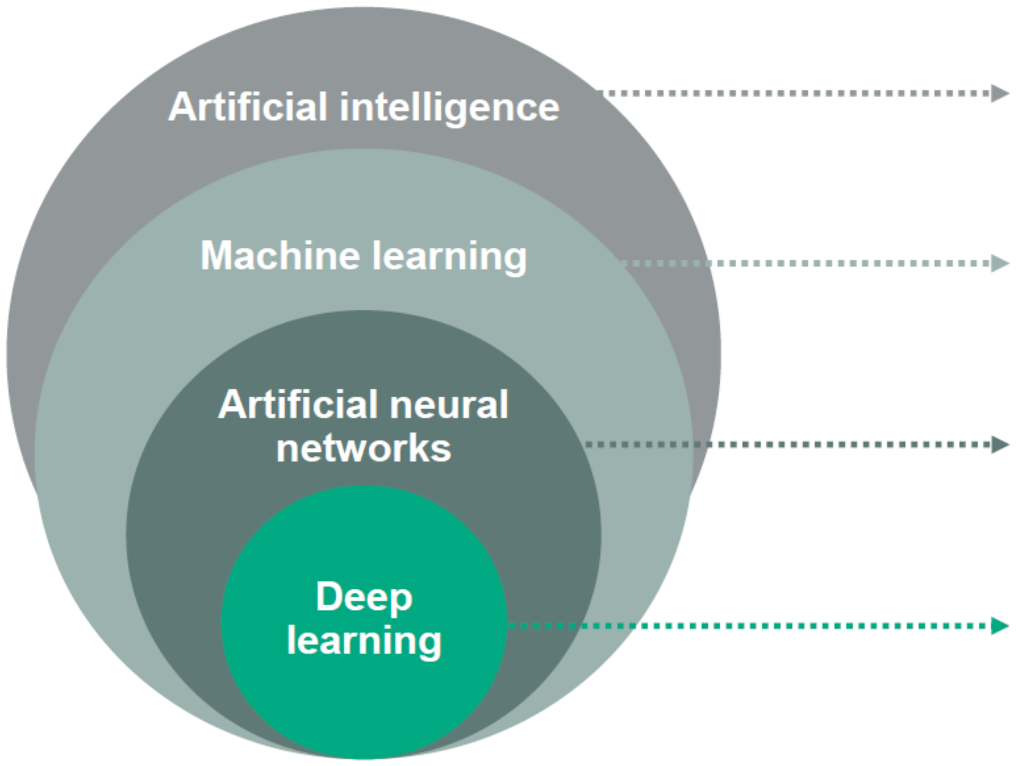
CSE473: Computational Intelligence

Regression

by:
Hossam Abd El Munim
Computer & Systems Engineering Dept.,
Ain Shams University,
1 El-Sarayat Street, Abbassia, Cairo 11517

Artificial Intelligence & Neural Network

AI intra-relationships



Artificial intelligence (AI)
Any techniques that enable machines to solve a task in a way like humans do

Machine learning (ML)
Algorithms that allow computers to learn from examples without being explicitly programmed

Artificial neural networks (ANN)
Brain-inspired machine learning models

Deep learning (DL)
A subset of ML which uses deep artificial neural networks as models and automatically builds a hierarchy of data representations

Hewlett Packard Enterprise

Machine Learning Problems

	Supervised Learning	***Unsupervised Learning***
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Dataset split

Training Images

- Train classifier

Validation Images

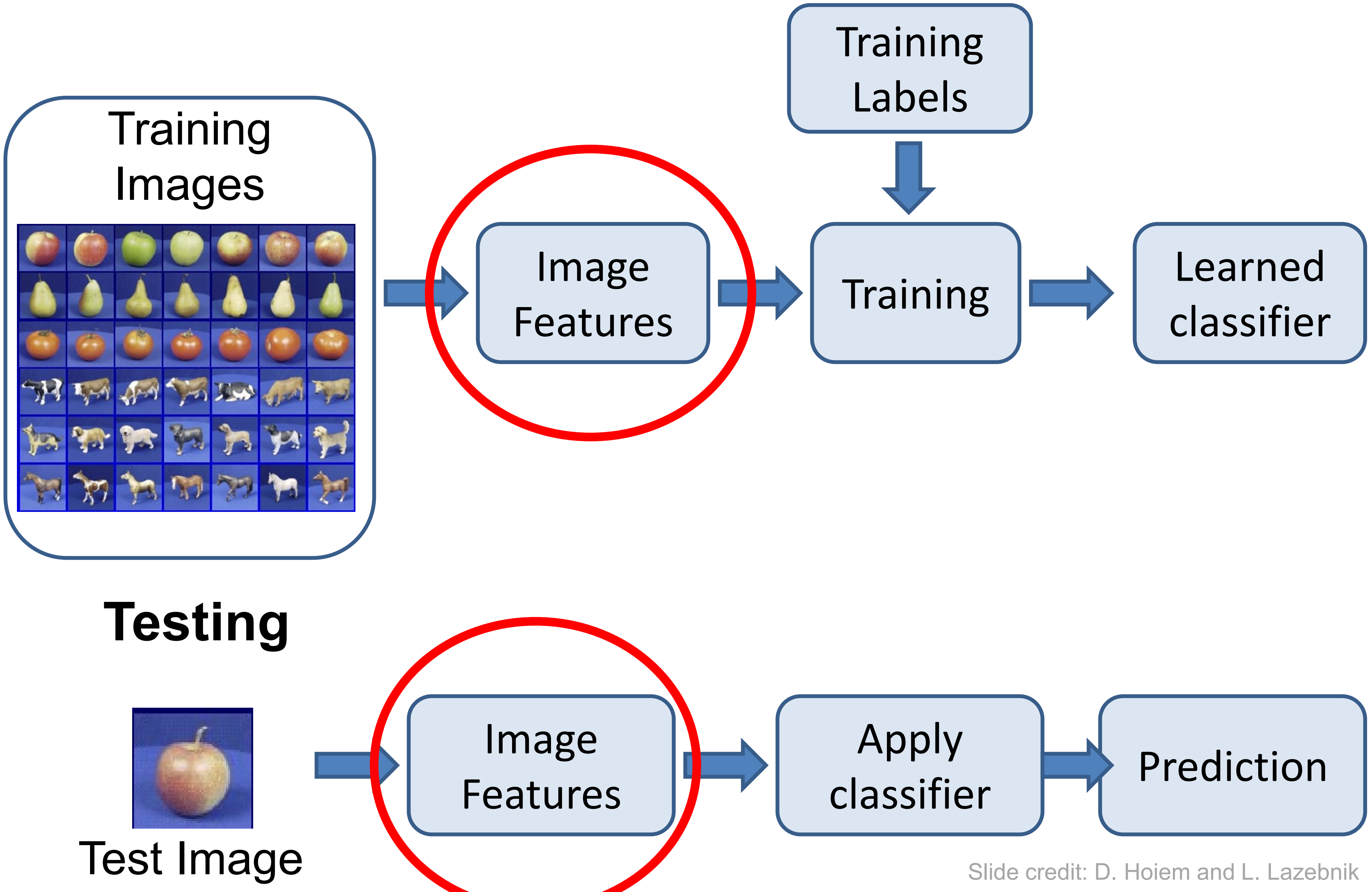
- Measure error
- Tune model hyperparameters

Testing Images

- Secret labels
- Measure error

Random train/validate splits = cross validation

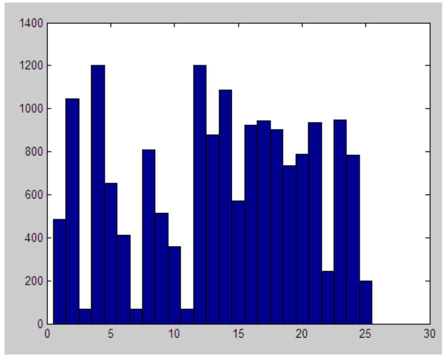
Training

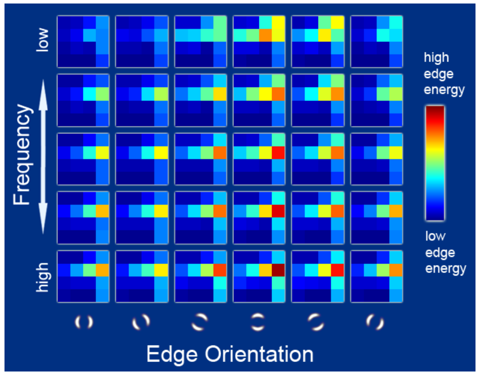


Slide credit: D. Hoiem and L. Lazebnik

Features

- Raw pixels
- Histograms
- Templates
- SIFT descriptors
  - GIST
  - ORB
  - HOG….





