



# Machine Learning

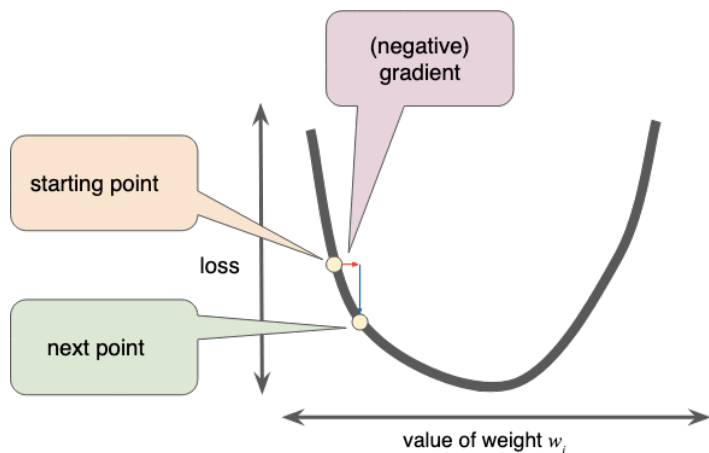
Prof. Adil Khan

# Objectives

1. A quick recap of last week
2. A quick recap of last week
3. Hyperparameter tuning and cross-validation
4. What is regularization? Why is it important? How does it work?
  - Constrained Optimization
  - L1 vs L2 as constrained optimization problems
5. High Dimensional Data
6. What is Principal Component Analysis? How does it help with high dimensional data? What is its objective function? How is it motivated?

# Recap (1)

## Gradient Descent



Repeat {

$$w_j = w_j - \alpha \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i) x_i^j$$

}

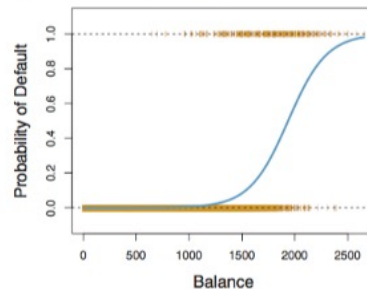
## Classification

- Eye color  $\in \{brown, blue, green\}$
- Email  $\in \{Spam, Ham\}$
- Expression  $\in \{Happy, Sad, Angry\}$
- Action  $\in \{Walking, Running, Cycling\}$

## Logistic Regression

$$p(x) = \frac{1}{1 + e^{-z}}$$

$$z = w_0 + w_1x_1 + \dots + w_px_p$$



# Recap (2)

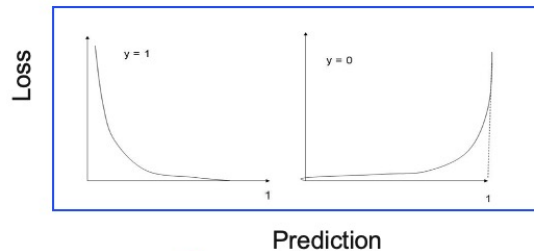
## MSE Loss Function for Logistic Regression

$$y = \frac{1}{1 + e^{-w^T x^i}}$$

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n (\mathbf{y}^i - \mathbf{w}^T \mathbf{x}^i)^2$$

- Due to non-linearity of the sigmoid function in hypothesis of Logistic Regression, MSE is **not convex** anymore

## Loss Function of Logistic Regression



$$\mathcal{L}(\mathbf{w}) = -\frac{1}{n} \sum_{i=1}^n y^i \log(p(x^i)) + (1 - y^i) \log(1 - p(x^i))$$

$$\underset{w_0, w_1}{\operatorname{argmin}} \mathcal{L}(\mathbf{w})$$

We solve it using **Gradient Descent**

$$w_j = w_j - \alpha \frac{1}{n} \sum_{i=1}^n (p(x^i) - \mathbf{y}^i) x_j^i$$

# Recap (3)

$$Acc = 1 - \frac{\# \text{ of missclassified examples}}{\# \text{ of samples}}$$

$$Precision = \frac{True \text{ Positives}}{True \text{ Positives} + False \text{ Positives}}$$

$$Recall = \frac{True \text{ Positives}}{True \text{ Positives} + False \text{ Negatives}}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

## Class-imbalance

Data For Cat / Dog Classifier



Data For Normal / Fraud Classifier



## Confusion Matrix

		Predicted Class Label	
		Negative	Positive
Actual Class Label	Negative	True Negatives	False Positives
	Positive	False Negatives	True Positives

# Extra: Relationship b/w MLE and Log-loss

- **Logistic Regression Model:**

$$\hat{y}_i = P(y_i = 1|X_i; w) = \frac{1}{1 + e^{-(X_i \cdot w)}}$$

- **Likelihood of Observing  $y_i$ :**

$$L(w|X_i, y_i) = \hat{y}_i^{y_i} \cdot (1 - \hat{y}_i)^{1-y_i}$$

- **Log-Likelihood (LL):**

$$\log L(w) = \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

- **Maximum Likelihood Estimation (MLE):**

– MLE finds parameters  $w$  that **maximize** the log-likelihood, i.e., maximize the probability of the observed data.

- **Log-Loss** (Negative Log-Likelihood):

$$\text{Log-Loss} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

– Minimizing log-loss is **equivalent** to maximizing the likelihood in MLE.

