Regression

Goal: Fit a model to predict a continous target value:

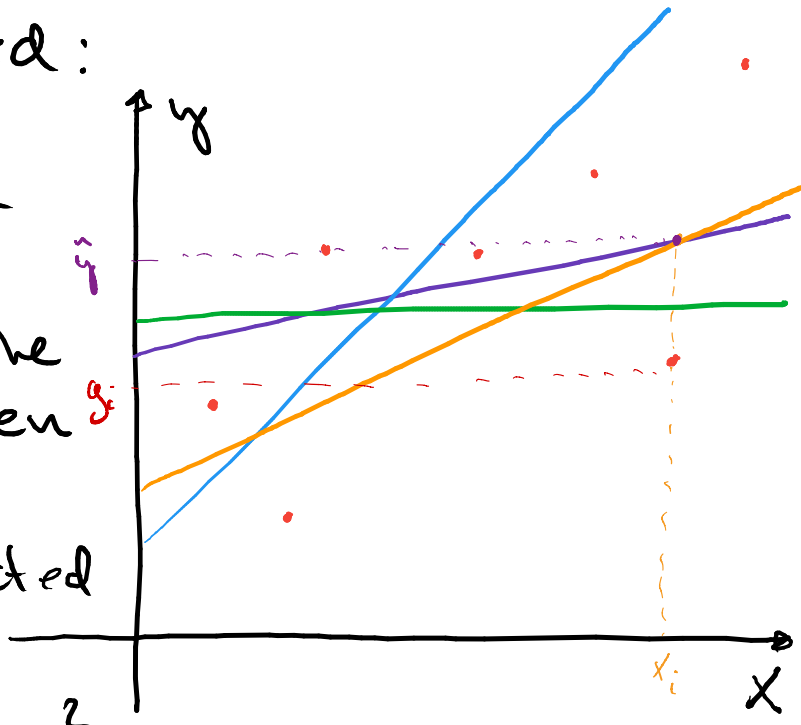
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots \beta_p x_p$$

Simple linear regression:

$$y = \beta_0 + \beta_1 x_1$$

Determine β_0, β_1 s.t. the "best" line is fitted:

By "best" we mean the one that reduces the error between the observed and the predicted value:

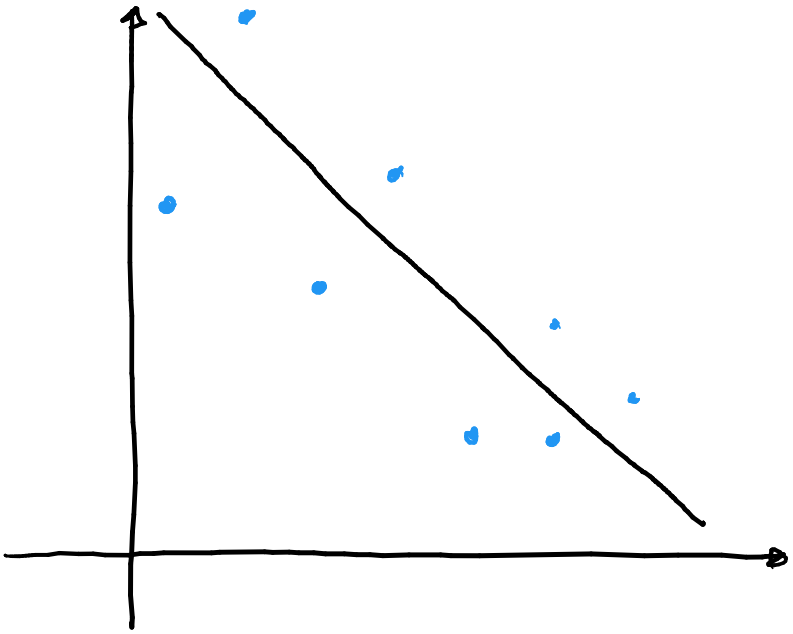


$$SSE = \sum (y_i - \hat{y}_i)^2$$

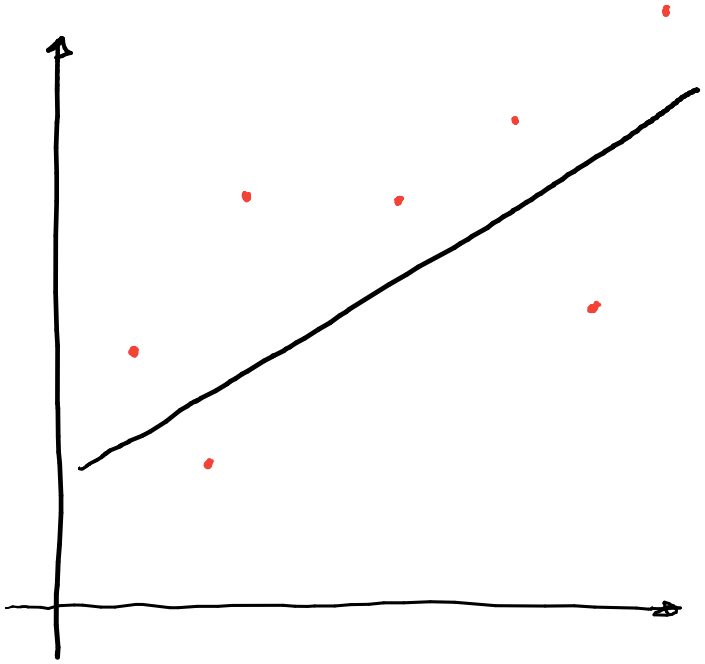
observed — predicted

The association between x and y is called the **correlation** and is denoted r: $-1 \leq r \leq 1$

$r < 0$:
Negative correlation



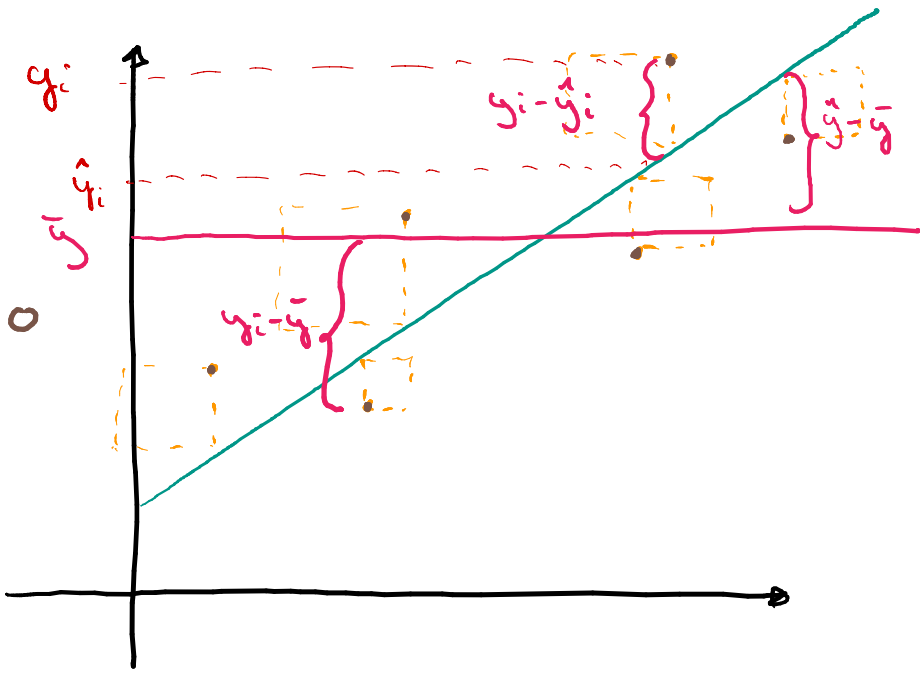
$r > 0$:
Positive correlation



When $r = 0$: No correlation

$$SSE = \sum (y_i - \hat{y}_i)^2$$

The goal of least squares regression is to minimize the sum of these squares, s.t. $\hat{y} = \beta_0 + \beta_1 x_1$ has the smallest possible error



Errors:

SST =

SSE =

SSR =

Deviations:

Total Deviation:

Explained :

Unexplained :

- $r^2 = \frac{SSR}{SST} = \frac{Explained}{Total}$ } r^2 tells us the amount of variance my model is able to explain

$0 \leq r^2 \leq 1$

r^2 tells us something about our model's predictive ability.

r tells us something about how well X and y co-vary.

