *l* th training example

Logistic Regression

- Equivalently, we can work with log of conditional likelihood:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \sum_l \ln P(y^l \mid \mathbf{x}^l, \mathbf{w})$$

- Conditional data log likelihood, l(W), can be written as

$$l(\mathbf{w}) = \sum_l y^l \ln P(y^l = 1 \mid \mathbf{x}^l, \mathbf{w}) + (1 - y^l) \ln P(y^l = 0 \mid \mathbf{x}^l, \mathbf{w})$$

- Note here that Y can take only values 0 or 1, so only one of the two terms in the expression will be non-zero for any given y^l

Logistic Regression

- We need to estimate:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \sum_{l} \ln P(y^{l} \mid \mathbf{x}^{l}, \mathbf{w})$$

$$l(\mathbf{w}) = \sum_{l} y^{l} \ln P(y^{l} = 1 \mid \mathbf{x}^{l}, \mathbf{w}) + (1 - y^{l}) \ln P(y^{l} = 0 \mid \mathbf{x}^{l}, \mathbf{w})$$

- Equivalently, we can minimize negative log likelihood using gradient descent technique

- No closed-form solution though. Iterative method required.

- [HW] Find the derivative of l(w) !

Logistic Regression

- Overfitting can arise especially when data has very high dimensions and is sparse.
- One approach -> modified “penalized log likelihood function,” which penalizes large values of **w**, as before.

$$\mathbf{w} = \arg\max_{\mathbf{w}} \sum_{l} \ln P(y^{l} \mid \mathbf{x}^{l}, \mathbf{w}) - \frac{\lambda}{2} \| \mathbf{w} \|^{2}$$

- [HW] Find the Derivative !

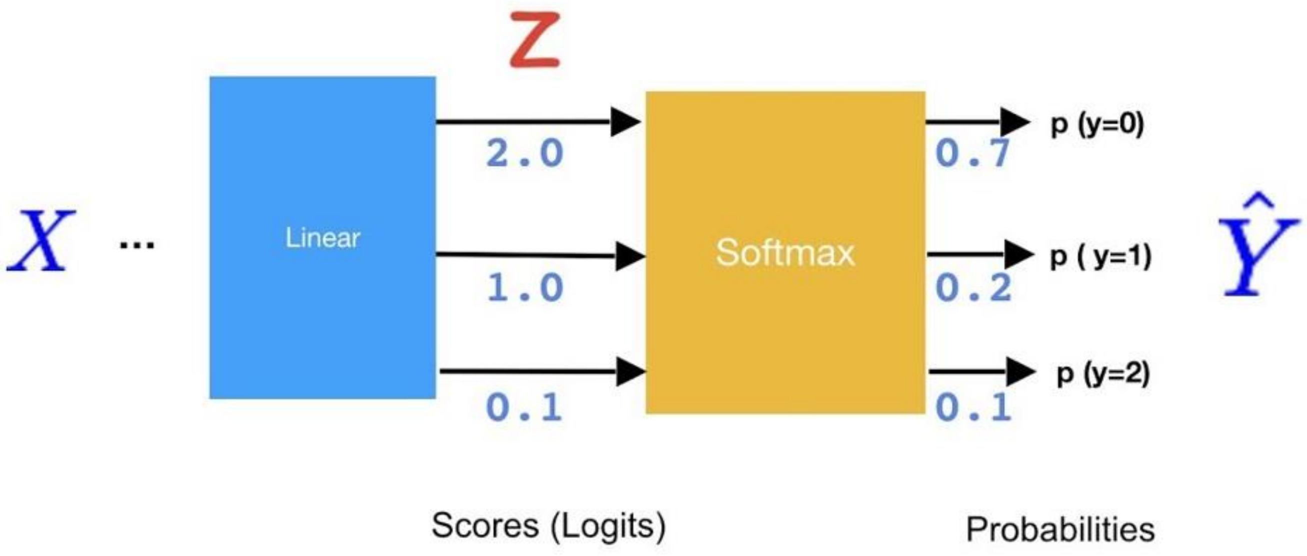
Logistic Regression

- LR: Functional form of P(Y|X), no assumption on P(X|Y)
- LR is a linear classifier
- LR optimized by conditional likelihood
- Extending logistic regression to multiple classes
  - Use softmax for each class k!