# Hyperparameters

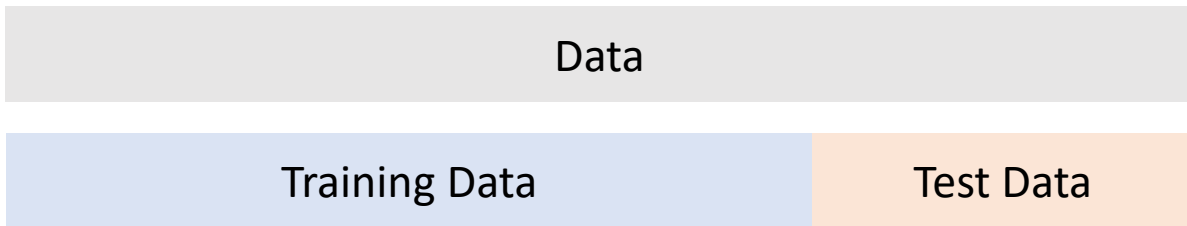
# Hyperparameters

- Hyperparameters
  - In ML, a hyperparameter is a parameter whose value is used to control the learning process
  - They are set (by the us) not learned like model parameters
- Examples:
  - $\alpha$  in GD



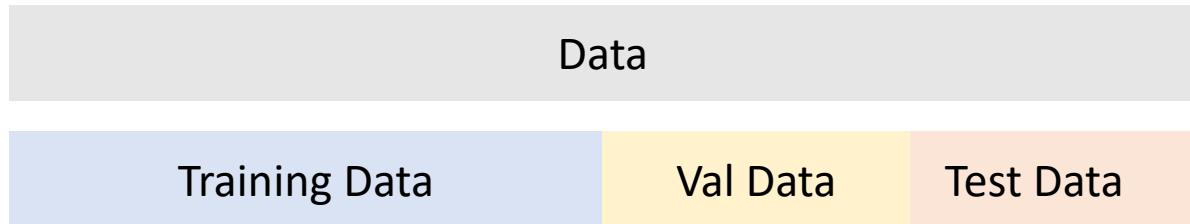
# Validation and Test Sets

- We can tune hyperparameters using the validation set
- But what is a validation set? Is it the same as test set?
- To answer this, let's see how we were splitting our data so far

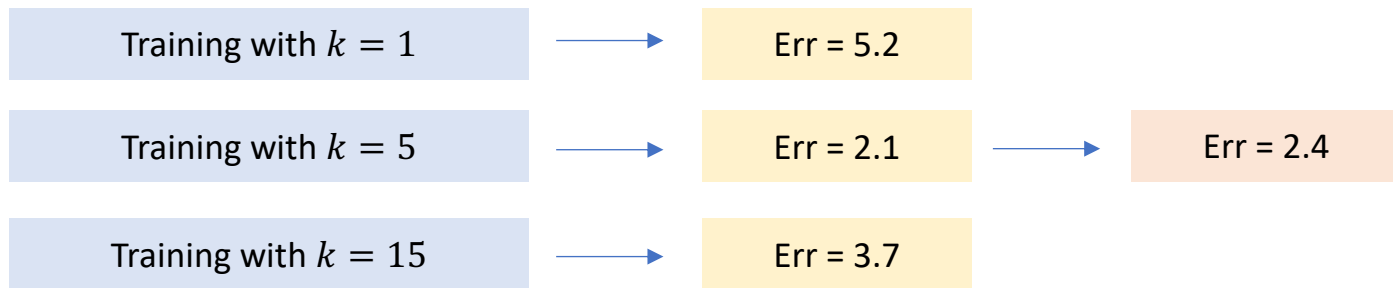


# Validation and Test Sets (2)

- But when we have hyperparameters to tune, we split as follows



- As we train model as follows



# Validation and Test Sets (2)

- Solves the “Model Selection” problem
  - Selecting the optimal hyperparameters for a given machine learning model
- But there is another problem
  - Once we have chosen a model, how can we estimate its true error?
  - Yes, we are using an independent “test” set to estimate the true error, BUT
  - The true error is a model’s error when tested on the ENTIRE POPULATION

# K-fold Cross-Validation

$$E = \frac{1}{K} \sum_{i=1}^K E_i$$

- Create K-fold partition of the dataset

K = 3

	Data		
	Fold 1	Fold 2	Fold 3
Exp: 1	Val	Train	Train
Exp: 2	Train	Val	Train
Exp: 3	Train	Tain	Val

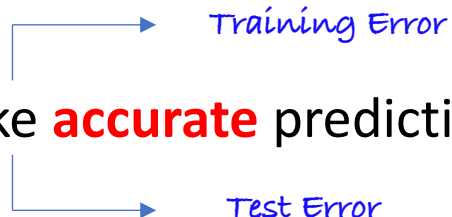
# Regularization

# Recall

*Goal of Machine Learning*

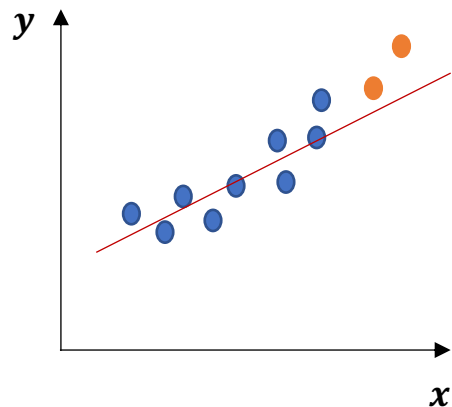
$$f: x \rightarrow y$$

- So that we can use the learned function to make predictions on **UNSEEN** data
- For example, we can use it to predict the risk-scores of contracting COVID of **NEW** patients based on their clinical symptoms
- And, naturally, we would like to make **accurate** predictions



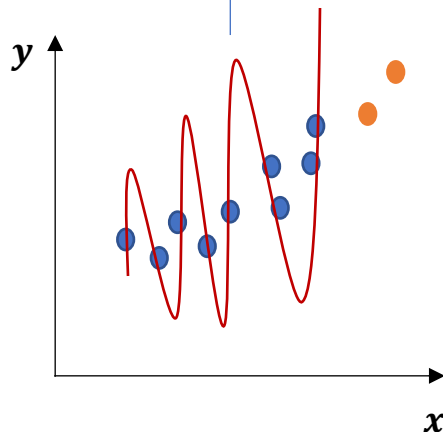
# What can go wrong?