L. Lazebnik

Machine Learning Problems

	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

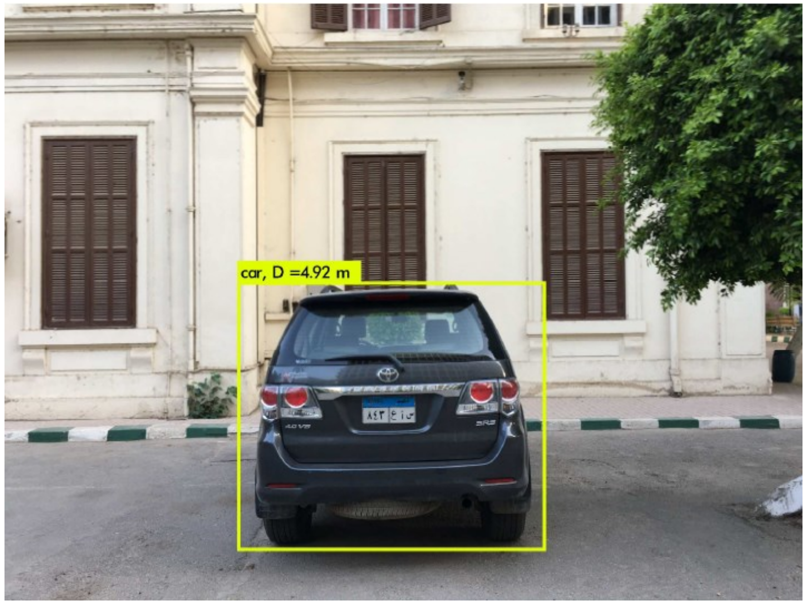
What is Regression?

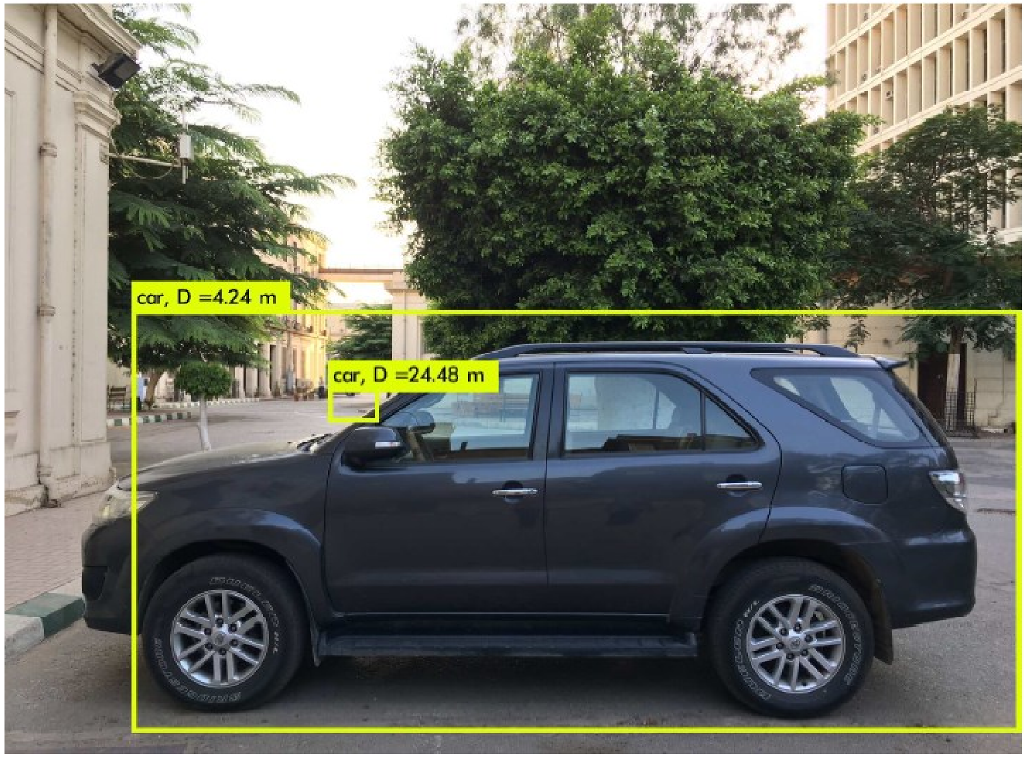
- Regression is to fit a model to a noisy data
- The goal is to predict continuous values, such as the head or the human pose from a given image.
- Regression problems are defined in opposition to classification problems, where the goal is to estimate categorical values (corresponding to labels/classes).

Example frames of the *Biwi head-pose dataset.*

Gabriele Fanelli, Matthias Dantone, Juergen Gall, Andrea Fossati, and Luc Gool. Random Forests for Real Time 3D Face Analysis. *IJCV*, 2013.

What is Regression?

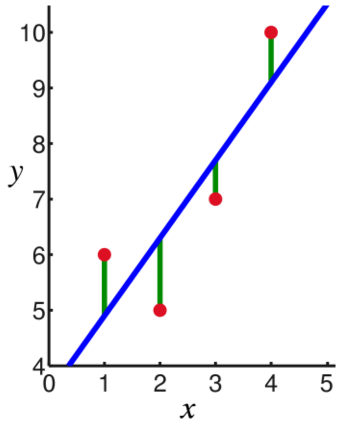


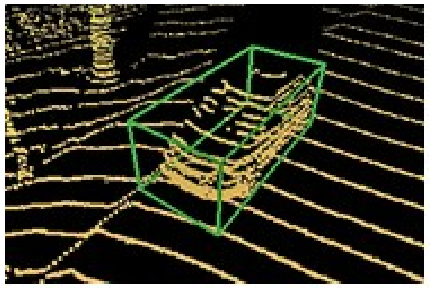


Least Squares Regression

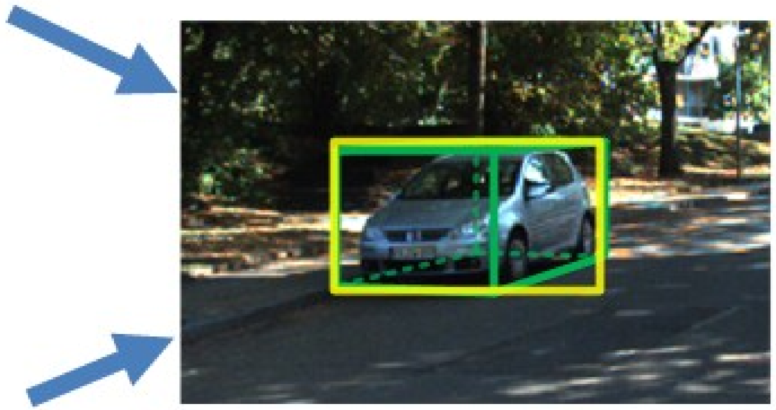
Given a data set, D = {($\mathbf{X}_1$, y_1), ($\mathbf{X}_2$, y_2)... ($\mathbf{X}_N$, y_N)} of labelled feature vectors where $y \in R$, we to estimate the vector **W** that minimizes the following objective function which measure the difference between the actual and desired outputs:-

$$J(\mathbf{W}) = [(y_1 - \mathbf{W}.\ \mathbf{X}_1)^2 + (y_2 - \mathbf{W}.\ \mathbf{X}_2)^2 + \ldots + (y_N - \mathbf{W}.\ \mathbf{X}_N)^2]/(2N)$$




3D detection

2D detection


Projected 3D detection (green) and 2D detection (yellow)

Gradient Descent Solution

Given a data set, D = {($\mathbf{X}_1$, y_1), ($\mathbf{X}_2$, y_2)... ($\mathbf{X}_N$, y_N)} of labelled feature vectors where $y \in R$, we to estimate the vector **W** that minimizes the following objective function which measure the difference between the actual and desired outputs:-

$$\partial J/\partial \mathbf{W}$$

$$=$$

$$-[(y_1-\mathbf{W}.\ \mathbf{X}_1)\mathbf{X}_1 + (y_2-\mathbf{W}.\ \mathbf{X}_2)\mathbf{X}_2 + \ldots + (y_N-\mathbf{W}.\ \mathbf{X}_N)\mathbf{X}_N]/N$$

$$\mathbf{W}^{t+1} = \mathbf{W}^{t} - \eta[\partial J/\partial \mathbf{W}]^{t}$$

η is the learning rate.

A Closed Form Solution in One Step (1)

$\mathbf{X}_i = [x_d^i, x_{d-1}^i, \ldots, x_1^i, x_0^i]^T$

$\mathbf{W} = [w_d, w_{d-1}, \ldots, w_1, w_0]^T$

$y_1 = \mathbf{W}^T\mathbf{X}_1 = (\mathbf{X}_1)^T\mathbf{W}$

$y_2 = \mathbf{W}^T\mathbf{X}_2 = (\mathbf{X}_2)^T\mathbf{W}$

.

.

.