How to determine Parameters $\hat{\beta}_i$:

1) Least squares
   - Statistics
   - linear algebra
2) Gradient Descent (Today)
   - often used to explain regression in M.L. since it is an easy entry to gradient descent
3) Singular Value Decomposition (Next Week)
   - Efficient. the one that is implemented

Simple linear Regression

↳ Supervised Learning Problem

Training Data: $\{(x_i, y_i)\}_{i=1}^{n}$

Model: $\hat{y} = f(x) = \beta_0 + \beta_1 x$

We define a loss function

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2$$

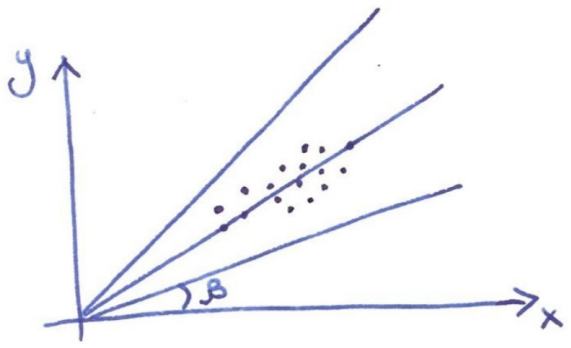
This is called MSE = Mean Squared Error

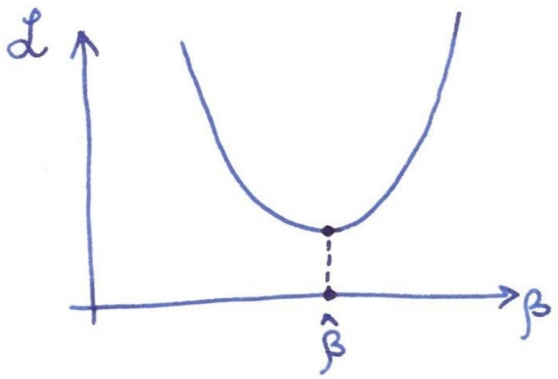
MSE =

Baby linear Regression:

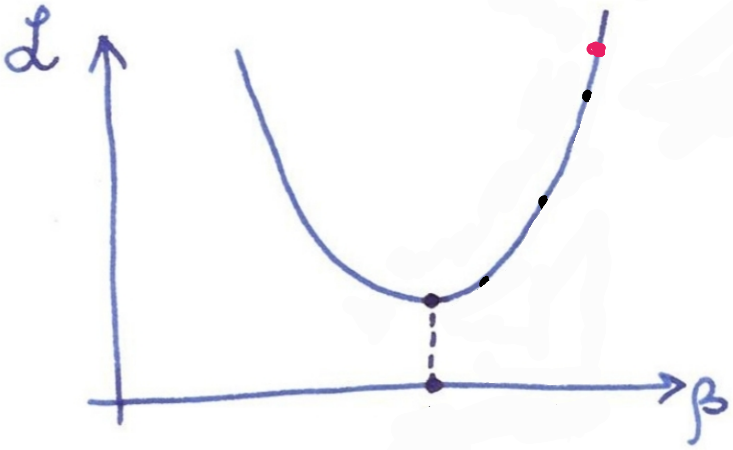
$$f(x) = \beta x$$

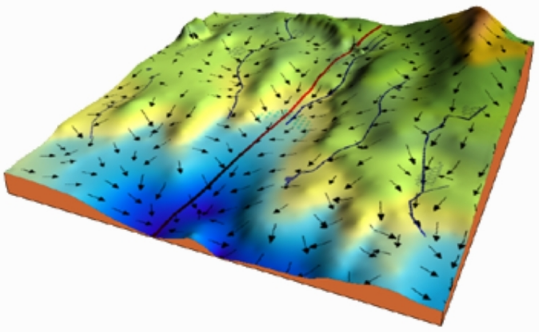
$$L(\beta) = \frac{1}{n}\sum (y_i - \hat{\beta} x_i)^2$$





Gradient Descent

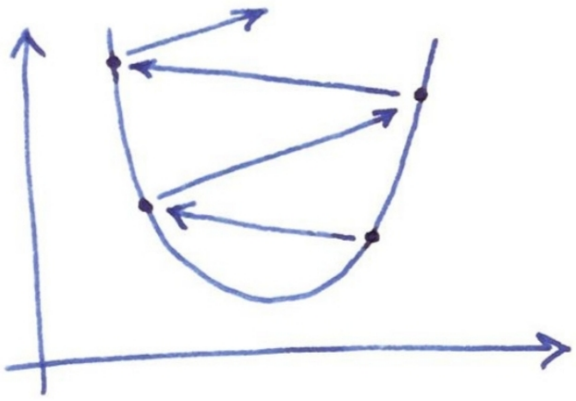




update Rule:

$$\beta \leftarrow \beta - \eta \cdot \frac{d\mathcal{L}(\beta)}{d\beta}$$

Here η^* (eta) is the learning rate



Too large η → Divergence

Too small η → slow convergence

Finding the derivative: Gradient Descent

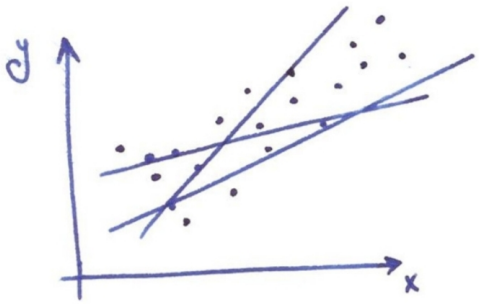
$$L(\beta) = \frac{1}{n} \sum (y_i - \beta x_i)^2$$

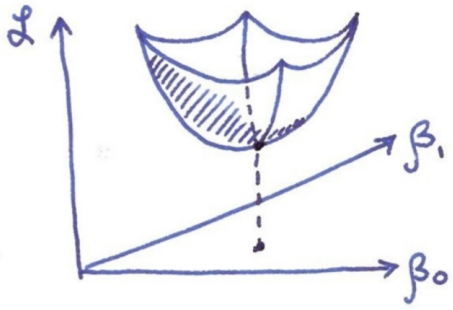
$$L'(\beta) =$$

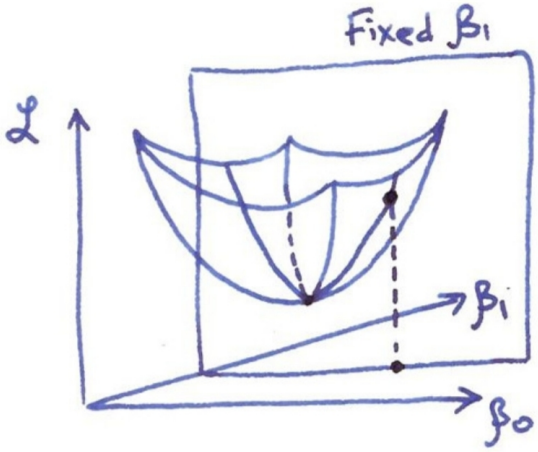
$$=$$

Linear Regression: Gradient Descent:

$$\mathcal{L}(\beta_0, \beta_1) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$







$$\beta_0 \leftarrow \beta_0 - \eta \frac{\partial \mathcal{L}}{\partial \beta_0}$$

$$\beta_1 \leftarrow \beta_1 - \eta \frac{\partial \mathcal{L}}{\partial \beta_1}$$

In vector form:

$$\bar{\beta} \leftarrow$$

Computing the gradient:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2$$

We need partial derivatives:

$$\frac{\partial \mathcal{L}}{\partial \beta_0} =$$

$$\frac{\partial \mathcal{L}}{\partial \beta_1} =$$

Why MSE?

Assume SSE:

n	L
100	200
500	800
1000	1200

ok if we want to compare models!

As n increases, SSE gets larger!

MSE

n	L	L/n
100	200	2
500	800	1.6
1000	1200	1.2

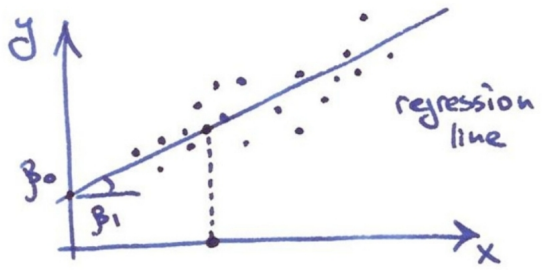
As n increases, MSE decreases

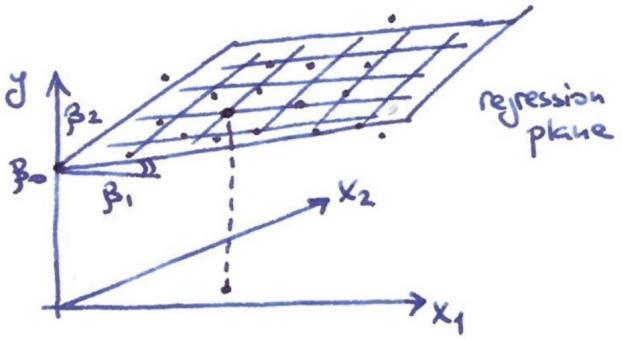
↳ error decreases as algorithm is gaining more experience.