- Training Data
- Unseen Test Data

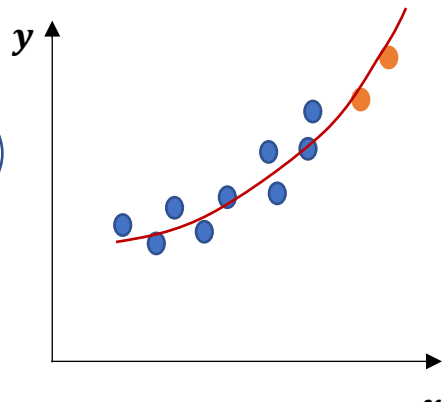


$$\frac{y = w_0 + w_1 x}{f}$$

Smallest Training Error  
Most Complex  
Largest Test Error



$$\frac{y = w_0 + w_1 x + w_2 x^2 + w_3 x^8}{f}$$



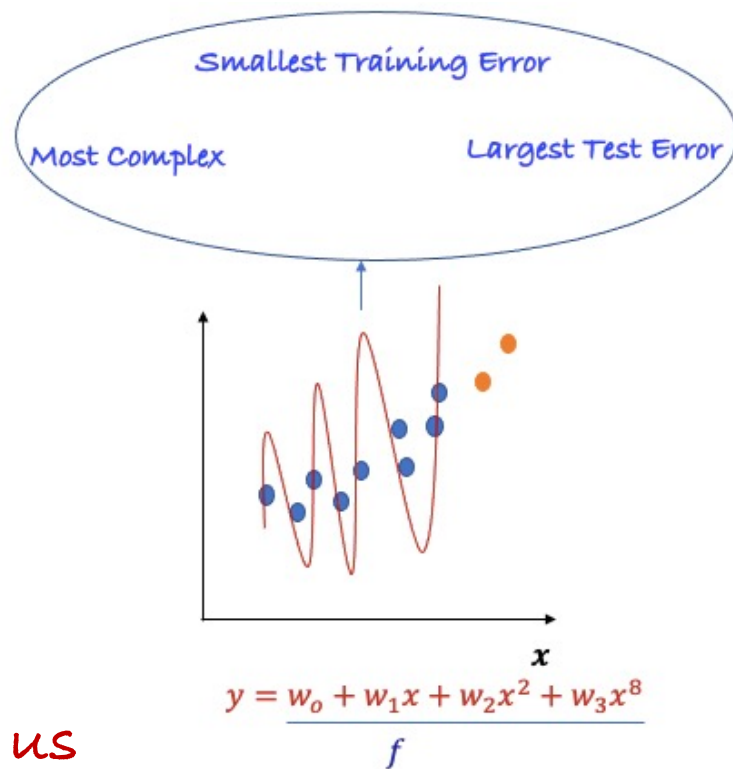
$$\frac{y = w_0 + w_1 x + w_2 x^2}{f}$$

# Why does it happen?

- It happens because when we are learning the function, we measure the success as

*Success = Goodness of Fit*

This is where Regularization helps us





# How does Regularization help?

- Improves the “**test error**” by compromising on the “**training error**”
- To achieve this, the most common approach is to change the way we measure success of learning as

*Success = Goodness of Fit + Simplicity of the Model*


*which is usually implemented as*


*Success = Goodness of Fit +  $\lambda$  Simplicity of the Model*

*where  $\lambda \in [0, \infty)$*

# Mathematically

*Success = Goodness of the Fit +  $\lambda$  Simplicity of the model*


$$\frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2$$


$$\sum_{j=1}^p w_j^2$$

# $L_2$ Regularization

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2 + \lambda \sum_{j=1}^p w_j^2$$

$\lambda \geq 0$  is a tuning parameter

Ridge Regression

# $L_2$ Regularization

1. If  $\lambda$  is at a “good” value, regularization helps to avoid overfitting
2. Choosing  $\lambda$  may be hard: cross-validation is often used
3. If there are irrelevant features in the input (i.e. features that do not affect the output),  $L_2$  will give them small, but non-zero weights
4. Ideally, irrelevant input should have weights exactly equal to 0

# $L_1$ Regularization

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2 + \lambda \sum_{j=1}^p |w_j|$$