$y_N = \mathbf{W}^T\mathbf{X}_N = (\mathbf{X}_N)^T\mathbf{W}$

A Closed Form Solution in One Step (2)

$$\mathbf{DW} = \mathbf{Y}$$

$$\mathbf{D} = \begin{pmatrix} \mathbf{X}_1^{\mathrm{T}} \\ \mathbf{X}_2^{\mathrm{T}} \\ \dots \\ \mathbf{X}_N^{\mathrm{T}} \end{pmatrix} \qquad \mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_{\mathrm{N}} \end{pmatrix}$$

A Closed Form Solution in One Step (3) Pseudo Inverse

$$[\mathbf{D}]_{(N)X(d+1)}[\mathbf{W}]_{(d+1)X1} = [\mathbf{Y}]_{(N)X1}$$

$$\mathbf{D}^{T}\mathbf{D}\mathbf{W} = \mathbf{D}^{T}\mathbf{Y}$$

$$\mathbf{W} = (\mathbf{D}^{T}\mathbf{D})^{-1}\mathbf{D}^{T}\mathbf{Y}$$

The Need for Regularization Illustration with Polynomial Curve Fitting

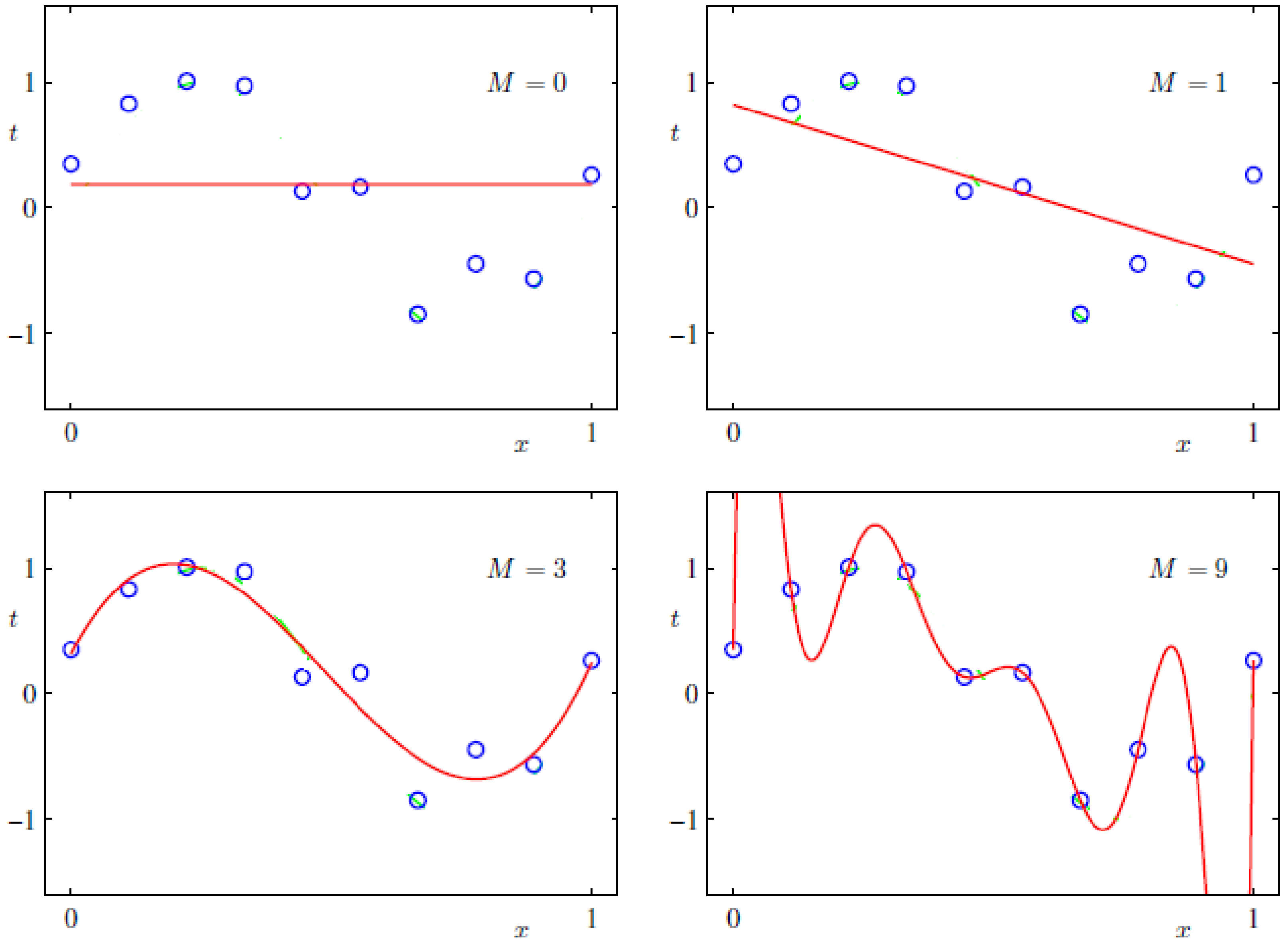
1D Example: Polynomial Curve Fitting

Given a data set, D = {(x_1, y_1), (x_2, y_2)... (x_N, y_N)} of points positions, we to estimate the vector **W** that minimizes the following objective function which measure the difference between the actual and desired outputs:-

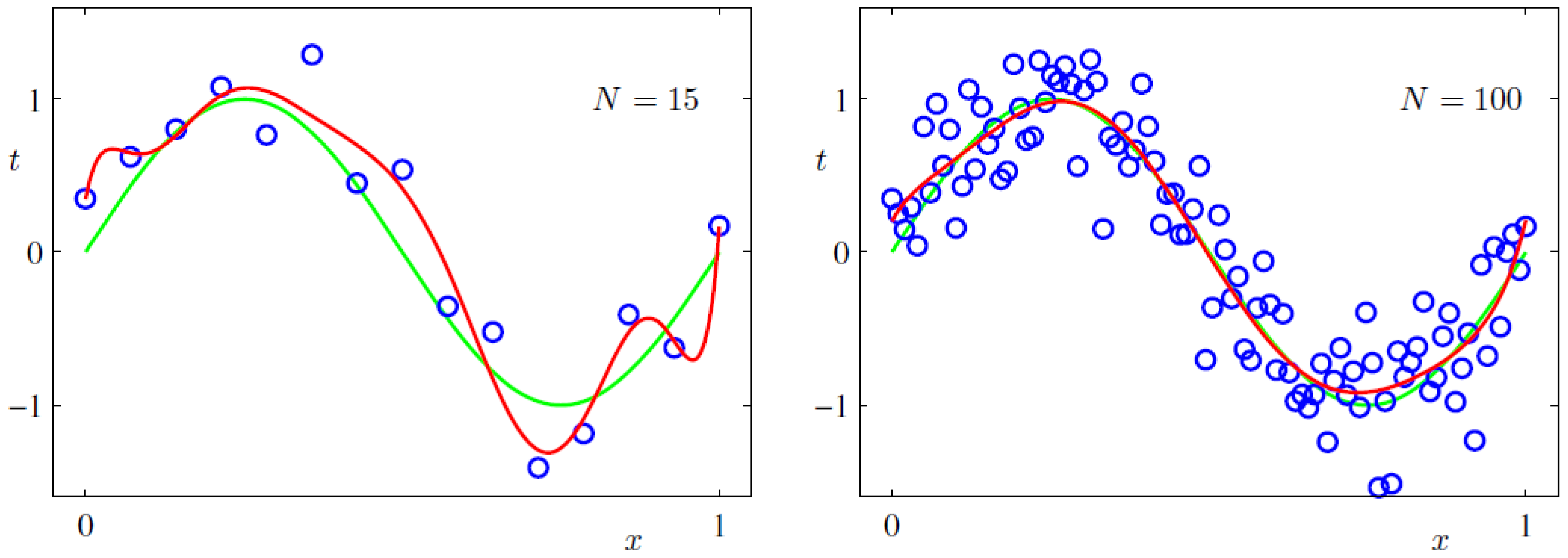
$$J(\mathbf{W}) = [(y_1-y(x_1, \mathbf{W}))^2 + (y_2-y(x_2, \mathbf{W}))^2 + \ldots + (y_N-y(x_N, \mathbf{W}))^2]/(2N)$$

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

Polynomial Order (M) = Model Complexity & Overfitting



Training Dataset Size and Overfitting



Plots of the solutions obtained by minimizing the sum-of-squares error function using the $M = 9$ polynomial for $N = 15$ data points (left plot) and $N = 100$ data points (right plot). We see that increasing the size of the data set reduces the over-fitting problem.

Parameters Magnitudes

Table of the coefficients $\mathbf{w}^\star$ for polynomials of various order. Observe how the typical magnitude of the coefficients increases dramatically as the order of the polynomial increases.