$$\mathbf{p(y = k|x) = \frac{exp(w_k^\top x)}{\sum_{i=1}^{K} exp(w_i^\top x)}}$$

Meet Softmax

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$



Probabilistic Generative Models

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1+\exp(-a)} = \sigma(a) \end{aligned}$$

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} \\ &= \frac{\exp(a_k)}{\sum_j \exp(a_j)} \end{aligned}$$

$$a_k = \ln p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k).$$

Probabilistic Generative Models

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)+p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1+\exp(-a)} = \sigma(a) \end{aligned}$$

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}.$$

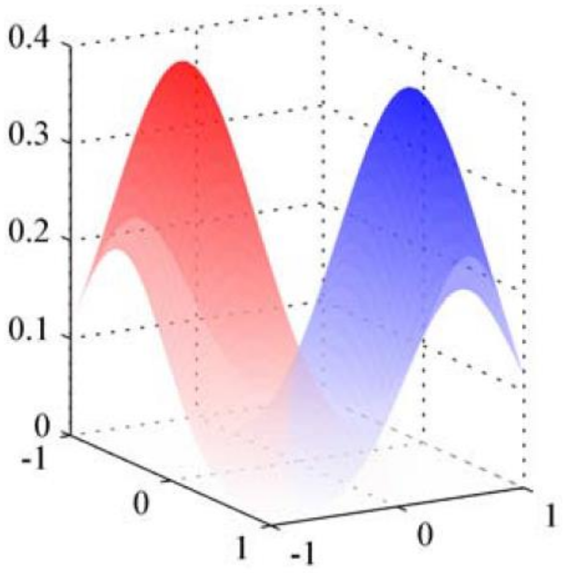
$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x}+w_0)$$

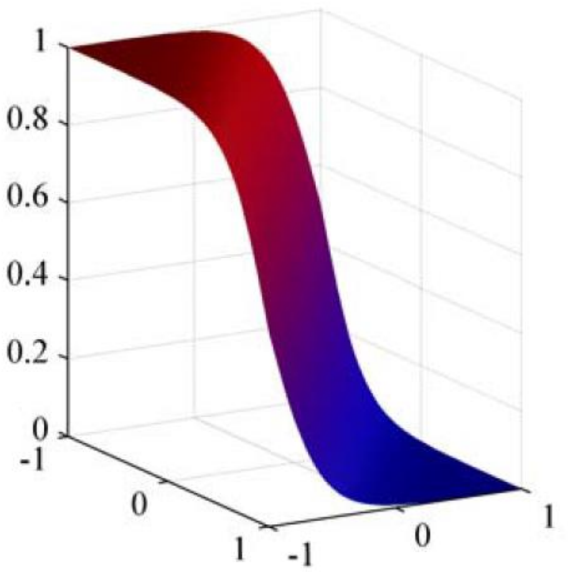
Probabilistic Generative Models

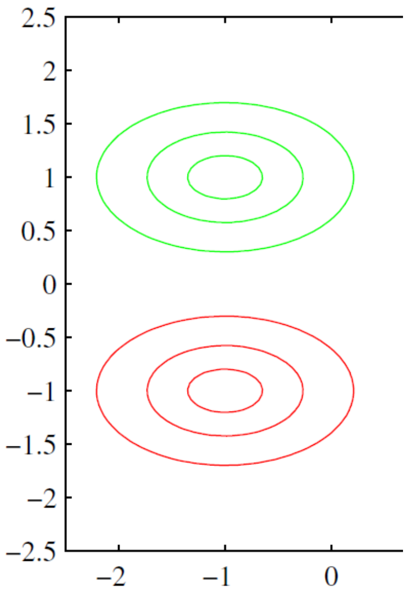
$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}.$$