$\lambda \geq 0$  is a tuning parameter

Lasso Regression

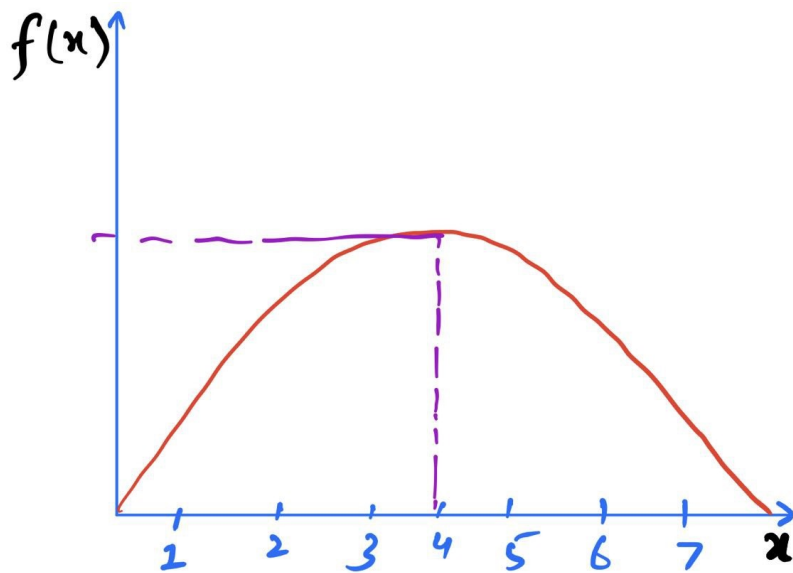
# $L_1$ Regularization

- Similar to  $L_2$ ,  $L_1$  shrinks the coefficient estimates towards zero
- However,  $L_1$  penalty has the effect of forcing some of the coefficient estimates to be **exactly equal to zero** when the tuning parameter  $\lambda$  is sufficiently large
- Hence, much like best subset selection,  $L_1$  performs variable selection
- As a result, models are generally much easier to interpret
- We say that the  $L_1$  yields **sparse models**

Why does  $L_1$  regularization give sparse models but  $L_2$  does not?

# Optimization Problem

- Mathematical problem in which we want to MAXIMIZE or MINIMIZE a given function



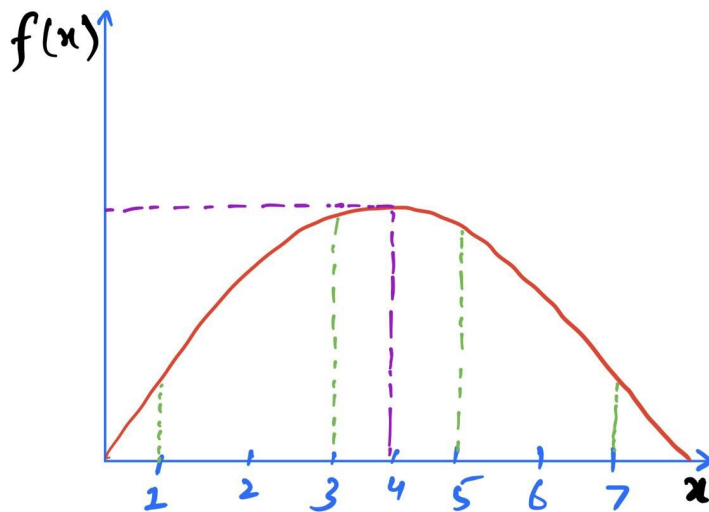
**Solution:**  $x = 4$

But what if we are told that  $x$  must be odd?



# Optimization Problem (2)

- Mathematical problem in which we want to MAXIMIZE or MINIMIZE a given function



But what if we are told that  $x$  must be odd?

**Solution:**  $x = 3$

# Constrained Optimization Problems

- Optimization problems where a function  $f(x)$  has to be maximized or minimized subject to (s.t.) some constraint(s)  $\phi(x)$

$$\min f(x)$$

$$\text{s.t. } \phi(x)$$

$$\max f(x)$$

$$\text{s.t. } \phi(x)$$

# Solving Constrained Optimization Problems

- Such optimization problems are solved using **Lagrange Multiplier Method**
- In particular, we take our objective function and the constraint(s) and do the following
  - We make a new objective function
  - This new objective function contains both the original objective and additional term(s)
  - The additional term(s) represent(s) our constraint(s)

# Solving Constrained Optimization Problems (2)

In particular, we take our objective function and the constraint(s) and do the following

- We make a new objective function
- This new objective function contains both the original objective and additional term(s)
- The additional term(s) represent(s) our constraint(s)

$$\begin{array}{ccc} \underset{w}{\operatorname{argmin}} f(w) & \longrightarrow & \underset{w}{\operatorname{argmin}} f(w) - \alpha(g(w) - a) \\ \text{subject to } g(w) < a & & \text{subject to } \alpha > 0 \end{array}$$

$\alpha$  are called Lagrange Multipliers

This is all the detail that you need to know about them for this course

# Alternate Forms of Regularization Objectives

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2 + \lambda \sum_{j=1}^p w_j^2$$

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2 + \lambda \sum_{j=1}^p |w_j|$$

$$\min_{\mathbf{w}} \left\{ \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2 \right\} \text{ subject to } \sum_{j=1}^p w_j^2 \leq s$$

$$\min_{\mathbf{w}} \left\{ \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{w}^T \cdot \mathbf{x}_i)^2 \right\} \text{ subject to } \sum_{j=1}^p |w_j| \leq s$$

