Introduction to Computer Vision

11. Learning from data

UCLA – CS 188 – Fall 2019

Fabien Scalzo, Ph.D.

Week 1			26-Sep	**Introduction**
Week 2	1-Oct	**Basic Image Processing**	3-Oct	**Feature Extraction and Classification**
Week 3	8-Oct	**Feature Tracking/Optical Flow**	10-Oct	**SVD, 2D camera model, projective plane**
Week 4	15-Oct	**2D Image transformations, RANSAC**	17-Oct	**Euclidean geometry, rigid body motion**
Week 5	22-Oct	**Epipolar Geometry**	24-Oct	**3D Cameras and processing**
Week 6	29-Oct	**Midterm**	31-Oct	**3D Cameras and processing**
Week 7	5-Nov	**Learning from data**	7-Nov	**Neural Networks**
Week 8	12-Nov	**Deep Learning**	14-Nov	**Deep Learning**
Week 9	19-Nov	**Object Detection**	21-Nov	**Generative Models**
Week 10	26-Nov	**Guest Lecture (Nikhil Naik)**	28-Nov	
Week 11	3-Dec	**Applications**	5-Dec	**Recap**
Week 12	10-Dec		12-Dec	**Final**

- The learning problem
- Linear Regression & Classification
- Loss, Risk, Empirical Risk
- Gradient descent
  - Numerical, Analytical
  - General Algorithm
  - Types of gradient descent
- Under & Overfitting
- Regularization
- Cross-validation

The learning problem

Goal

To find the parameters of the model f such that it can map any point from input space X to the output space Y.

$$f: X \rightarrow Y$$

Sampled observations $\{\boldsymbol{x_i}, \boldsymbol{y_i}\}_{\boldsymbol{N}}$ from joint distribution P(x,y)

- pairs of input variables $x_i \in \mathbf{X}$ and output labels $y_i \in \mathbf{Y}$
- $\{\boldsymbol{x_i}, \boldsymbol{y_i}\}_{\boldsymbol{N}}$ denotes a set of N pairs of labeled data points
- sampled (i.i.d) from a process/distribution that is typically unknown

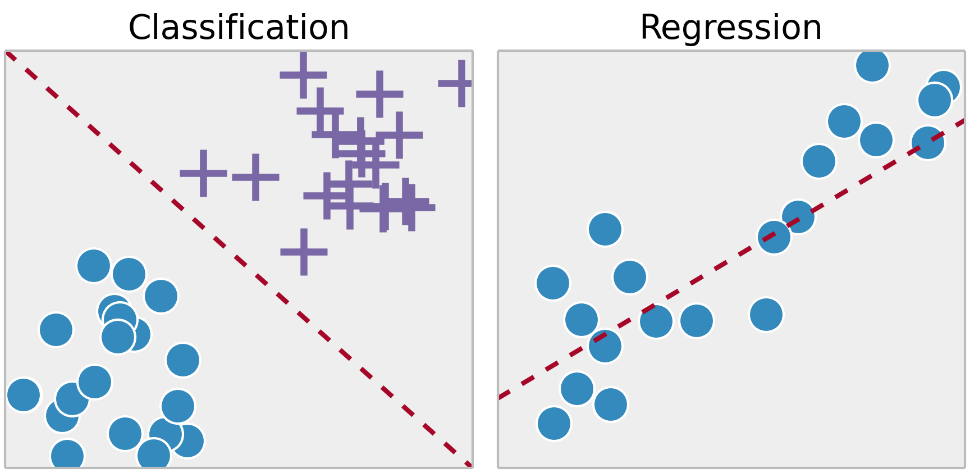
The learning problem

Observed data

- pairs of input variables $x_i \in \mathbf{X}$ and output labels $y_i \in \mathbf{Y}$
- $\{\boldsymbol{x_i}, \boldsymbol{y_i}\}_{\boldsymbol{N}}$ denotes a set of N pairs of labeled data points
- sampled (iid) from a process that is typically unknown -> our dataset

Categorical $\mathbf{Y}$

$\boldsymbol{Y} \in \{\mathbf{0}, \mathbf{1}\}$



Continuous $\mathbf{Y}$

$\mathbf{Y} \in \boldsymbol{R}$

Loss, Risk, and Empirical Risk

How well will the model work in practice on new data?

Risk: $$R(h) = \mathbf{E}[L(h(x), y)] = \int L(h(x), y)\, dP(x, y).$$

In general, the risk R(h) cannot be computed because the distribution P(x,y) is unknown