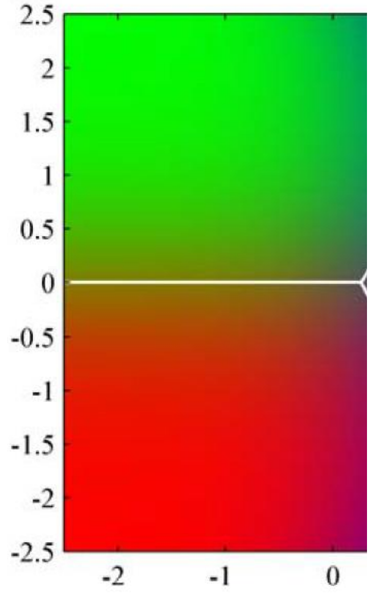
$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0)$$

$$\begin{aligned} \mathbf{w} &= \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}. \end{aligned}$$









Probabilistic Generative models

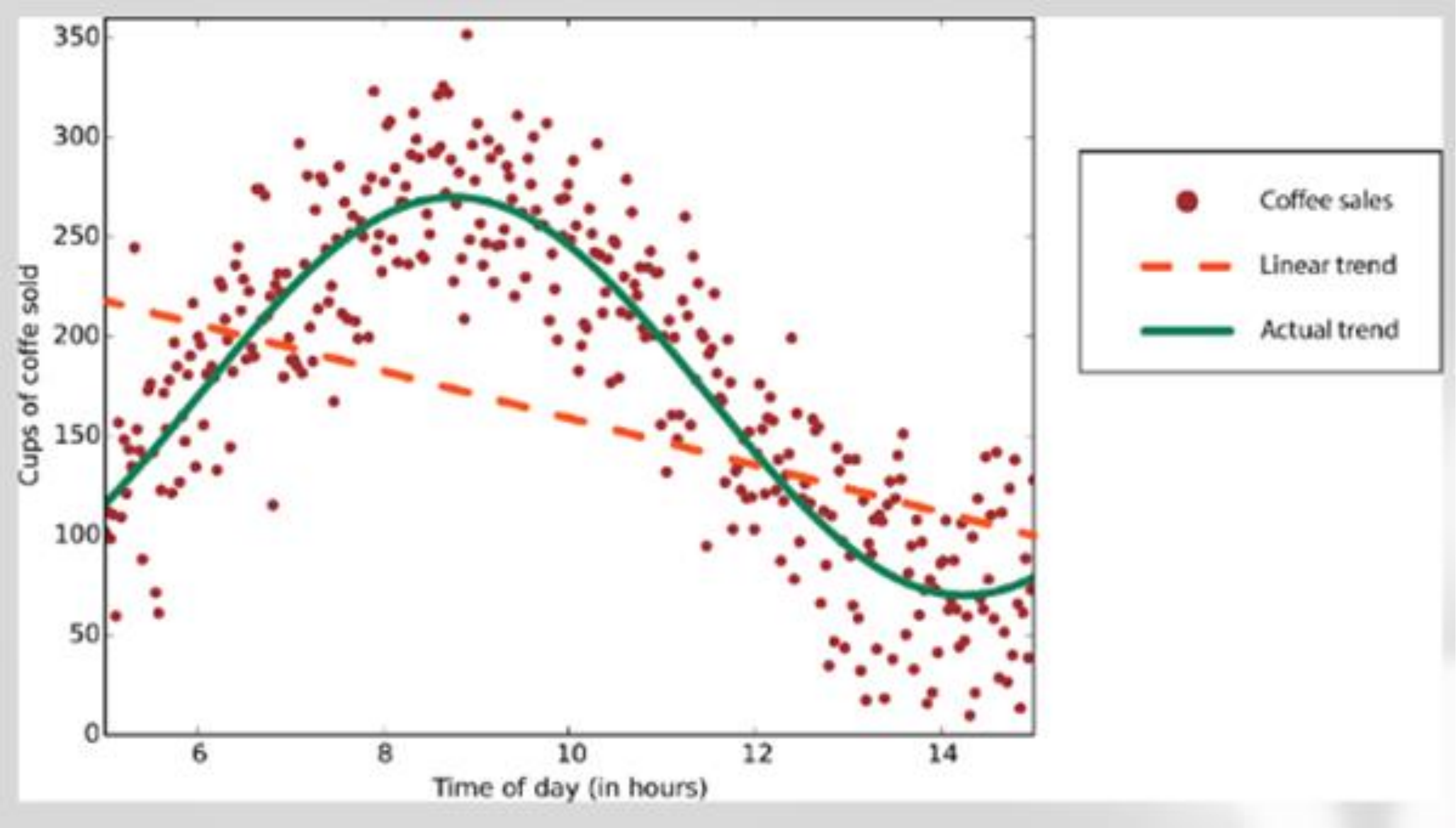
Classification : Evaluation metrics

Accuracy : $$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}[y_i == \hat{y}_i]$$