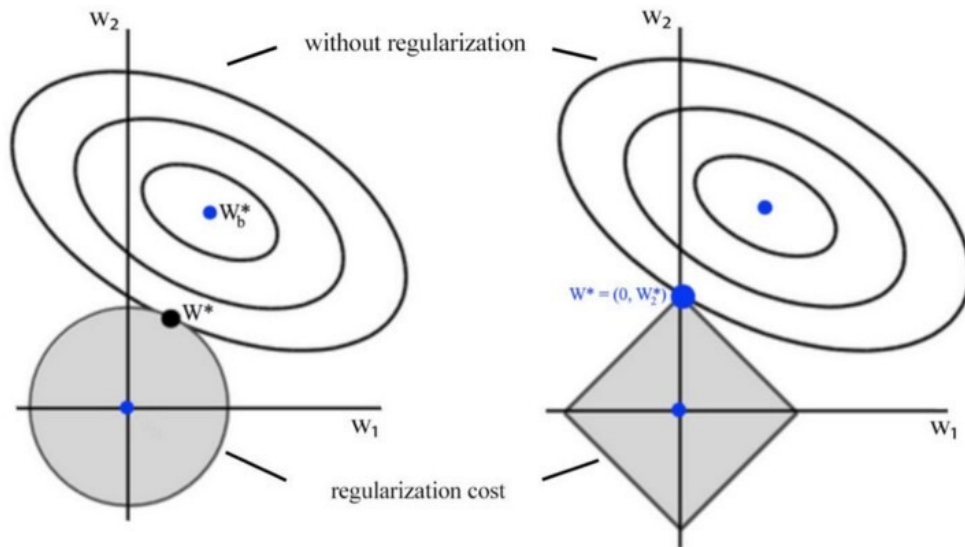
Thus, these are constrained optimization problems

# Constraints in a Two Dimensional Space

$$w_1^2 + w_2^2 \leq s$$

$$|w_1| + |w_2| \leq s$$

# Optimization Problem in a Two-Dimensional Space



L2 regularization promotes small parameters

L1 regularization promotes sparse parameters

# High Dimensional Data



# Curse of Dimensionality (1)

- Example of high dimensional data
  - Subjects: 3192
  - Measurements: 500,000

nature.com > nature > letters > article


MENU ▾

**nature**  
International journal of science

 Altmetric: 247 Citations: 607 [More detail >>](#)

Letter

## Genes mirror geography within Europe

John Novembre , Toby Johnson, Katarzyna Bryc, Zoltán Kutalik, Adam R. Boyko, Adam Auton, Amit Indap, Karen S. King, Sven Bergmann, Matthew R. Nelson, Matthew Stephens & Carlos D. Bustamante

*Nature* **456**, 98–101 (06 November 2008)  
doi:10.1038/nature07331  
[Download Citation](#)

Received: 30 May 2008  
Accepted: 12 August 2008  
Published online: 31 August 2008  
[Addendum: 13 November 2008](#)

# Curse of Dimensionality (2)

- High Dimensional Data is
  - **Difficult** to visualize
  - **Difficult** to analyze
  - **Difficult** to understand – to get insight from the data (correlation and predictions)

# The General Problem

- We have a dataset of  $n$  data points, where each point is  $p$ -dimensional

$$X = \{(\mathbf{x}_i) | \mathbf{x}_i \in \mathbb{R}^p\}_{i=1}^n$$

- The number of parameters in a machine learning model usually depends on the parameter  $p$
- Thus if  $p$  is very large, it can make parameter estimation challenging
- It can also make visualization of the data very hard

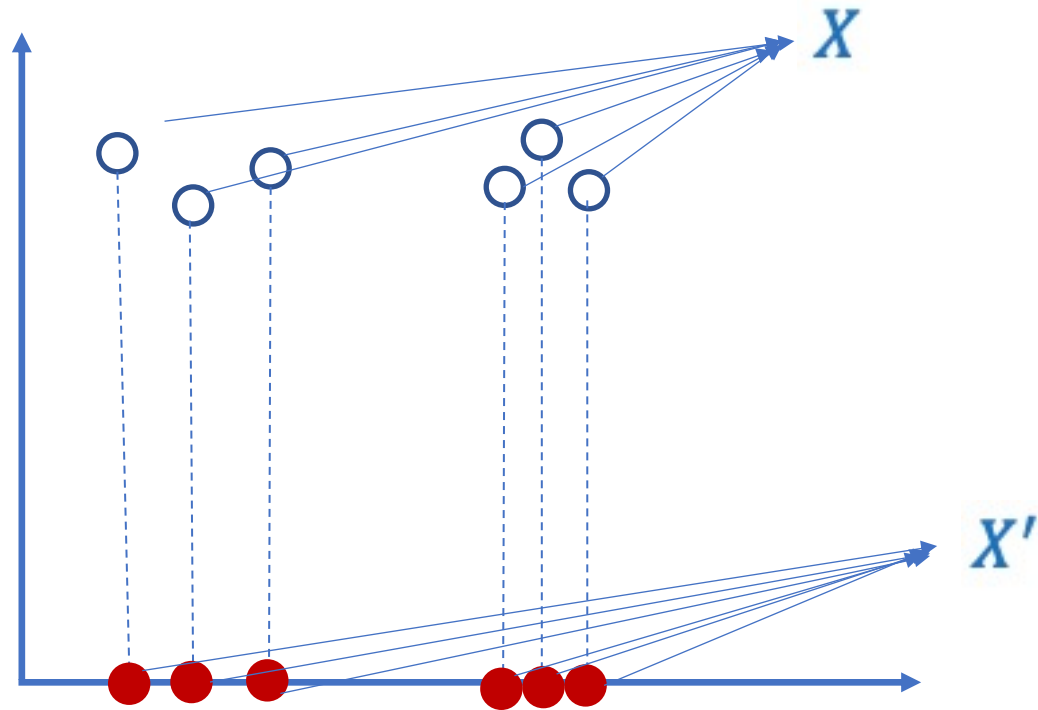
# Solution

- To solve the problem, we usually transform every  $p$ -dimensional point  $\mathbf{x}_i$  to a new  $d$ -dimensional point  $\mathbf{x}'_i$
- Such that  $d < p$