	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^\star$	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

Definition of overfitting

- When the model describes the noise, rather than the signal.
- when a model is capturing idiosyncrasies of the data rather than generalities.
  - Often caused by too many parameters relative to the amount of training data.
  - E.g. an order-*N* polynomial can intersect any *N+1* data points
- How can you tell the difference between overfitting, and a bad model?

Possible detection of overfitting

- Stability
  - An appropriately fit model is stable under different samples of the training data
  - An overfit model generates inconsistent performance

- Performance
  - A good model has low test error
  - A bad model has high test error

Regularization

One technique that is often used to control the over-fitting phenomenon in such cases is that of *regularization*, which involves adding a penalty term to the error function (1.2) in order to discourage the coefficients from reaching large values. The simplest such penalty term takes the form of a sum of squares of all of the coefficients, leading to a modified error function of the form

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

where $\|\mathbf{w}\|^2 \equiv \mathbf{w}^{\mathrm{T}}\mathbf{w} = w_0^2 + w_1^2 + \ldots + w_d^2$, and the coefficient λ governs the relative importance of the regularization term compared with the sum-of-squares error term.

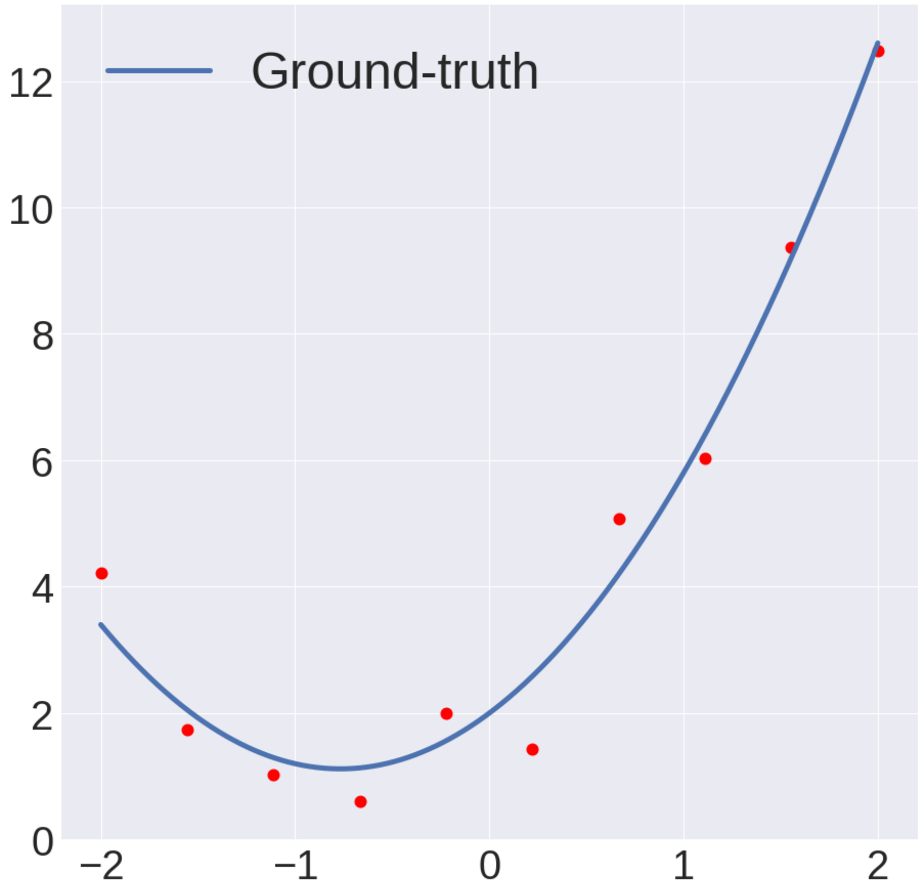
We can Show that:

$$\mathbf{W} = (\mathbf{D}^T\mathbf{D} + \lambda\mathbf{I})^{-1}\mathbf{D}^T\mathbf{Y}$$