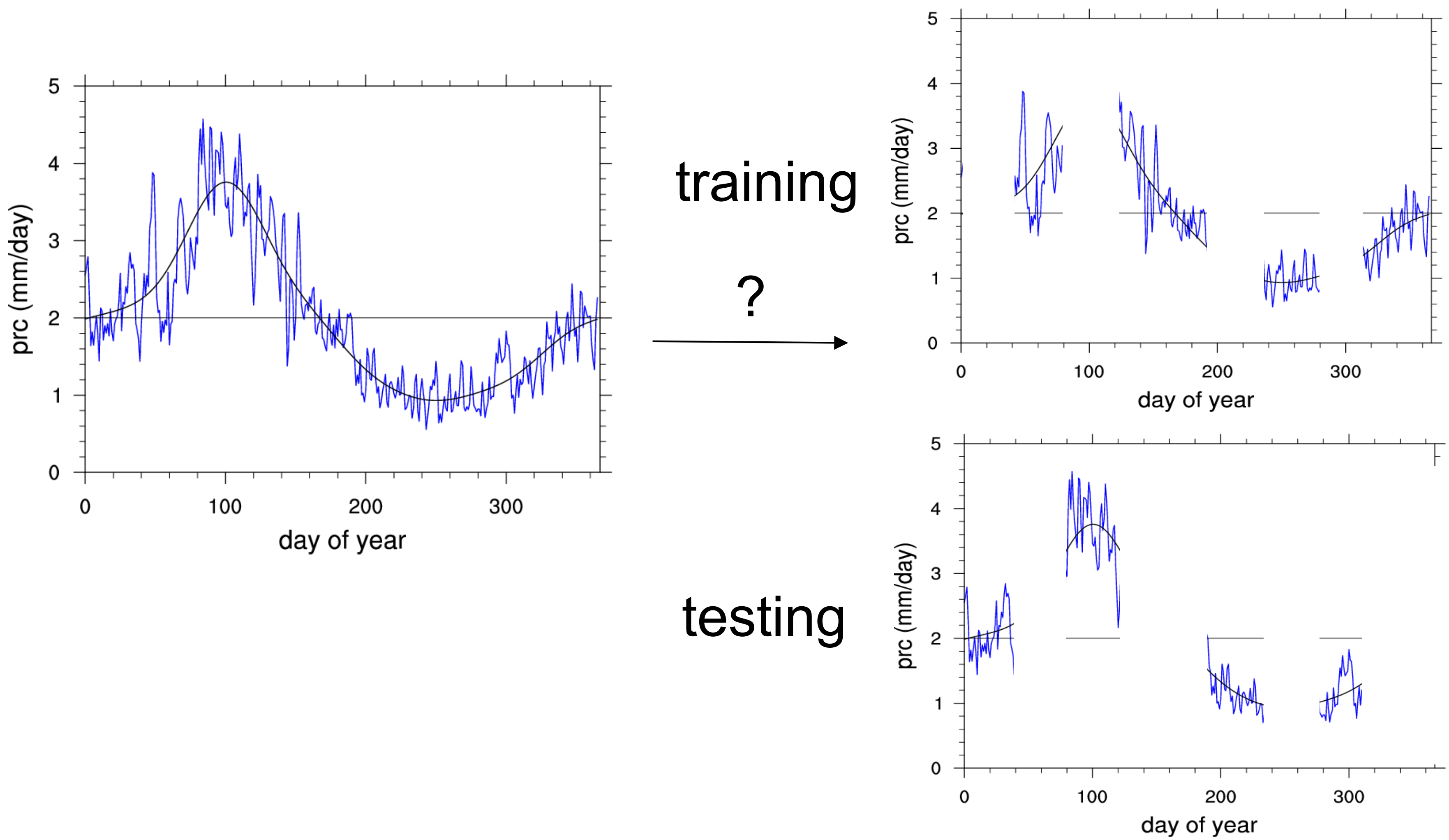
Loss: measures how different the predicted output is from the groundtruth

Example: $$L(\hat{y}, y) = \begin{cases} 1 & \text{If} \quad \hat{y} \neq y \\ 0 & \text{If} \quad \hat{y} = y \end{cases}.$$

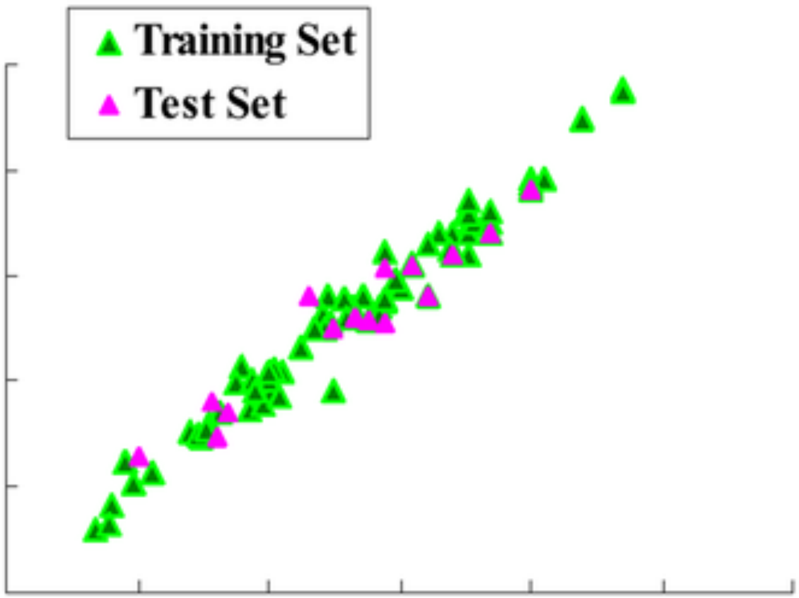
Empirical Risk: Approximation to true risk R(h); average loss over dataset

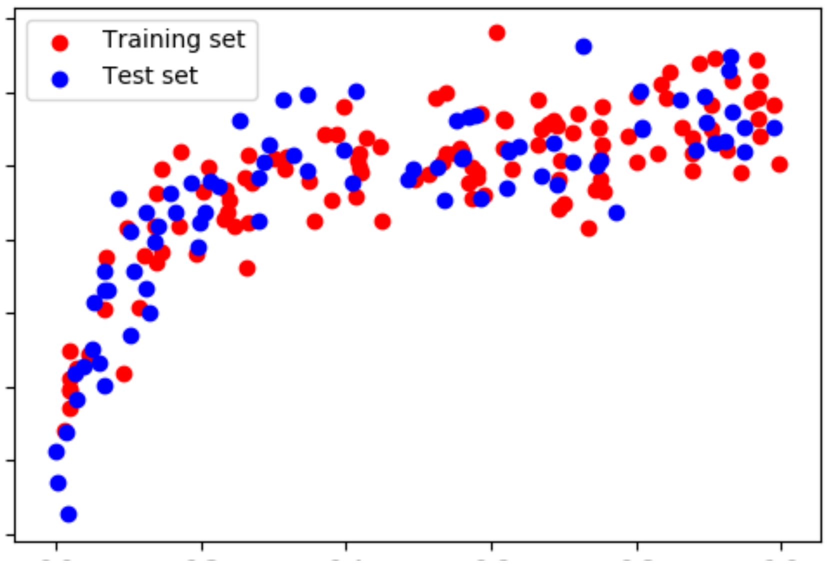
$$R_{\text{emp}}(h) = \frac{1}{n} \sum_{i=1}^{n} L(h(x_i), y_i).$$