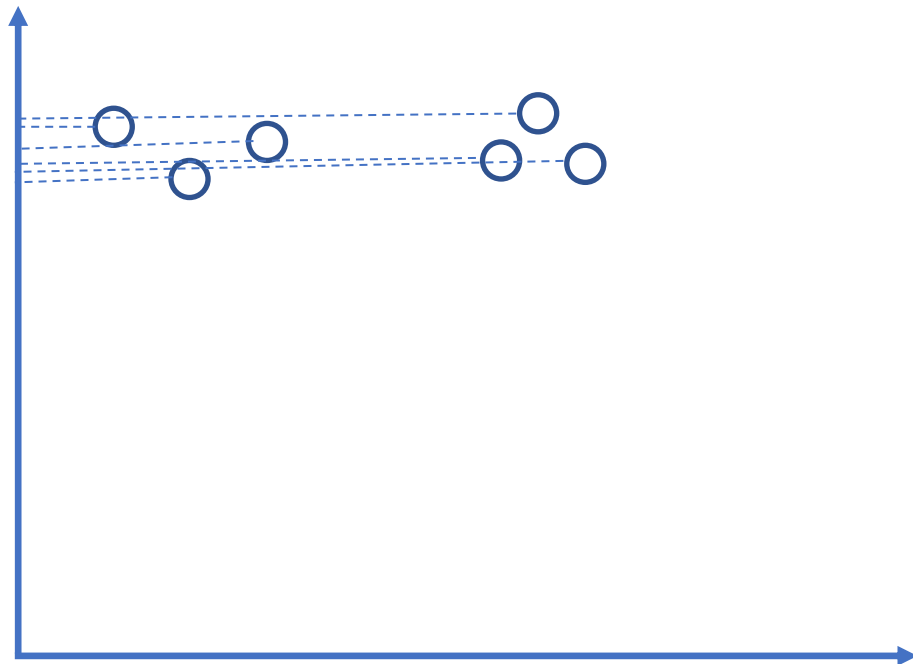
$$X' = \{(\mathbf{x}'_i) | \mathbf{x}'_i \in \mathbb{R}^d\}_{i=1}^n$$