AGAIN

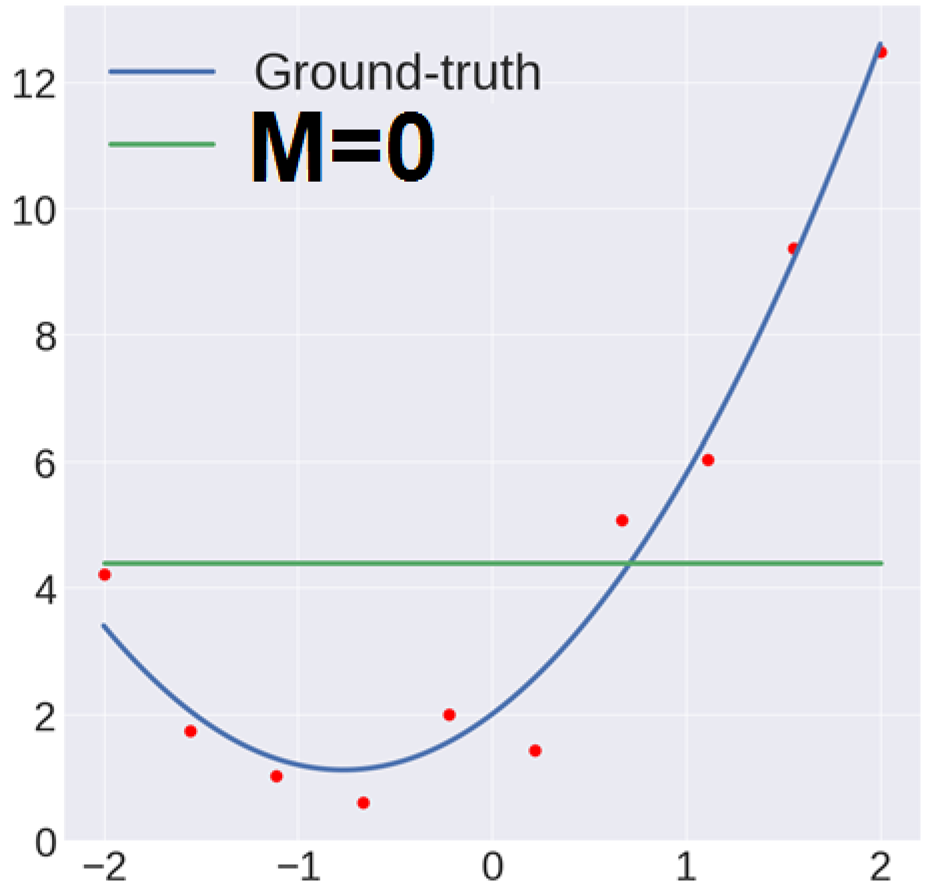
(Over/Under)fitting and Complexity

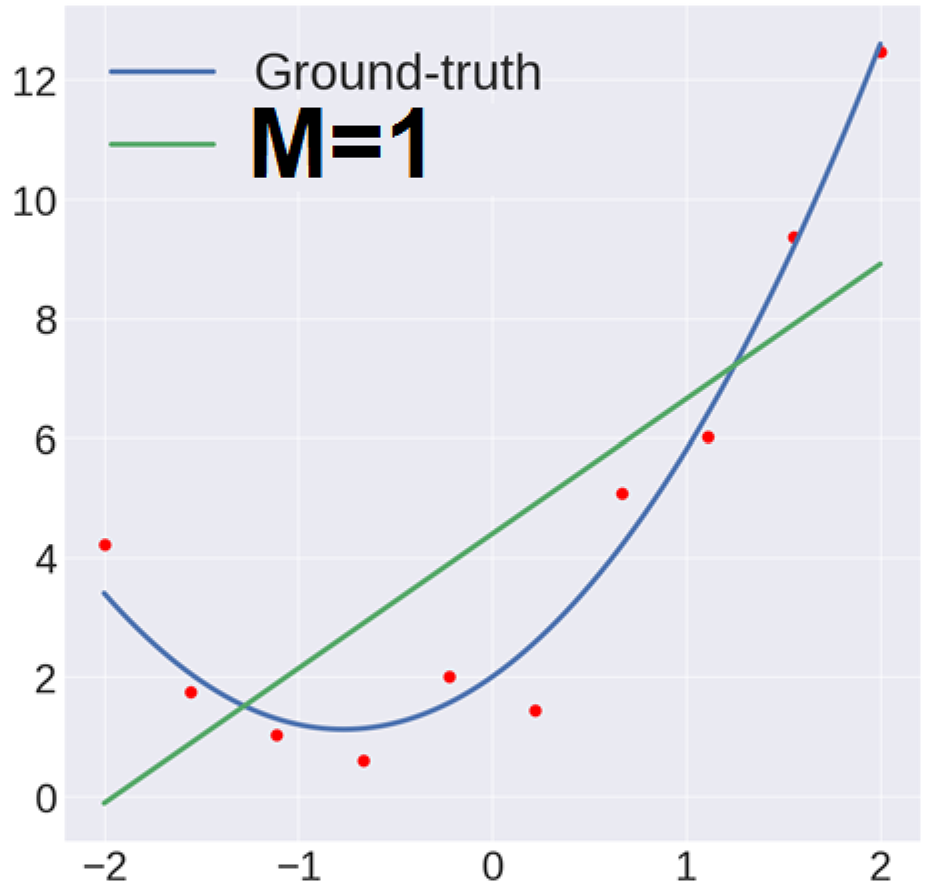
Model: $1.5x^2 + 2.3x + 2 + n(0,0.5)$



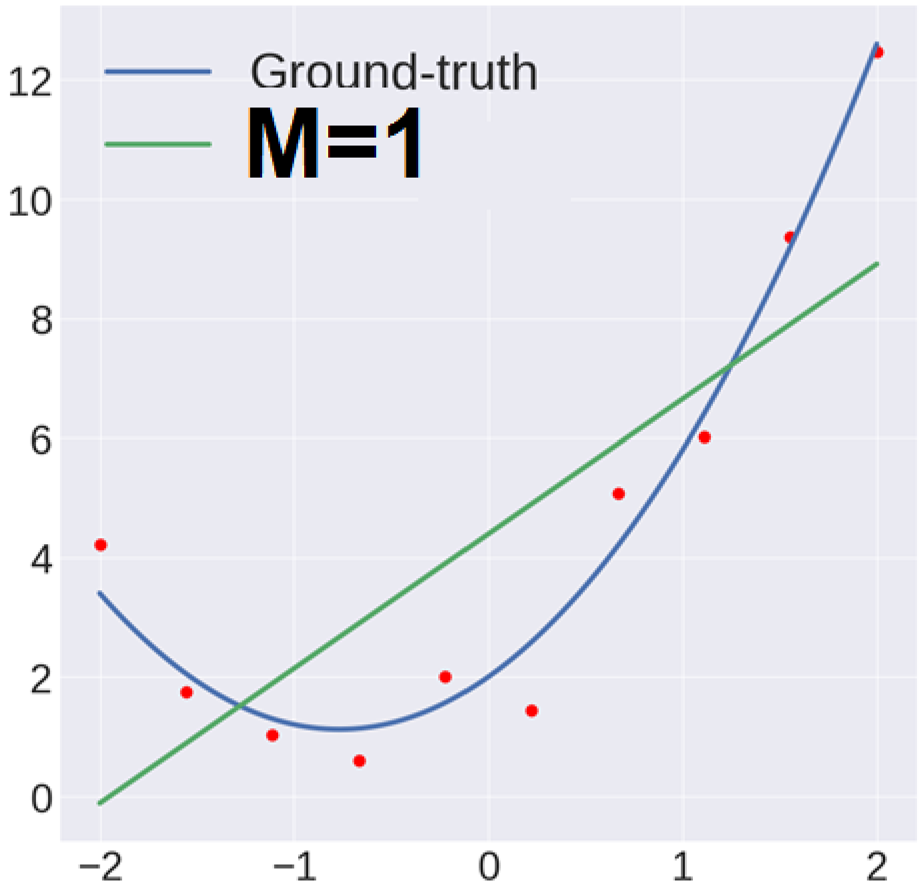
Underfitting

Model: $1.5x^2 + 2.3x+2 + n(0,0.5)$





Underfitting

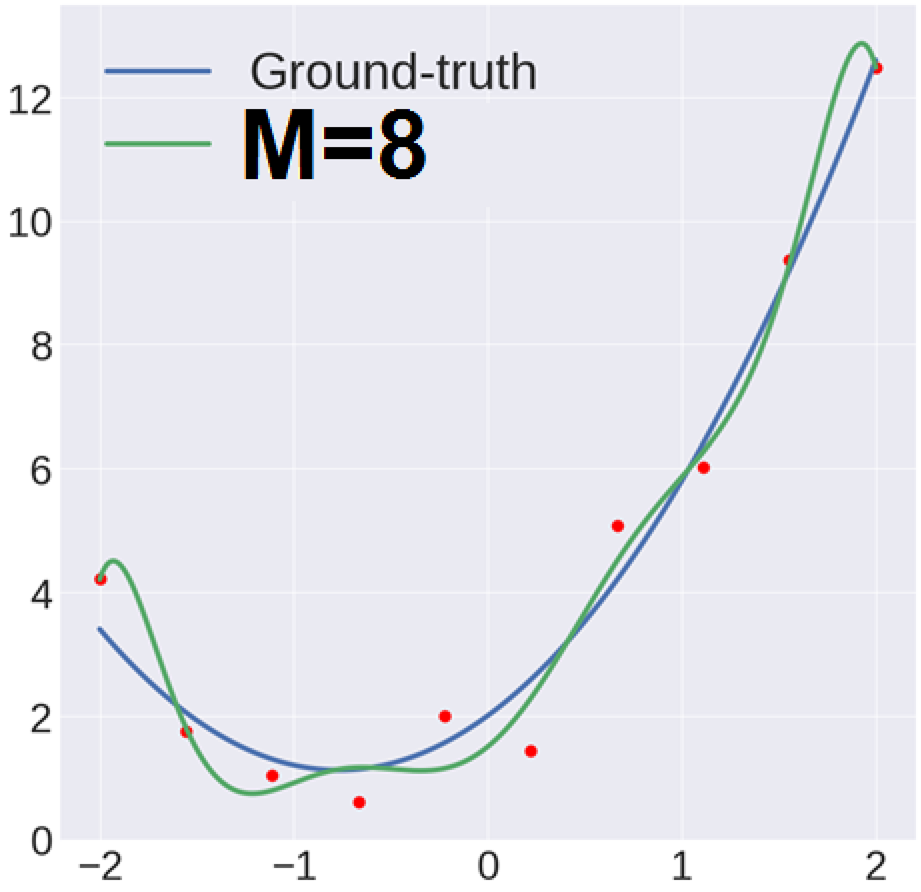


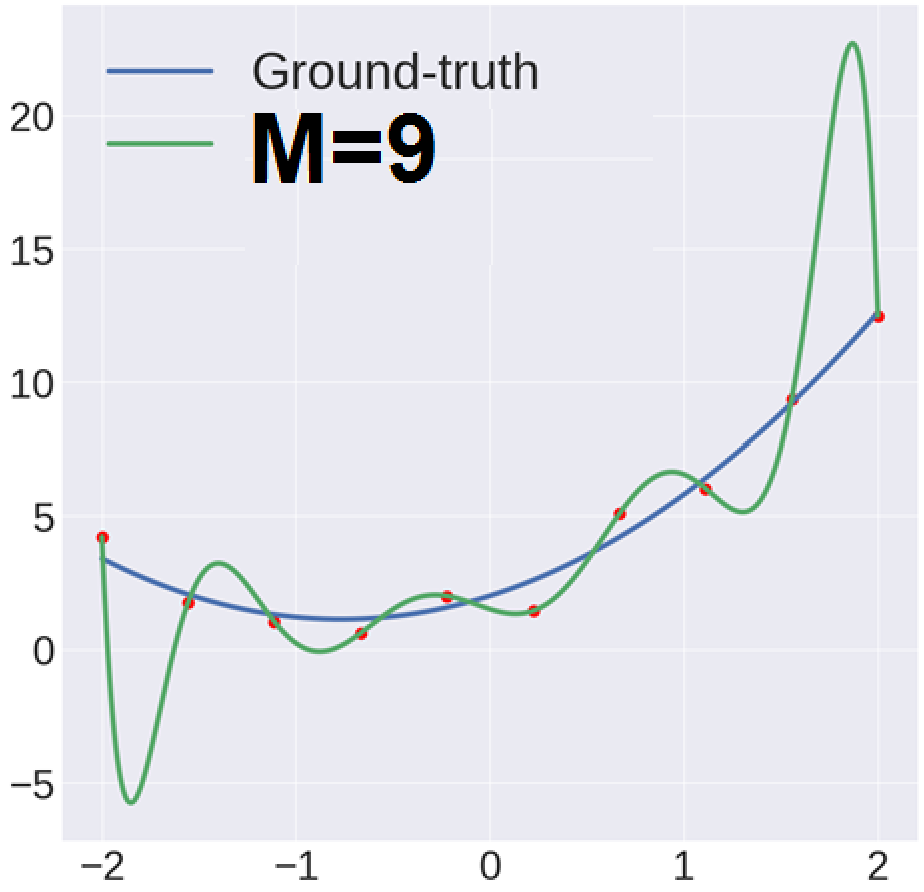
Model doesn’t have the parameters to fit the data.

Bias (statistics): Error intrinsic to the model.