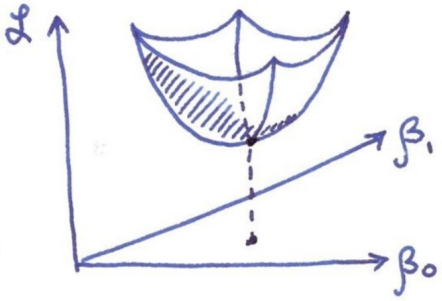
RMSE = Root mean squared error:

$$L = \sqrt{\frac{1}{n}\Sigma(y_i - \hat{y}_i)^2}$$

Multiple linear Regression has >1 predictors







The model:

$$f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p$$

It is convenient to define $x_0 \equiv 1$:

$$f(x) = \bar{\beta} \cdot \bar{x} =$$

The loss and Gradient:

$$\mathcal{L}(\beta) =$$

Partial Derivatives:

$$\frac{\partial \mathcal{L}}{\partial \beta_k} = -\frac{2}{n} \sum_{i=1}^{n} \left(y^{(i)} - \beta^\top \mathbf{x}^{(i)}\right) x_k^{(i)}$$

Gradient:

$$\nabla \mathcal{L} = -\frac{2}{n} \sum_{i=1}^{n} \left(y^{(i)} - \beta^\top \mathbf{x}^{(i)}\right) \mathbf{x}^{(i)}$$

Design Matrix:

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}^{(1)} \\ \mathbf{x}^{(2)} \\ \vdots \\ \mathbf{x}^{(n)} \end{pmatrix} = \begin{pmatrix} x_0^{(1)} & x_1^{(1)} & \cdots & x_p^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \cdots & x_p^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_0^{(n)} & x_1^{(n)} & \cdots & x_p^{(n)} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_0 & \mathbf{x}_1 & \cdots & \mathbf{x}_p \end{pmatrix}$$

Response/observation vector:

$$\mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{pmatrix}$$

Given X and β:

$$\hat{y} = X\beta$$

We can state the loss function as:

$$L = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}_i)^2 = \frac{1}{n}\|y - X\hat{\beta}\|^2$$

$$= \frac{1}{n}(y - X\hat{\beta})^T(y - X\hat{\beta})$$

Gradient:

$$\nabla L = -\frac{2}{n}X^T(y - X\hat{\beta}) = 0$$

Matrix Algebra:

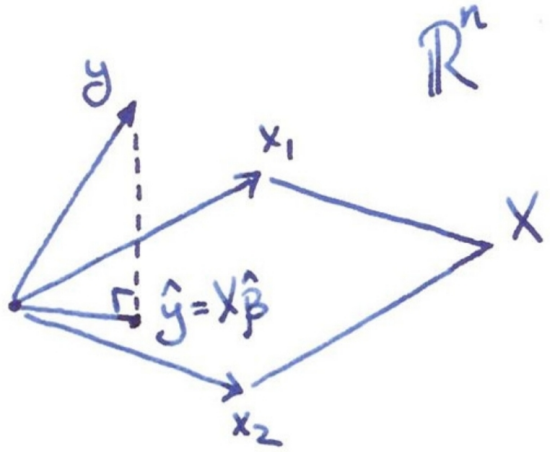
① $A\bar{x} = \bar{b}$

$\bar{x} = A^{-1}\bar{b}$

② $[A \; b] \overset{\text{ref}}{\sim} [I \; \bar{x}]$

$\hat{y}$ = y-hat:

$\hat{y} = X\hat{\beta} =$



The orthogonal projection of y onto the feature-space spanned by the features $(X_1, X_2 \cdots X_p)$ in the n-dimensional space.

$\|y - X\hat{\beta}\|^2$ minimal $\Rightarrow$ $\perp$

$X^T(y - X\hat{\beta}) = 0 \Rightarrow \perp$

Linear Regression boils down to orthogonal projections!

Performance of regression:

1) MSE
2) r^2
3) RMSE

Base Case: - In classification: random guess

- In regression: Guess the mean $\bar{y}$

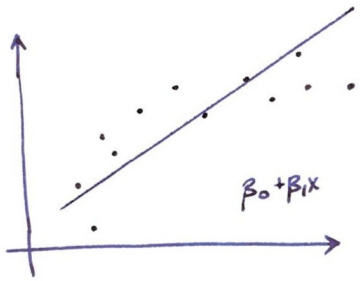
Polynomial regression:

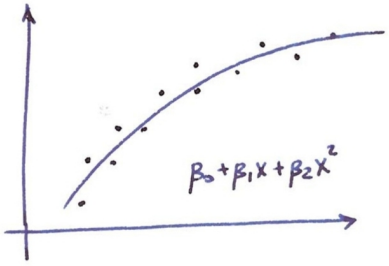
Assume we model y and x and include x^2, x^3 etc → Still linear model:

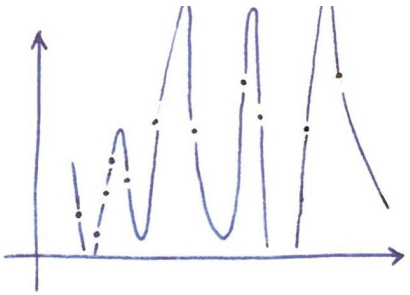
$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \dots$$

$$X = \begin{bmatrix} 1 & x_1^1 & x_1^3 & \cdots \\ 1 & x_2^1 & x_2^3 & \cdots \\ 1 & x_3^2 & x_3^3 & \cdots \\ \vdots & \vdots & \vdots & \\ 1 & \vdots & \vdots & \cdots \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

Underfitting/overfitting:







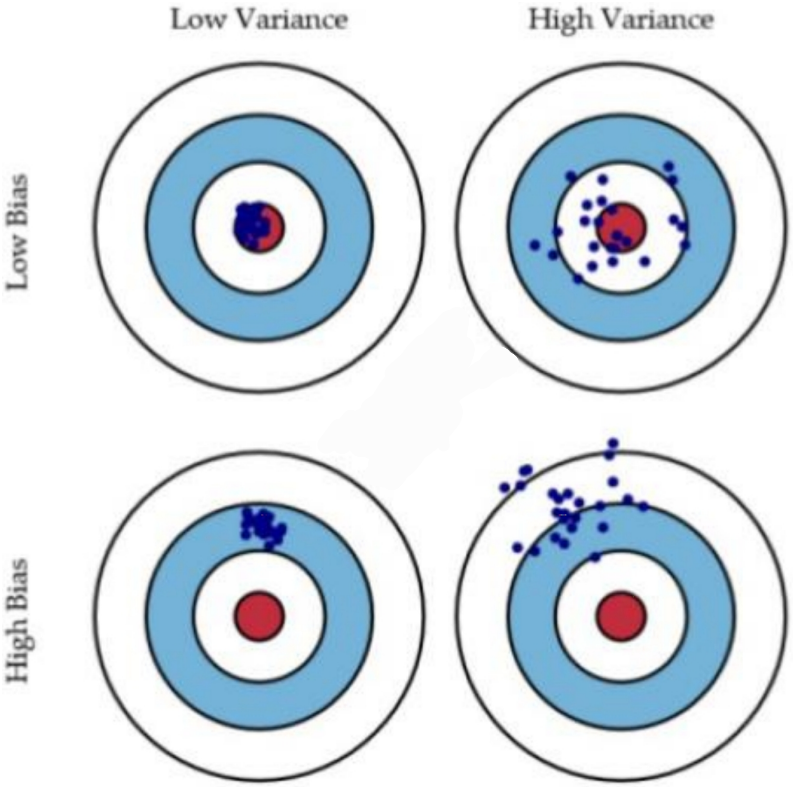
Underfitting
High Bias
Low Variance

Bias: Inability to learn from data

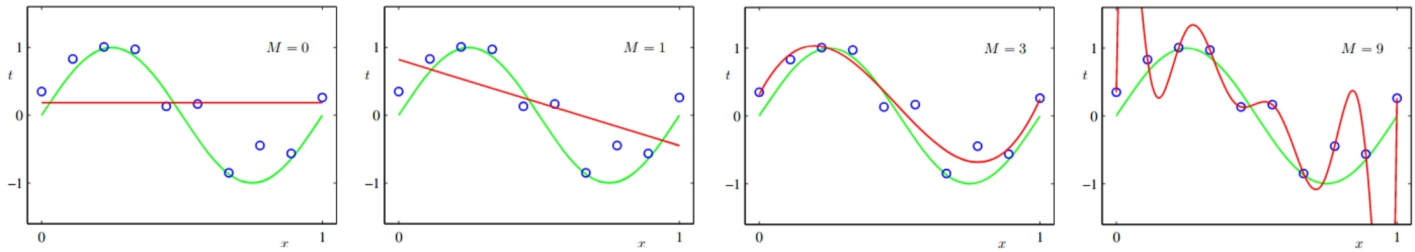
Overfitting
High Variance
Low Bias

Variance: rely too much on data
→ difference between data sets

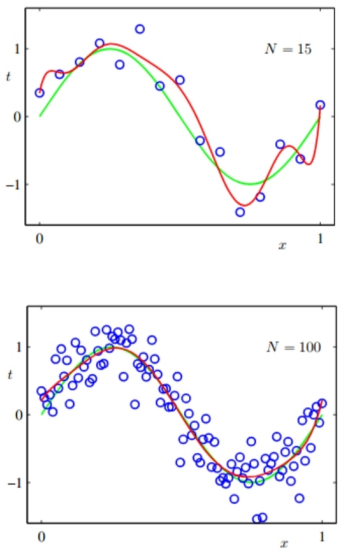