Separate dataset into training and test set



Separate dataset into training and test set



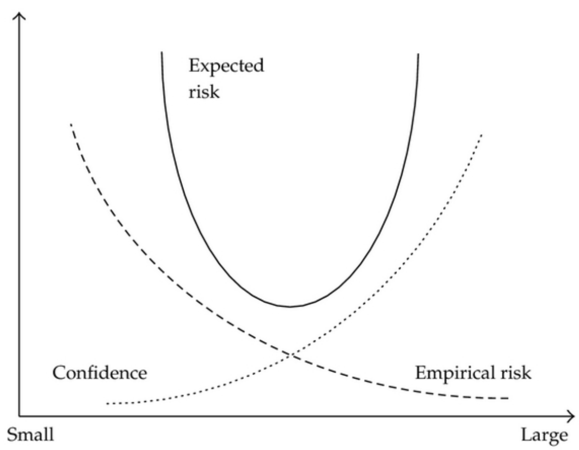


Minimizing the Empirical Risk

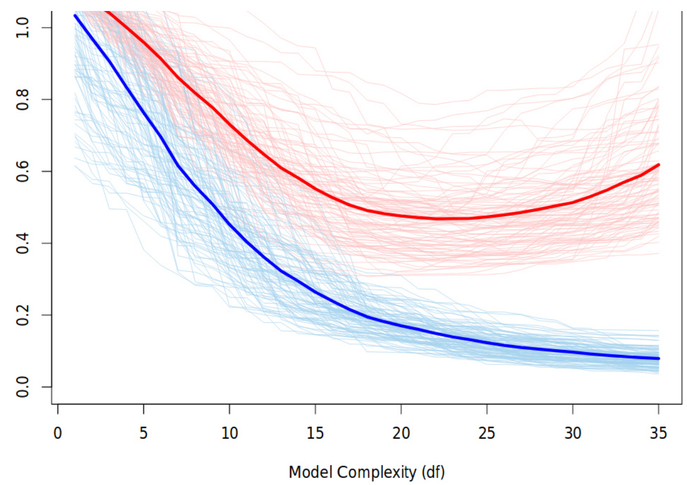
$$R_{\text{emp}}(h) = \frac{1}{n} \sum_{i=1}^{n} L(h(x_i), y_i).$$

$$\hat{h} = \arg\min_{h \in \mathcal{H}} R_{\text{emp}}(h).$$

on training set
and test set



Model complexity



Loss functions for regression

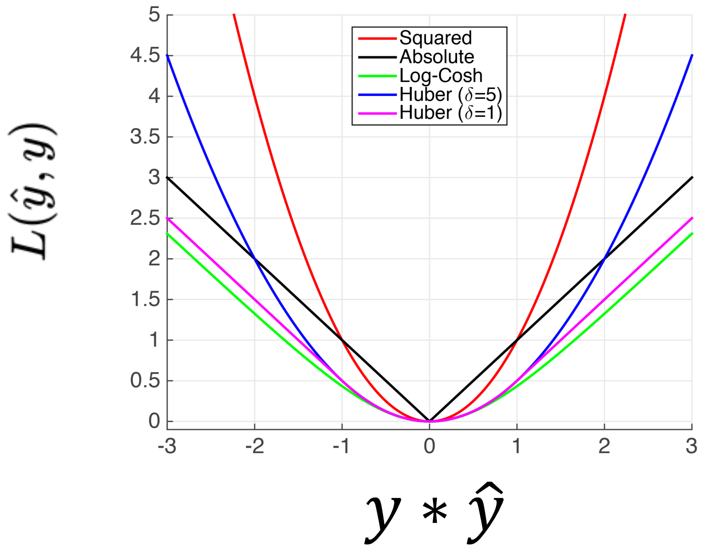
1.Squared Loss $(h(\mathbf{x}_i) - y_i)^2$

2.Absolute Loss $|h(\mathbf{x}_i) - y_i|$

3.Huber Loss

- $\frac{1}{2}(h(\mathbf{x}_i) - y_i)^2$ if $|h(\mathbf{x}_i) - y_i| < \delta$,
- otherwise $\delta(|h(\mathbf{x}_i) - y_i| - \frac{\delta}{2})$

4.Log-Cosh Loss
$log(cosh(h(\mathbf{x}_i) - y_i))$,
$cosh(x) = \frac{e^x + e^{-x}}{2}$