- This process is called **projection**

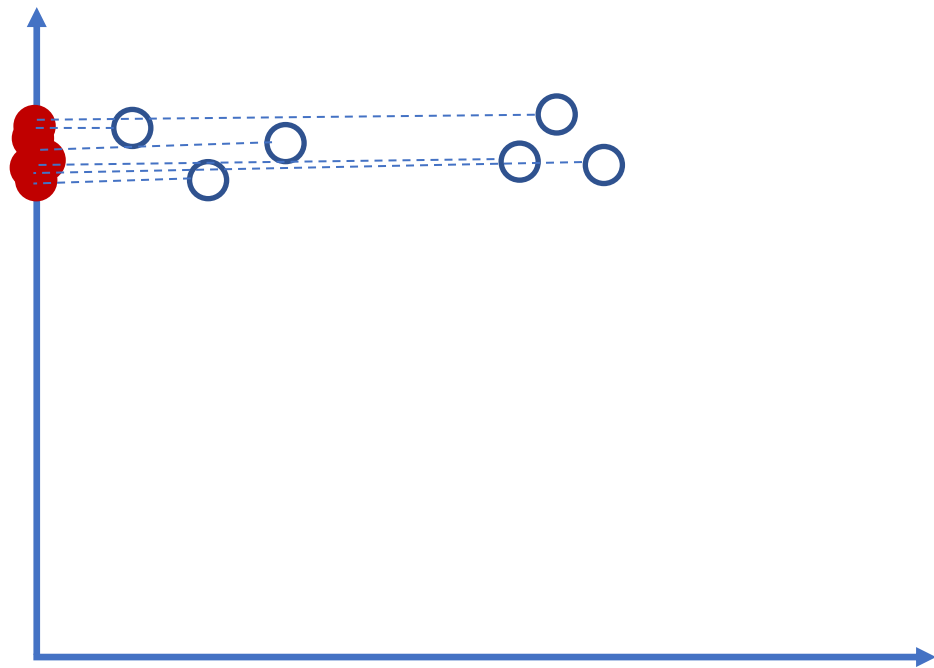
# Data Projection



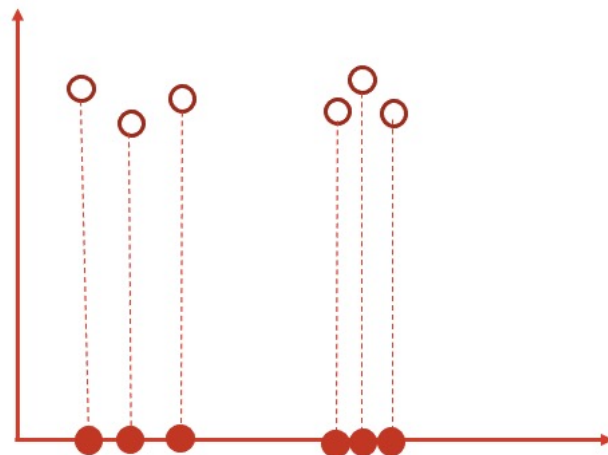
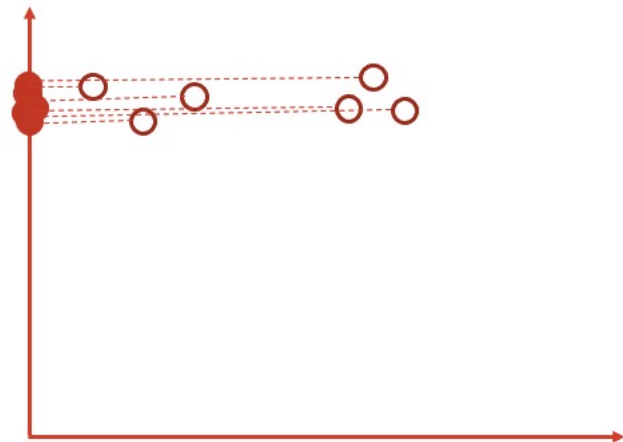
# Data Projection (2)



# Data Projection (3)



# Data Projection (4)



Which one should we choose between these two?



# Thus

- When projecting data to a lower dimensional space, we would like to retain as much of the structural information about our data as possible
- And this is where **variance** can help us
- We can compute variance in each one-dimensional space as

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x'_i - \mu_{x'})^2$$

$$\mu_{x'} = \frac{1}{n} \sum_{i=1}^n x'_i$$

- Which we will try maximize when deciding on our projecting directions.

# Principal Component Analysis (PCA)

# PCA

- One of the most widely used technique for projecting data into lower dimensions

# PCA (2)

- When projecting data from  $p$ -dimensional to a  $d$ -dimensional space, PCA defines  $d$  vectors, each represented as  $\mathbf{w}_j$  where  $j = 1, \dots, d$
- Each vector is  $p$ -dimensional – that is,  $\mathbf{w} \in \mathbb{R}^p$
- The  $i$  –  $th$  projected point is represented as  $\mathbf{x}'_i = [x'_{i1}, x'_{i2}, \dots, x'_{id}]^T$

$$x'_{id} = \mathbf{w}_d^T \mathbf{x}_i$$

# PCA (3)

- PCA uses **variance** in the projected space as its criteria to choose  $\mathbf{w}_d$
- In particular,  $\mathbf{w}_1$  will be the vector that will keep the variance in  $x'_{i1}$  as high as possible
- $\mathbf{w}_2$  Will be chosen to maximize the variance, too, but with an additional constraint
- $\mathbf{w}_2$  must be orthogonal to  $\mathbf{w}_1$

In general,

$$\mathbf{w}_i^T \mathbf{w}_j = 0, \quad \forall i \neq j$$

# PCA (4)

- In addition the previous constraint, the PCA also demands that

$$\mathbf{w}_d^T \mathbf{w}_d = 1$$

- Which means that each vector should have a length of 1

# PCA (5)

- Finally, PCA requires that each original dimension has zero mean

$$\mu_x = \frac{1}{n} \sum_{i=1}^n x_i = 0$$

# What do we know so far?

1. PCA reduces dimensionality by projecting data from a high dimensional space to a lower dimensional space
2. It does so by using a set of vectors
3. Vectors will be chosen such that they maximize the variance in the project space
4. Furthermore, the vectors
  - Should be orthogonal
  - Have unit length
5. Finally, data should have zero mean



# How Does PCA Work?

# How does PCA work?

- In order to understand how PCA works, let's start with projection into a 1-dimensional space, that is,  $d = 1$
- In this case, for each  $\mathbf{x}_i$  the result of projection of will be a scalar value

$$x'_i = \mathbf{w}^T \mathbf{x}_i$$

# How does PCA work? (2)

- The variance in the projected space is given by

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x'_i - \mu_{x'})^2$$

$$= \frac{1}{n} \sum_{i=1}^n (x'_i - 0)^2$$

$$= \frac{1}{n} \sum_{i=1}^n (x'_i)^2$$

$$\mu_{x'} = \frac{1}{n} \sum_{i=1}^n x'_i$$

$$= \frac{1}{n} \sum_{i=1}^n \mathbf{w}^T \mathbf{x}_i$$

$$= \mathbf{w}^T \left( \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \right) = \mathbf{w}^T \boldsymbol{\mu}_x$$

$$\mu_{x'} = 0$$

# How does PCA work? (3)

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x'_i)^2$$

- We also know that

$$x'_i = \mathbf{w}^T \mathbf{x}_i$$

- Substituting its value in the above equations gives us

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i)^2$$

# How does PCA work? (4)

$$\begin{aligned}\sigma^2 &= \frac{1}{n} \sum_{i=1}^n (\mathbf{w}^T \mathbf{x}_i)^2 &= \frac{1}{n} \sum_{i=1}^n \mathbf{w}^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{w} \\ & &= \mathbf{w}^T \left( \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T \right) \mathbf{w}\end{aligned}$$

- Where  $\mathbf{C}$  is the sample covariance matrix

$$\sigma^2 = \mathbf{w}^T \mathbf{C} \mathbf{w}$$

# Recall that the PCA Wants to

- Maximize the variance  $\sigma^2$
- And we just derived that

$$\sigma^2 = \mathbf{w}^T \mathbf{C} \mathbf{w}$$

- Therefore, the projection that maximized  $\sigma^2$  would also maximize  $\mathbf{C}$