Bias-Variance Trade off



Overfitting and high variance



	$M=0$	$M=1$	$M=3$	$M=9$
$w_0^\star$	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



Bishop, *Pattern Recognition and Machine Learning*

How to avoid: Regularization

Idea: Penalize large coefficient

$$\mathcal{L} = \frac{1}{n} \|y - X\beta\|^2$$

Here λ is <u>penalty</u> term and λ is called the <u>regularisation parameter</u>

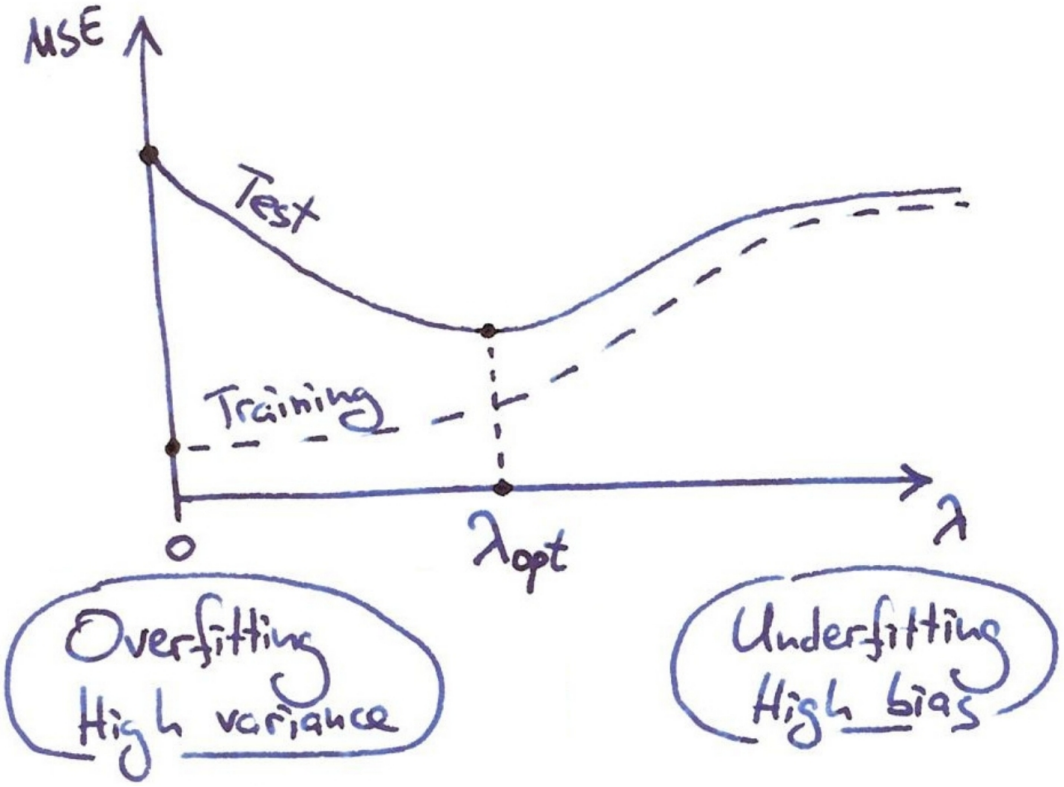
Common choices:

$$R(\beta) = \|\beta\|_2^2 = \sum \beta_i^2 \qquad \text{ridge } L_2$$

$$R(\beta) = \|\beta\|_1 = \sum |\beta_i| \qquad \text{lasso } L_1$$

$$R(\beta) = \lambda_1 \|\beta\| + \lambda_2 \|\beta\|_2^2 = \lambda \cdot \sum (\alpha \cdot \|\beta\|_1 + (1-\alpha)\|\beta\|^2) \qquad \text{Elastic Net}$$