Overfitting

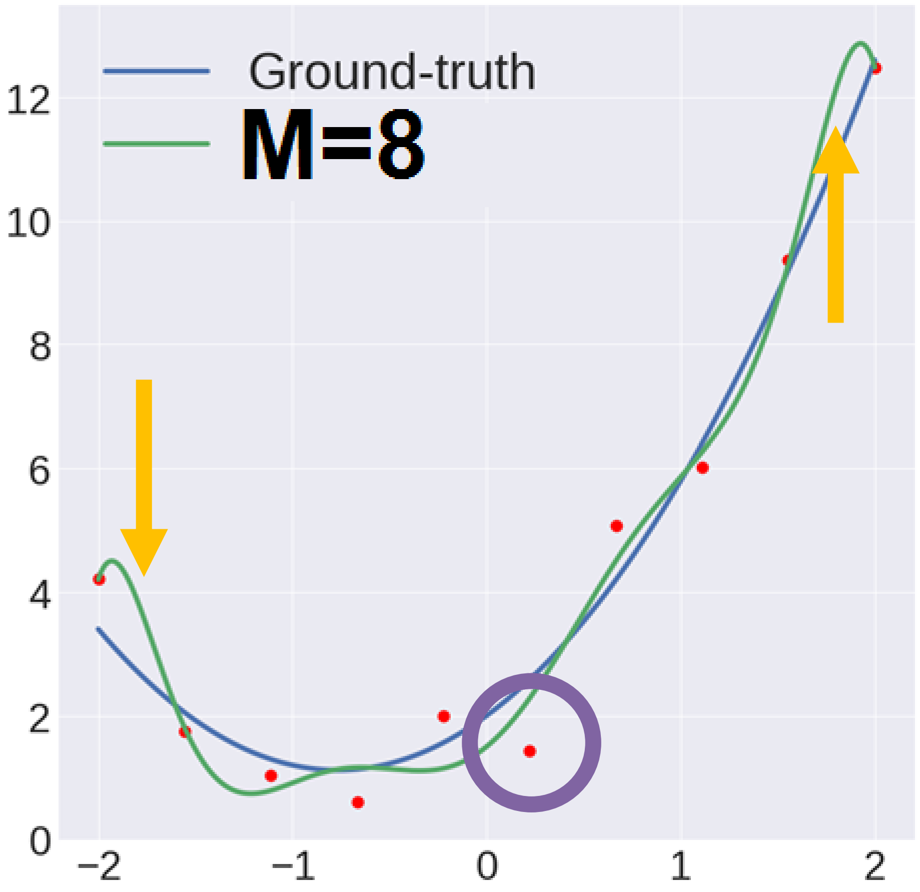
Model: $1.5x^2 + 2.3x+2 + n(0,0.5)$

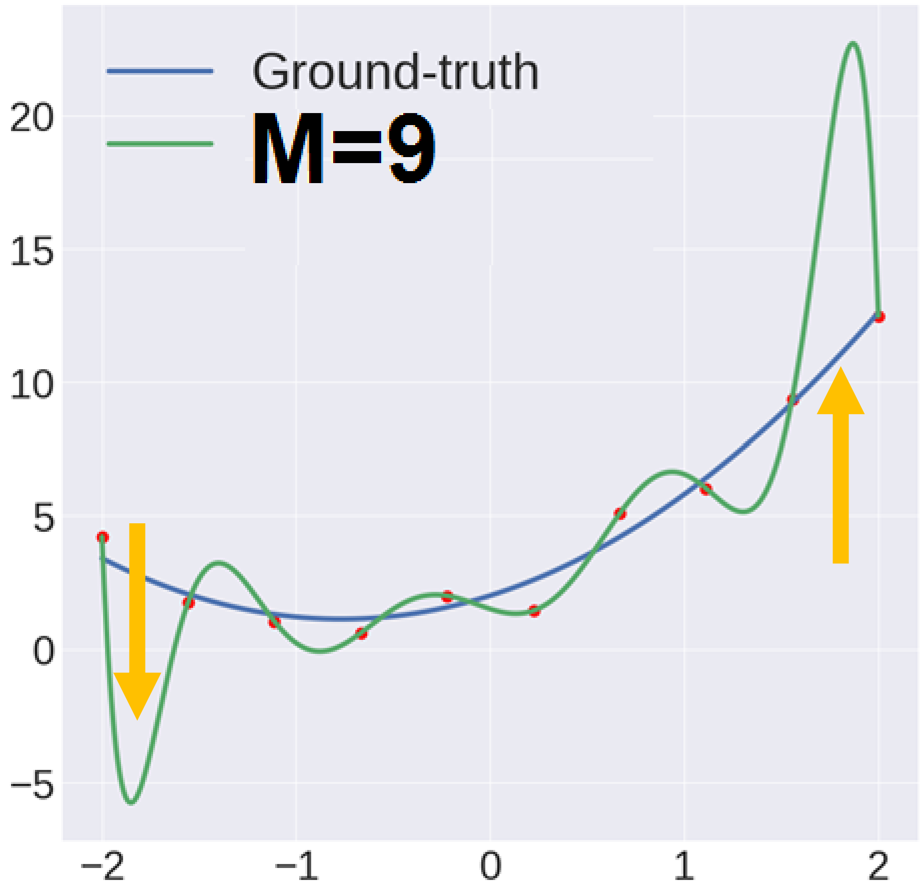




Overfitting

Model has high ***variance***: remove **one point**, and model changes dramatically





Fitting Model

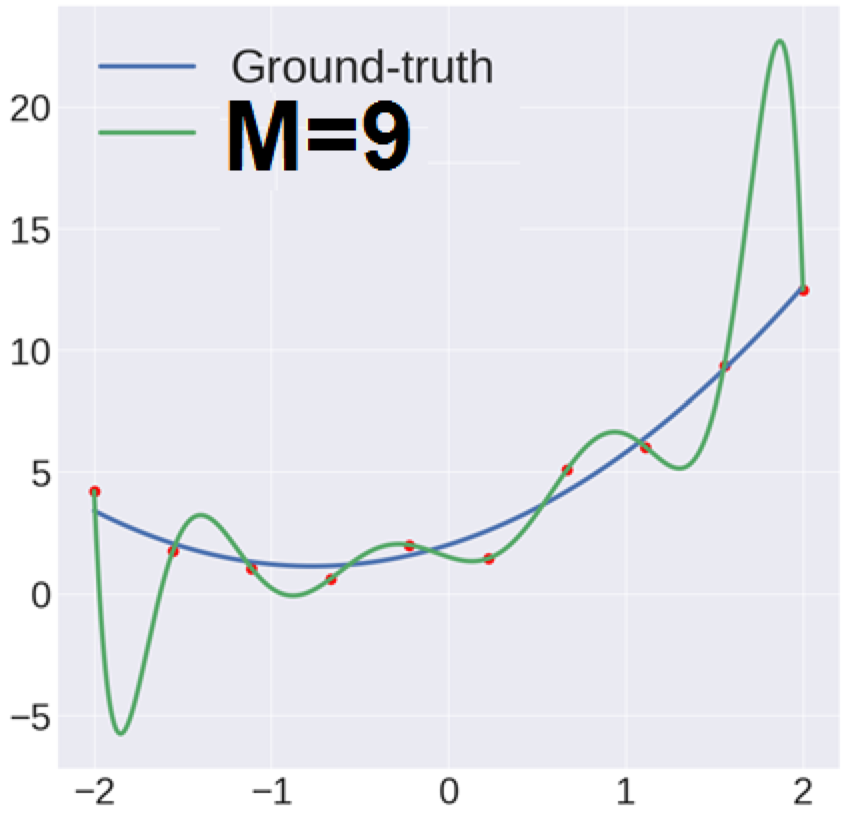
Again, fitting polynomial, but with regularization

$$\underset{\mathrm{W}}{\arg\min} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{W}\|^2 + \lambda \|\boldsymbol{W}\|^2$$