# Maximizing $\sigma^2$ : Trivial Solution

- Increase the value of the elements in  $\mathbf{w}$
- And that is why, we already set the constraint

$$\mathbf{w}^T \mathbf{w} = 1$$

- Thus we are dealing with a constraint optimization problem

# PCA Objective

- Find  $\mathbf{w}$  that maximizes the following

$$\mathcal{L} = \mathbf{w}^T \mathbf{C} \mathbf{w} - \lambda(\mathbf{w}^T \mathbf{w} - 1)$$

- Where  $\lambda$  is the Lagrange multiplier



# Finding the Optimum $\mathbf{w}$

$$\mathcal{L} = \mathbf{w}^T \mathbf{C} \mathbf{w} - \lambda(\mathbf{w}^T \mathbf{w} - 1)$$

- Take partial derivative with respect to  $\mathbf{w}$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{w}} &= 2\mathbf{C} \mathbf{w} - \lambda(2\mathbf{w} - \mathbf{0}) \\ &= 2\mathbf{C} \mathbf{w} - \lambda(\mathbf{w}) \end{aligned}$$

- Setting it to 0, we get a very important result

$$\lambda \mathbf{w} = \mathbf{C} \mathbf{w}$$

# Let's Analyze What We Got

$$\lambda \mathbf{w} = \mathbf{C} \mathbf{w}$$

- $\lambda$  is a scalar
- $\mathbf{w}$  is a vector
- $\mathbf{C}$  is a matrix

1. Thus, multiplying  $\mathbf{w}$  with  $\mathbf{C}$  only scales it (only changes its length)
2. Thus,  $\mathbf{w}$  that maximizes the variance is one of the **eigenvectors** of  $\mathbf{C}$  and  $\lambda$  is the eigen value of  $\mathbf{w}$

# But $w$ is Which EigenVector of $C$ ?

- $C$  is an  $p \times p$  matrix
- Thus, it has  $p$  eigenvectors
- How do we know which one corresponds to highest variance in the projected space?

# But $\mathbf{w}$ is Which EigenVector of $\mathbf{C}$ ? (2)

- Our expression for variance  $\sigma^2$  is

$$\sigma^2 = \mathbf{w}^T \mathbf{C} \mathbf{w}$$

- We also know that  $\mathbf{w}^T \mathbf{w} = 1$

- Thus we can write the equation of  $\sigma^2$  as

$$\sigma^2 \mathbf{w}^T \mathbf{w} = \mathbf{w}^T \mathbf{C} \mathbf{w}$$

But  $\mathbf{w}$  is Which EigenVector of  $\mathbf{C}$ ? (3)

$$\sigma^2 \mathbf{w}^T \mathbf{w} = \mathbf{w}^T \mathbf{C} \mathbf{w}$$

- Removing  $\mathbf{w}^T$  from both sides, we get

$$\sigma^2 \mathbf{w} = \mathbf{C} \mathbf{w}$$

- Note that we just showed that  $\lambda \mathbf{w} = \mathbf{C} \mathbf{w}$

- Thus

$$\sigma^2 \mathbf{w} = \lambda \mathbf{w} = \mathbf{C} \mathbf{w}$$

# Takeaway

$$\sigma^2 \mathbf{w} = \lambda \mathbf{w} = C \mathbf{w}$$

- Given a (eigenvector  $\mathbf{w}$ , eigenvalue  $\lambda$ ) pair of  $C$ ,  $\lambda$  corresponds to the amount of variance in the projected space defined by  $\mathbf{w}$
- Thus if we found  $p$  (eigenvector, eigenvalue) pairs of  $C$ , the pair with the highest eigenvalue corresponds to the vector that would maximize the variance in the projected space the most

Thus, Given  $X = \{(x_i) | x_i \in \mathbb{R}^p\}_{i=1}^n$ ,  
PCA Works as Follows

1. Transform the data to have zero mean by subtracting  $\mu_x$  from each point
2. Compute the sample covariance matrix  $C$
3. Find  $p$  (eigenvector, eigenvalue) pairs of  $C$
4. Find the eigenvectors corresponding to  $d$  highest eigenvalues  $w_1, w_2, \dots, w_d$
5. Compute  $X'$  as  $X' = XW$ , where  $W = [w_1, w_2, \dots, w_d]$

# Recommended Reading

1. Section 7.2, *A First Course in Machine Learning*, by Simon Rogers and Mark Girolami
2. Section 6.2 from *Introduction to Statistical Learning* by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani



# Summary

- Hyperparameters
- Cross-validation
- Regularization
- Constrained Optimization Problems
- L1 vs L2 as constrained optimization problems
- High Dimensional Data and Their Issues
- Principal Component Analysis

# Reflection

1. In what ways does the constraint imposed by Lasso regression lead to sparse models, and why might sparsity be desirable in certain machine learning problems?
2. Considering the curse of dimensionality, how does PCA help in mitigating its effects, and what are the limitations of PCA in preserving the original data structure?
3. How does the choice of the number of principal components in PCA affect the balance between dimensionality reduction and the retention of meaningful variance in the data? What criteria can be used to make this decision?
4. Reflect on the mathematical underpinnings of PCA, specifically the role of eigenvalues and eigenvectors in defining the principal components. How does this relate to the variance explained by each component?
5. Discuss the practical implications of using regularization and dimensionality reduction techniques in real-world data sets. How do these techniques influence the model development process, from feature selection to model validation?