Loss functions for classification

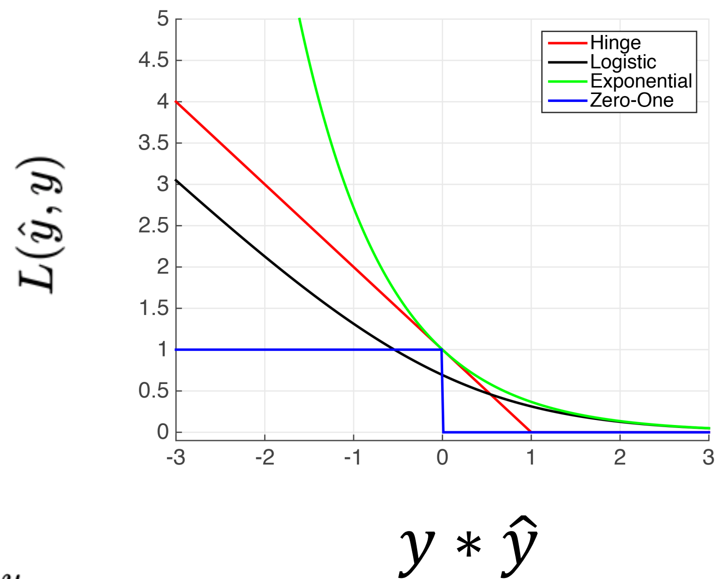
1.Hinge-Loss
$max[1 - h_{\mathbf{w}}(\mathbf{x}_i)y_i, 0]^p$

2.Log-Loss
$log(1 + e^{-h_{\mathbf{w}}(\mathbf{x}_i)y_i})$

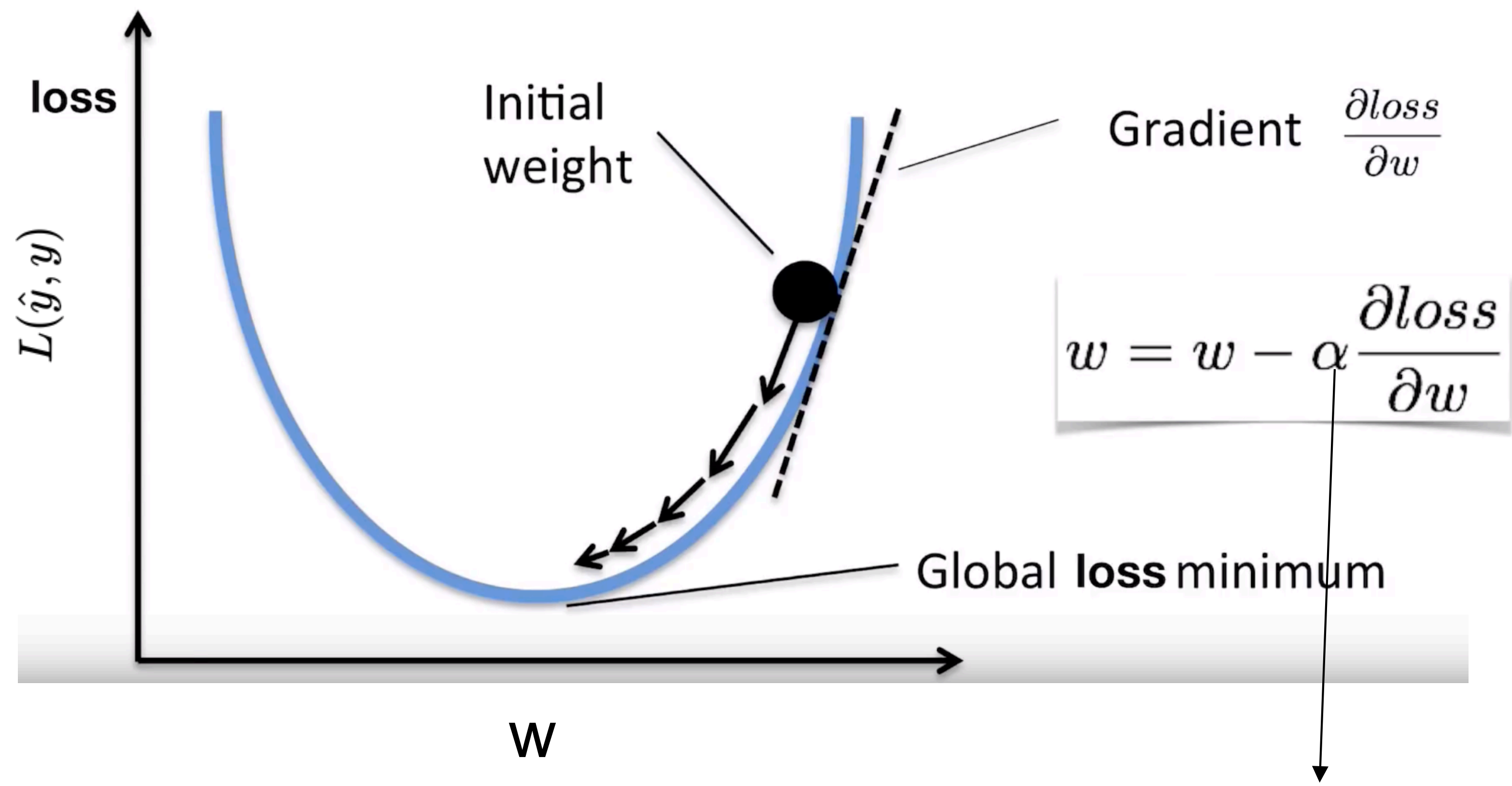
3.Exponential Loss
$e^{-h_{\mathbf{w}}(\mathbf{x}_i)y_i}$

4.Zero-One Loss
$\delta(\text{sign}(h_{\mathbf{w}}(\mathbf{x}_i)) \neq y_i) = \begin{cases} 1 & \text{If} \quad \hat{y} \neq y \\ 0 & \text{If} \quad \hat{y} = y \end{cases}.$