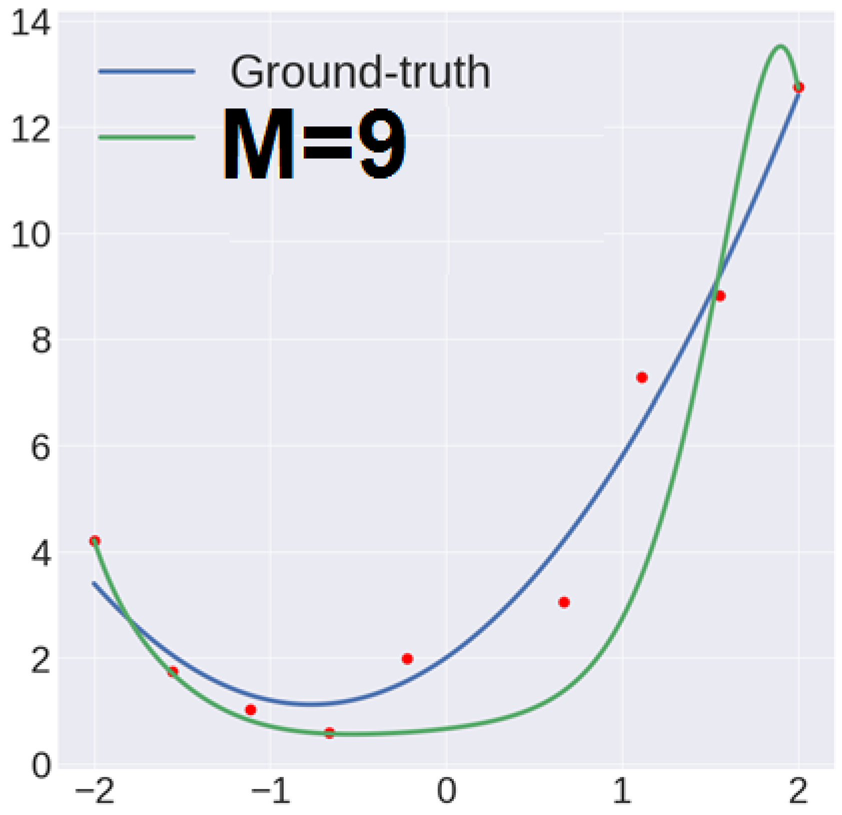
$$\begin{bmatrix} x_d^1 & \cdots & x_2^1 & x_1^1 & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ x_d^N & \cdots & x_2^N & x_1^N & 1 \end{bmatrix} \quad \begin{bmatrix} w_d \\ \vdots \\ w_0 \end{bmatrix}$$

Adding Regularization

No regularization:
fits all data points



Regularization:
can't fit all data points



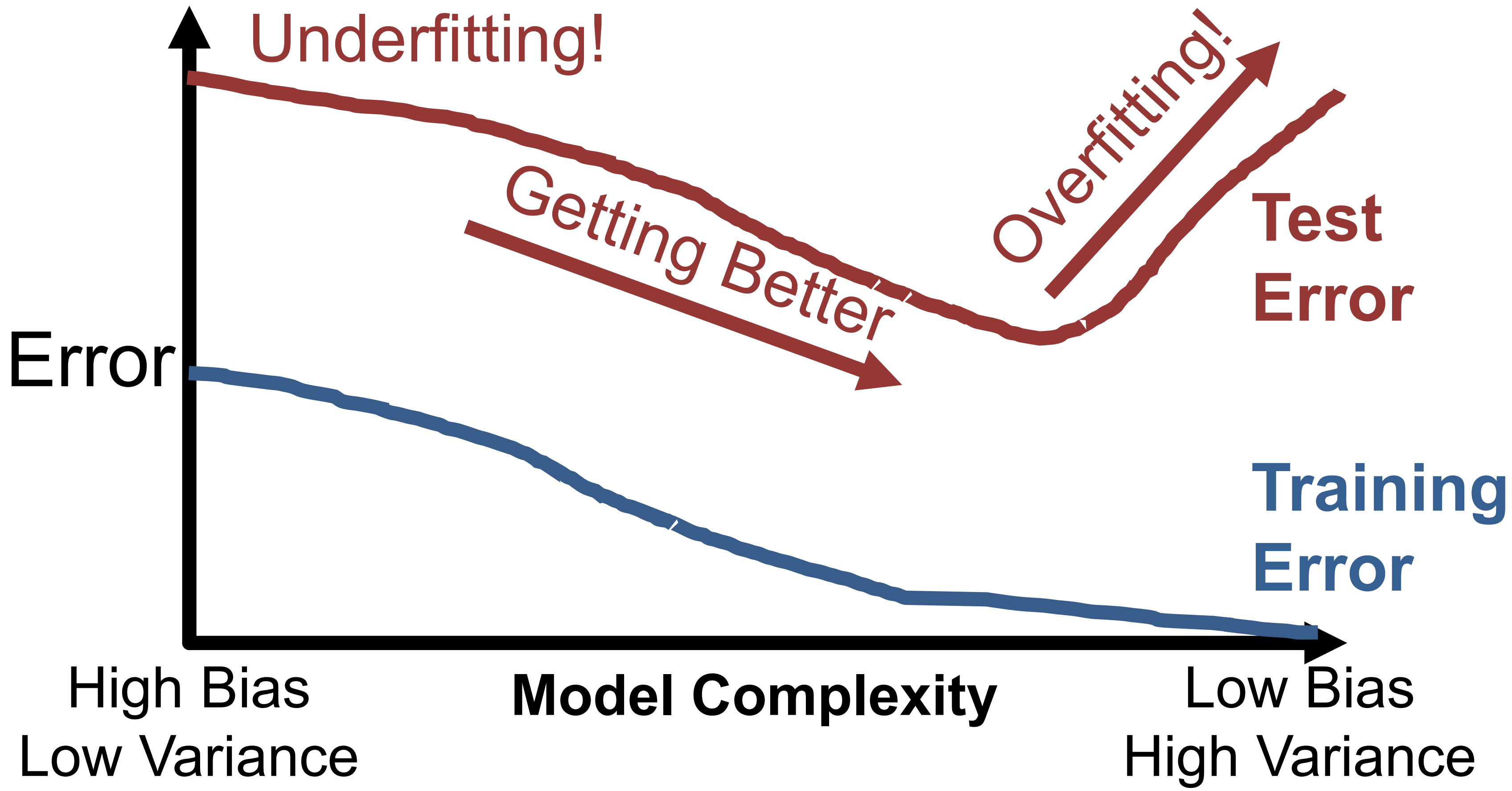
In General

Error on new data comes from combination of:

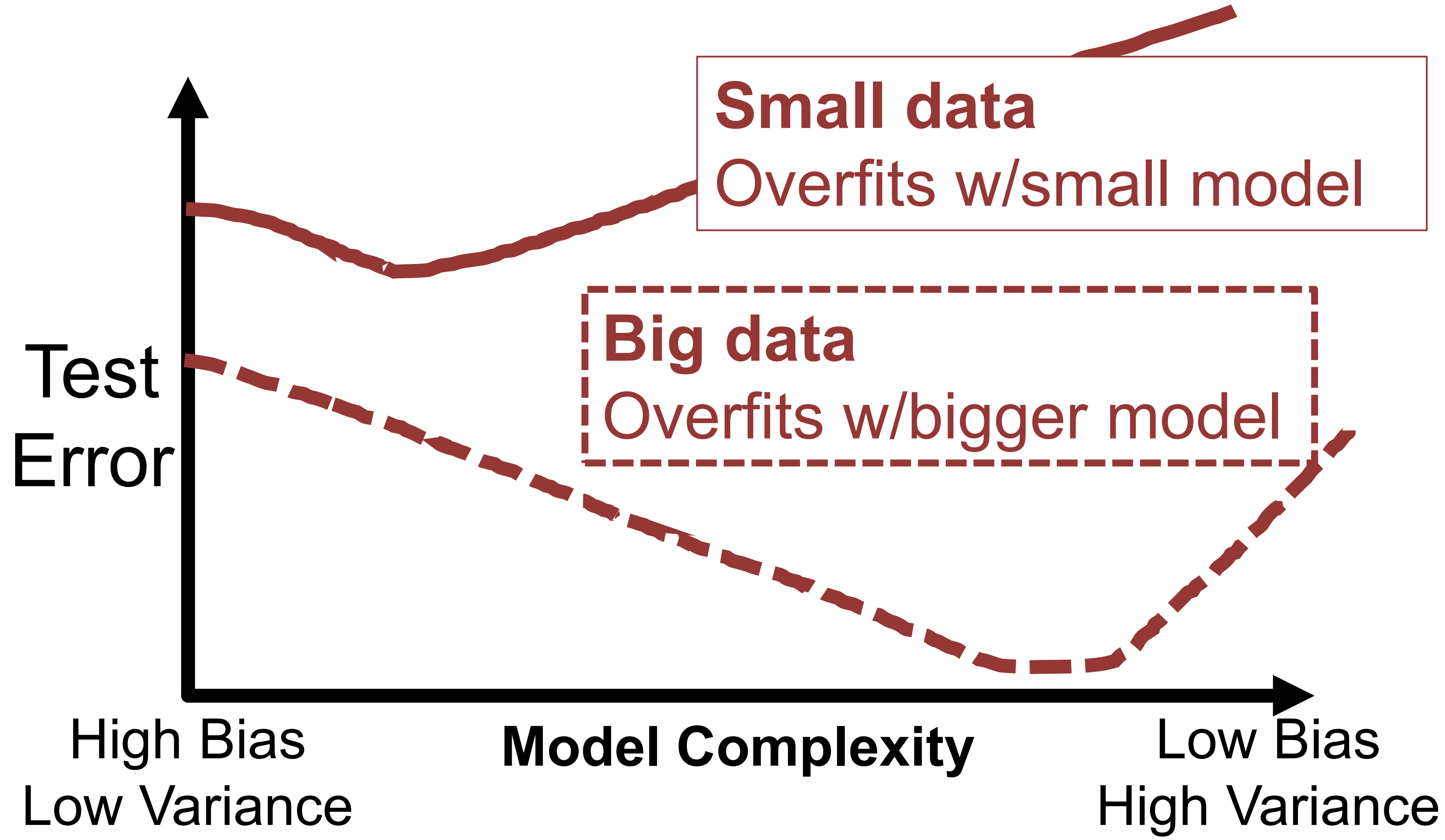
1. **Bias:** model is oversimplified and can’t fit the underlying data
2. **Variance:** you don’t have the ability to estimate your model from limited data
3. **Inherent:** the data is intrinsically difficult

Bias and variance trade-off. Fixing one hurts the other.