<u>Loss function:</u> $\mathcal{L} = \frac{1}{n} \|y - X\beta\|^2 + \lambda R(\beta)$



Ridge Regression:

Loss function:

$$\mathcal{L} = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|^2.$$

Gradient:

$$\nabla\mathcal{L} = -\frac{2}{n}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + 2\lambda\boldsymbol{\beta}.$$

Gradient descent:

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta\nabla\mathcal{L} = \boldsymbol{\beta} + \eta\frac{2}{n}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) - 2\eta\lambda\boldsymbol{\beta} =$$

$$= \underbrace{(1 - 2\eta\lambda)}_{\text{"weight decay"}}\boldsymbol{\beta} + \eta\frac{2}{n}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

coefficients decrease

Analytic Solution:

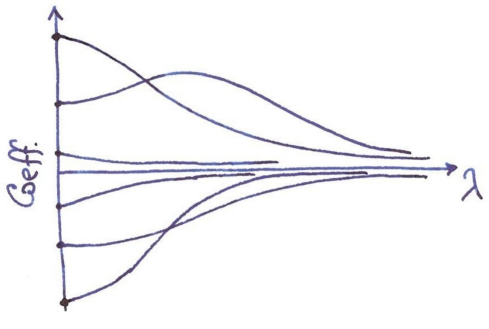
$$-\frac{2}{n}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) + 2\lambda\hat{\boldsymbol{\beta}} = 0$$

$$\mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}} + n\lambda\hat{\boldsymbol{\beta}} = \mathbf{X}^\top\mathbf{y}$$

$$(\mathbf{X}^\top\mathbf{X} + n\lambda\mathbf{I})\hat{\boldsymbol{\beta}} = \mathbf{X}^\top\mathbf{y}$$

$$\boxed{\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X} + n\lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}}$$

This is an example of a shrinkage estimator.



Lasso:

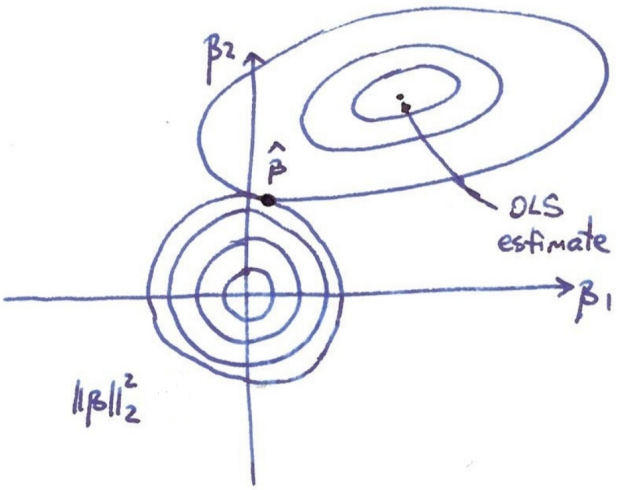
$$L = \frac{1}{n} \|y - X\beta\|^2$$

No analytic solution

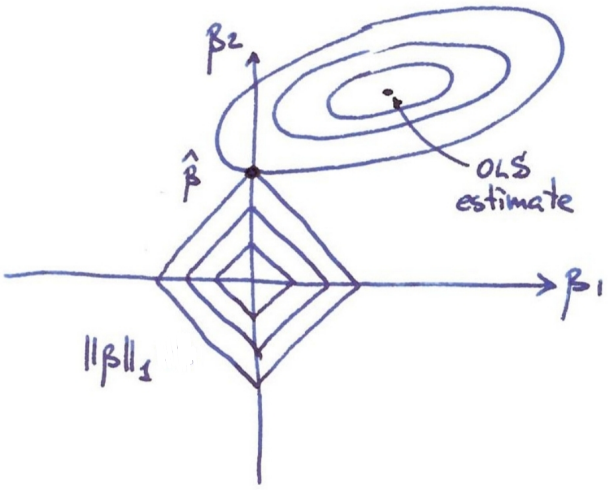
→ Drive coefficient to zero (sparse)

↳ feature selection!

Ridge vs. Lasso



Ridge



Lasso

Elastic Net:

"Combines" ridge and lasso:

α: "How much" lasso vs. ridge

$$R(\beta) = \lambda_1 \|\beta\| + \lambda_2 \|\beta\|_2^2 = \lambda \cdot \sum (\alpha \cdot \|\beta\|_1 + (1-\alpha)\|\beta\|^2)$$