$\boldsymbol{Y} \in \{-\mathbf{1}, +\mathbf{1}\}$



Gradient descent



Learning rate

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Computing the Gradient

How do we compute the gradient $\frac{\partial loss}{\partial w}$

- Numerical

Evaluate the difference of the Loss at $\mathrm{L}(\mathrm{w})$ and $\mathrm{L}(\mathrm{w}+\epsilon)$

- Analytical

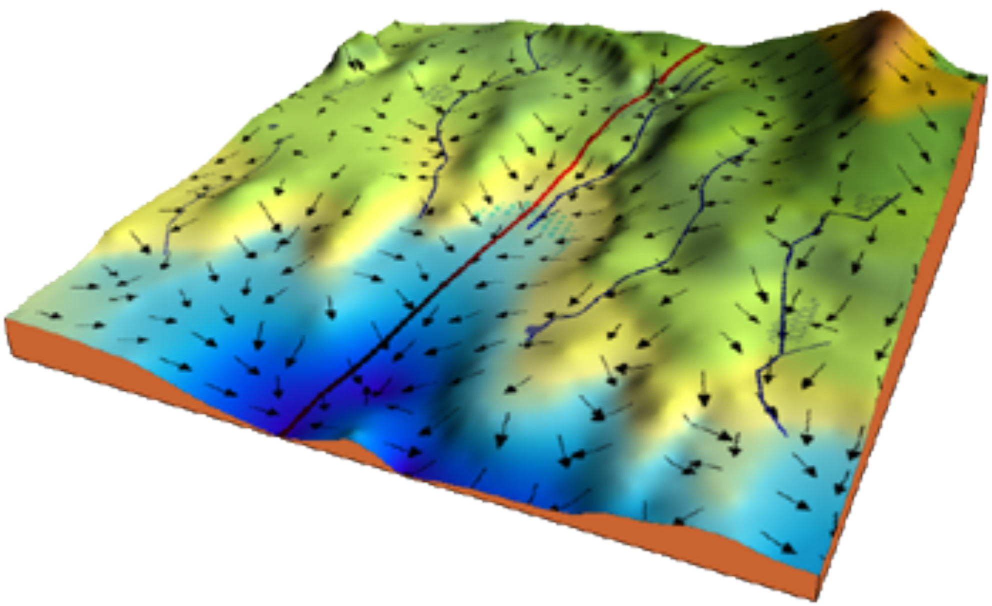
Given the cost function:

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

The gradient can be calculated as:

$$f'(m, b) = \begin{bmatrix} \frac{df}{dm} \\ \frac{df}{db} \end{bmatrix} = \begin{bmatrix} \frac{1}{N} \sum -2x_i(y_i - (mx_i + b)) \\ \frac{1}{N} \sum -2(y_i - (mx_i + b)) \end{bmatrix}$$

Gradient



Gradient Descent Algorithms

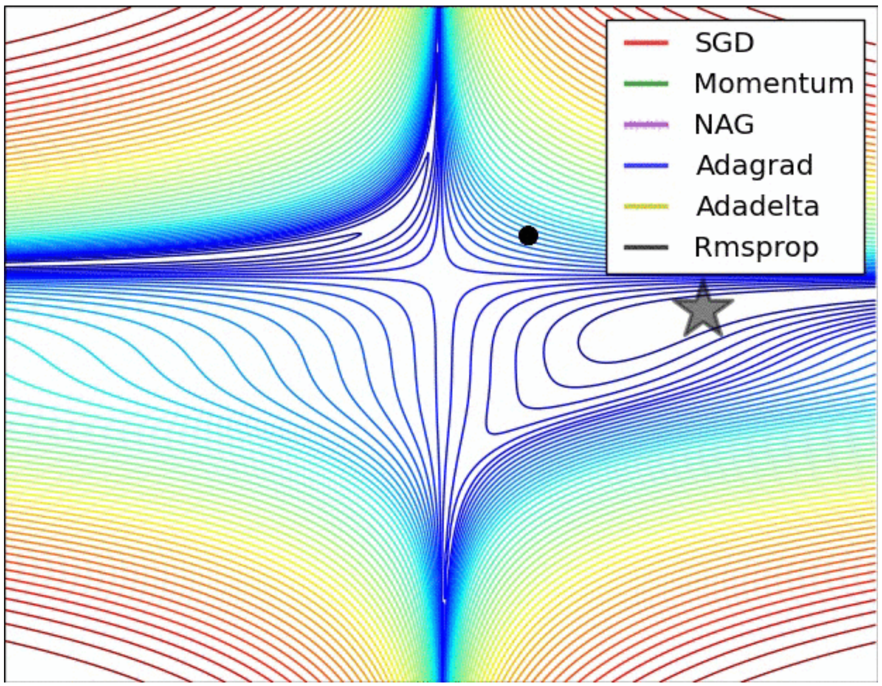
- Gradient descent variants
  - Batch gradient descent
    - Error is computed over all training samples at each iteration
  - Stochastic gradient descent
    - Error is computed over one training sample at each iteration
  - Mini-batch gradient descent
    - Error is computed over a subset of the training samples at each iteration