		Actual Label	
		Positive	Negative
Predicted Label	Positive	**True Positive (TP)**	**False Positive (FP)**
	Negative	**False Negative (FN)**	**True Negative (TN)**

Accuracy	(TP + TN) / (TP + TN + FP + FN)	The percentage of predictions that are correct
Precision	TP / (TP + FP)	The percentage of positive predictions that are correct
Sensitivity (Recall)	TP / (TP + FN)	The percentage of positive cases that were predicted as positive
Specificity	TN / (TN + FP)	The percentage of negative cases that were predicted as negative

Supervised learning : Regression



Number of vehicles passing a junction

p(y|x)?

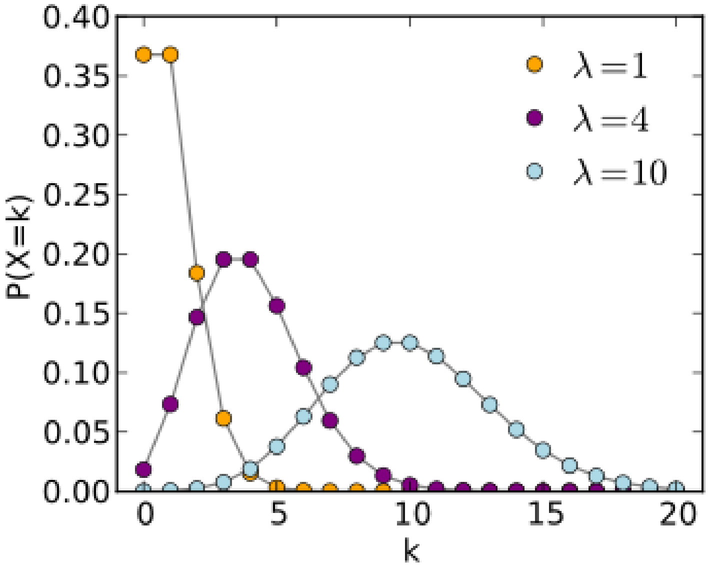
Y take values 0,1,2,3,….. but not 2.1, 3.4, 5.55……

https://medium.com/@heyozramos/regression-vs-classification-86d73c281c5e

Poisson Regression

- Poisson distribution : Model number of events occurring in a fixed interval of time/space

$$P(k \text{ events in interval}) = e^{-\lambda}\frac{\lambda^k}{k!}$$



- λ is the average (mean) number of events
- Y has a Poisson distribution, and assumes the logarithm of its expected value can be modeled by a linear combination of unknown parameters.

$$\lambda := \mathrm{E}(Y \mid x) = e^{\theta' x}$$

$$p(y \mid x; \theta) = \frac{\lambda^y}{y!}e^{-\lambda} = \frac{e^{y\theta' x}e^{-e^{\theta' x}}}{y!}$$

Poisson Regression : Learning parameters

- Likelihood $$p(y_1, \ldots, y_m \mid x_1, \ldots, x_m; \theta) = \prod_{i=1}^{m} \frac{e^{y_i \theta' x_i} e^{-e^{\theta' x_i}}}{y_i!}.$$

- Estimate parameters by maximum likelihood estimation

$$\ell(\theta \mid X, Y) = \log L(\theta \mid X, Y) = \sum_{i=1}^{m} \left(y_i \theta' x_i - e^{\theta' x_i} - \log(y_i!) \right).$$

- Use gradient descent to find the optimal value of θ.

Thank you !

Reference

[1] Christopher Bishop, Pattern Recognition and Machine Learning
[2] Kevin Murphy, Machine Learning : A Probabilistic Perspective