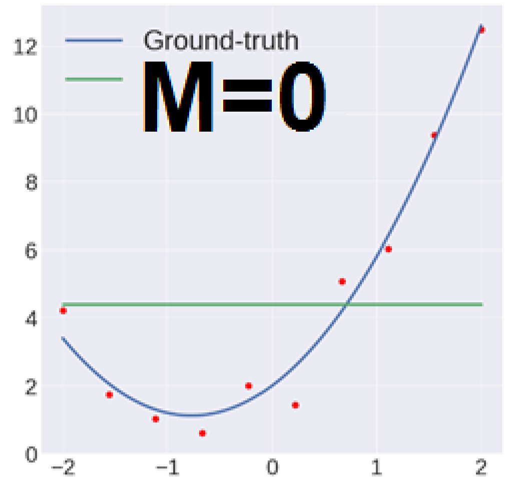
Underfitting and Overfitting

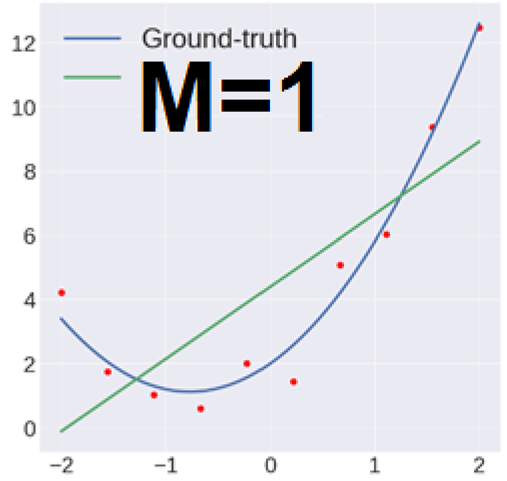


Underfitting and Overfitting



Underfitting



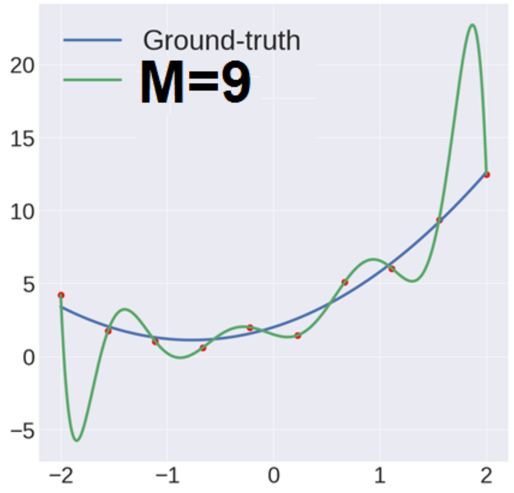


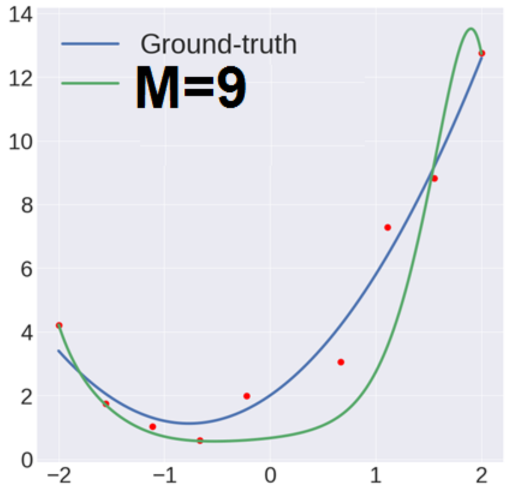
Do poorly on both training and validation data due to bias.

Solution:

1. More features
2. More powerful model
3. Reduce regularization

Overfitting





Do well on training data, but poorly on validation data due to variance

Solution:

1. More data
2. Less powerful model
3. Regularize your model more

Rule: first make sure you *can* overfit, then stop overfitting.

Logistic Regression

Two Categories: Binary Classification Sigmoid Activation Function

This type of loss function implements a fundamental machine learning method, referred to as *logistic regression*. Alternatively, one can use a sigmoid activation function to output $\hat{y} \in (0, 1)$, which indicates the probability that the observed value y is 1. Then, the negative logarithm of $|y/2 - 0.5 + \hat{y}|$ provides the loss, assuming that y is coded from $\{-1, 1\}$. This is because $|y/2 - 0.5 + \hat{y}|$ indicates the probability that the prediction is correct. This observation illustrates that one can use various combinations of activation and loss functions to achieve the same result.

$$L = -\log\left(\left|\frac{y}{2} - \frac{1}{2} + \hat{y}\right|\right)$$

$$\hat{y} = \frac{1}{1 + e^{-(\mathbf{W}.\mathbf{X})}}$$

Classification Decision

If $\widehat{y} = \frac{1}{1+e^{-(\mathbf{W}.\mathbf{X})}}$ >0.5

Prediction is +1

Else Prediction is -1

Maximum Likelihood

We will now describe how the loss function corresponding to likelihood estimation is set up. This methodology is important because it is used widely in many neural models. For positive samples in the training data, we want to maximize $P(y_i = 1)$ and for negative samples, we want to maximize $P(y_i = -1)$. For positive samples satisfying $y_i = 1$, one wants to maximize $\hat{y}_i$ and for negative samples satisfying $y_i = -1$, one wants to maximize $1 - \hat{y}_i$. One can write this casewise maximization in the form of a consolidated expression of always maximizing $|y_i/2 - 0.5 + \hat{y}_i|$. The products of these probabilities must be maximized over all training instances to maximize the likelihood:

$$\text{Likeihood} = \prod_{i=1}^{N} |y_i/2 - 1/2 + \hat{y}_i|$$

What about minimizing its NEGATIVE?

Total Loss

Given a data set, D = {($\mathbf{X}_1$, y_1), ($\mathbf{X}_2$, y_2)... ($\mathbf{X}_N$, y_N)} of labelled feature vectors where y ϵ {-1, +1}, we to estimate the vector **W** that minimizes the following objective/loss function ():-

$$\mathrm{L} = \sum_{i=1}^{N} \mathrm{L_i} = \sum_{i=1}^{N} -\log(|\,y_i/2 - 1/2 + \hat{y}_i|))$$

Total Loss : Gradient Descent (1)

$$\mathrm{L} = -\sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{L_i} = \sum_{\mathrm{i}=1}^{\mathrm{N}} -\log(\,|\,\mathrm{y_i}/2 - 1/2 + \ \widehat{\mathrm{y_i}}|))$$

Total Loss : Gradient Descent (2)

$$\mathrm{L_i} = \begin{cases} -\,log(\widehat{\mathrm{y_i})}\ if\ \mathrm{y_i} = +1 \\ -log(\,1 - \widehat{\mathrm{y_i})}\ if\ \mathrm{y_i} = -1 \end{cases}$$

$$\partial \mathrm{L_i} / \partial\, \mathrm{W} = \begin{cases} \dfrac{(\partial \mathrm{y_i} / \partial\, \mathbf{W})}{\widehat{y_i}}\ if\ \mathrm{y_i} = +1 \\ \dfrac{-(\partial \mathrm{y_i} / \partial\, \mathbf{W})}{1 - \widehat{y_i}}\ if\ \mathrm{y_i} = -1 \end{cases}$$

Total Loss : Gradient Descent (3)

$$\partial \hat{y}_i / \partial \mathbf{W} = \exp(-\mathbf{W}.\mathbf{X}_i)(\hat{y}_i)^2 \mathbf{X}_i$$

$$\partial L_i / \partial \mathbf{W} = \begin{cases} \dfrac{-\mathbf{X}_i}{1 + \exp(\mathbf{W}.\mathbf{X}_i)} & if\ y_i = +1 \\ \dfrac{+X_i}{1 + \exp(-\mathbf{W}.\mathbf{X}_i)} & if\ y_i = -1 \end{cases}$$

Total Loss : Gradient Descent (4)

$$\frac{\partial \mathrm{L_i}}{\partial \mathbf{W}} = \frac{-\mathrm{y_i}\mathbf{X}_\mathrm{i}}{1 + \exp(\mathrm{y_i}\mathbf{W}.\mathbf{X}_\mathrm{i})}$$