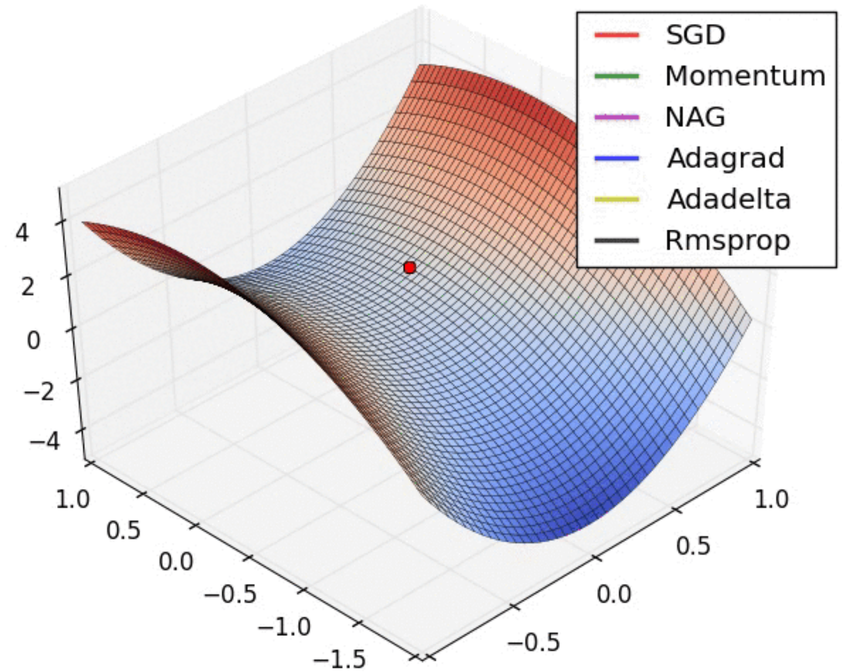
Gradient Descent Algorithms

- Momentum
- Nesterov accelerated gradient
- Adagrad
- Adadelta
- RMSprop
- Adam
- AdaMax
- Nadam
- AMSGrad

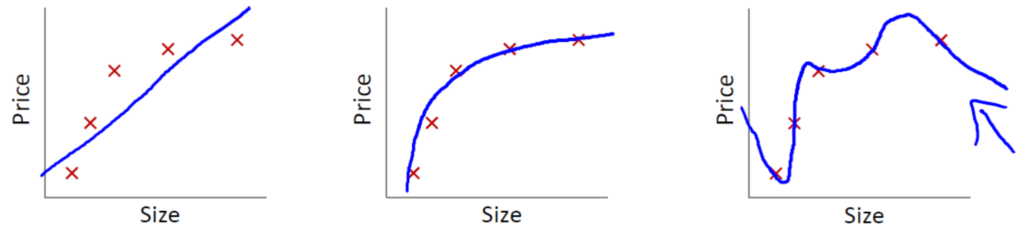
Gradient Descent with Momentum considers the past gradients during the update

Gradient Descent Algorithms





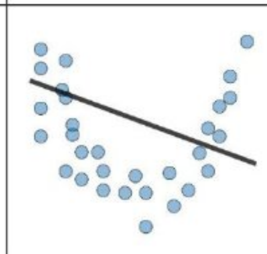
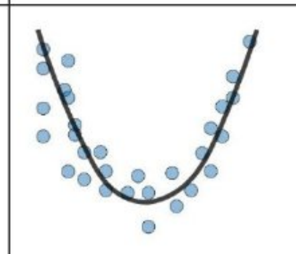
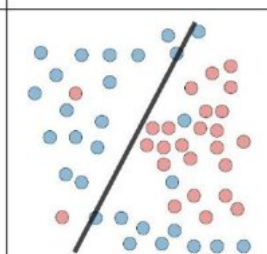
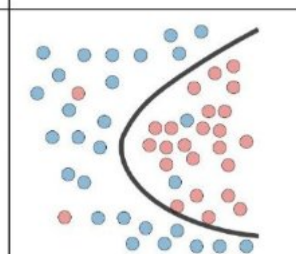
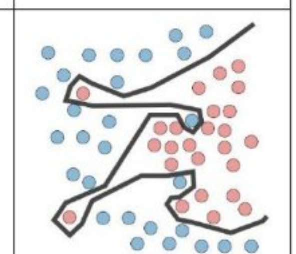
Fitting Data



$\rightarrow \theta_0 + \theta_1 x$ "Underfit" "High bias"

$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$ "Just right"

$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$ "Overfit" "High variance"

	Underfitting	Just right	Overfitting
Symptoms	- High training error - Training error close to test error - High bias	- Training error slightly lower than test error	- Low training error - Training error much lower than test error - High variance
Regression			
Classification			

Regularization

The process of making the prediction function fit the training data less well in the hope that it generalizes new data better. Regularization penalizes large values for w

ℓ_0 complexity: number of non-zero coefficients
ℓ_1 "lasso" complexity: $\sum_{i=1}^{d} |w_i|$, for coefficients $w_1, \ldots, w_d$
ℓ_2 "ridge" complexity: $\sum_{i=1}^{d} w_i^2$ for coefficients $w_1, \ldots, w_d$

Regularized Empirical Risk Minimization

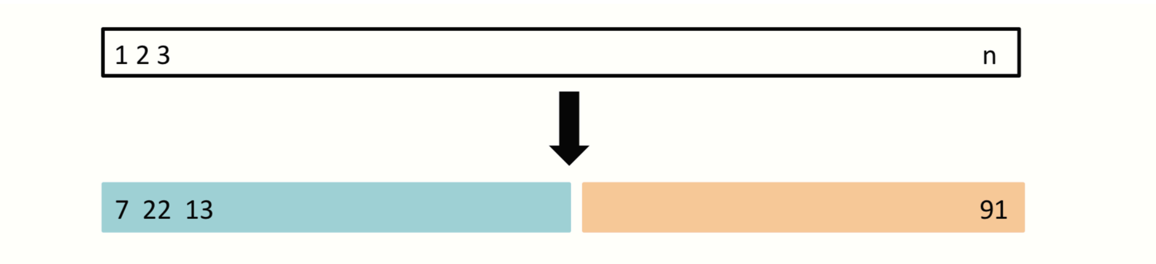
$$\operatorname*{argmin}_{\mathbf{w}} \underbrace{R(\mathbf{w})}_{\text{Regularization}} + C \sum_{i=1}^{N} \underbrace{L(y_i, \mathbf{w}^\mathsf{T}\mathbf{x}_i)}_{\text{Loss}}$$

Cross-validation

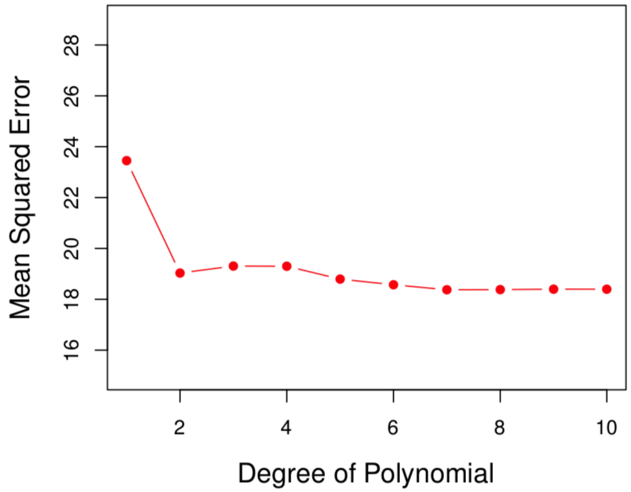
Goal: Estimate the test error for a supervised learning method.

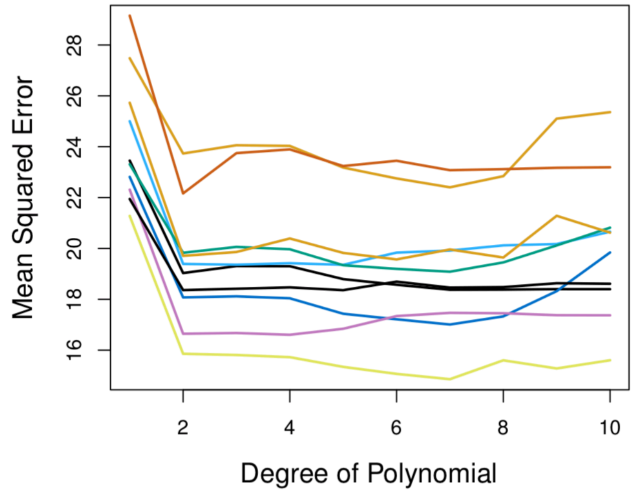
Strategy:

- Split the data in two parts.
- Train the method in the first part.
- Compute the error on the second part.



Cross-validation



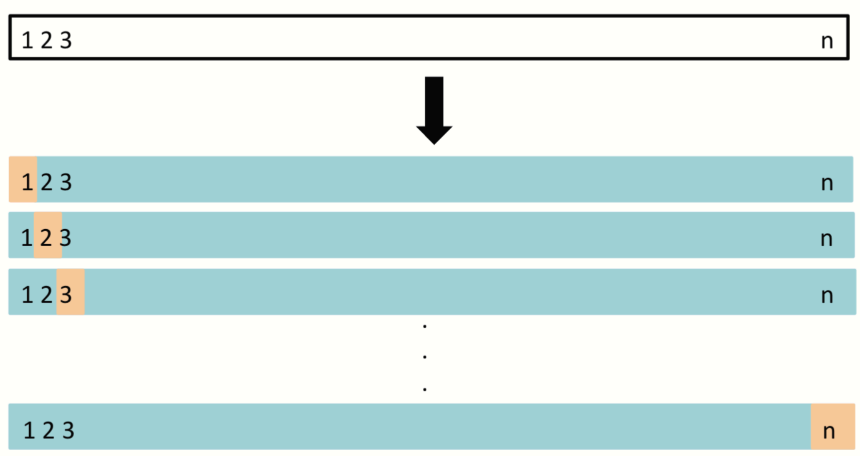


Problems: 1. Every split yields a different estimate of the error. 2. Only a subset of points is used to evaluate the model.

Cross-validation

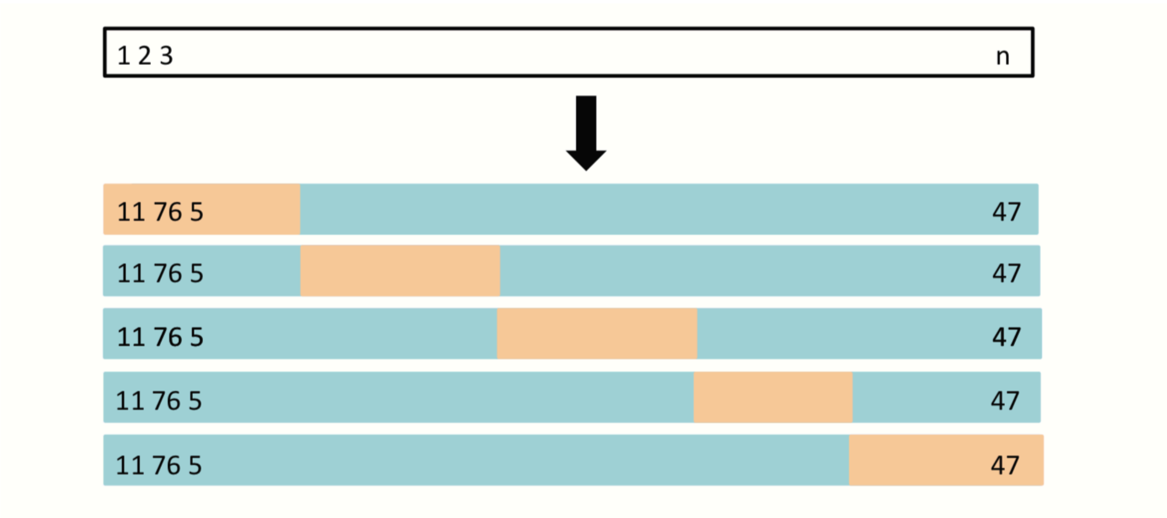
Leave one out cross-validation

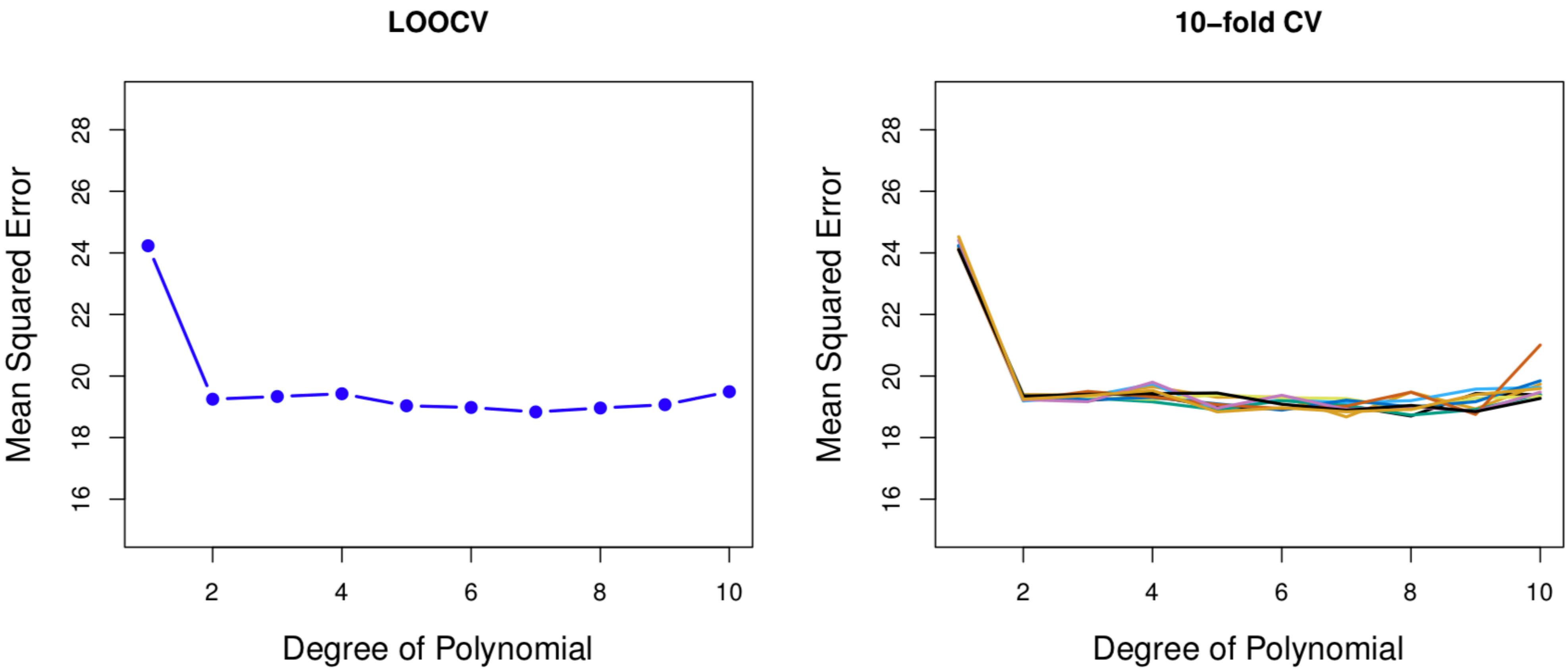
- For every $i = 1, \dots, n$:
  - train the model on every point except i,
  - compute the test error on the held out point.
- Average the test errors.



k-fold cross-validation

- Split the data into k subsets or *folds*.
- For every $i = 1, \ldots, k$:
  - train the model on every fold except the ith fold,
  - compute the test error on the ith fold.
- Average the test errors.




LOOCV
10−fold CV
Mean Squared Error
16
18
20
22
24
26
28
2
4
6
8
10
Degree of Polynomial

The wrong way to do cross validation

Reading: Section 7.10.2 of The Elements of Statistical Learning.

We want to classify 200 individuals according to whether they have cancer or not. We use logistic regression onto 1000 measurements of gene expression.

Proposed strategy:

- Using all the data, select the 20 most significant genes using z-tests.
- Estimate the test error of logistic regression with these 20 predictors via 10-fold cross validation.

The wrong way to do cross validation

To see how that works, let's use the following simulated data:

- Each gene expression is standard normal and independent of all others.
- The response (cancer or not) is sampled from a coin flip — no correlation to any of the "genes".

What should the misclassification rate be for any classification method using these predictors?

Roughly 50%.

The wrong way to do cross validation

We run this simulation, and obtain a CV error rate of 3%!

Why is this?

- Since we only have 200 individuals in total, among 1000 variables, at least some will be correlated with the response.
- We do variable selection using *all the data*, so the variables we select have some correlation with the response in every subset or fold in the cross validation.

The **right** way to do cross validation

- Divide the data into 10 folds.
- For $i = 1, \ldots, 10$:
  - Using every fold except i, perform the variable selection and fit the model with the selected variables.
  - Compute the error on fold i.
- Average the 10 test errors obtained.

In our simulation, this produces an error estimate of close to 50%.

Moral of the story: Every aspect of the learning method that involves using the data — variable selection, for example — must be cross-validated.