Dimensionality Reduction and Nearest Neighbor Methods

Machine Learning Decal

Hosted by Machine Learning at Berkeley

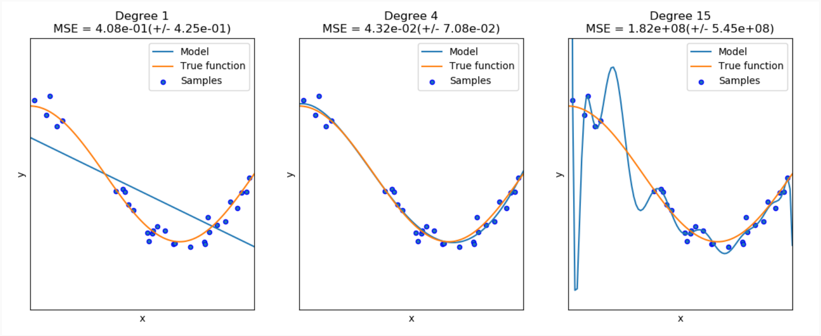
Overview

Agenda

Motivation

Previously ...

We have seen how adding features can lead to overfitting.



Overfitting

“A conspiracy theory of data.”

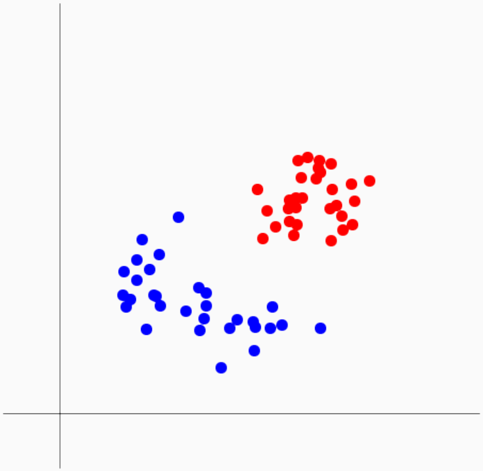
Benefits of Dimensionality Reduction

- Reduce variance of the model – Less room for noise to enter the data.
- . . .

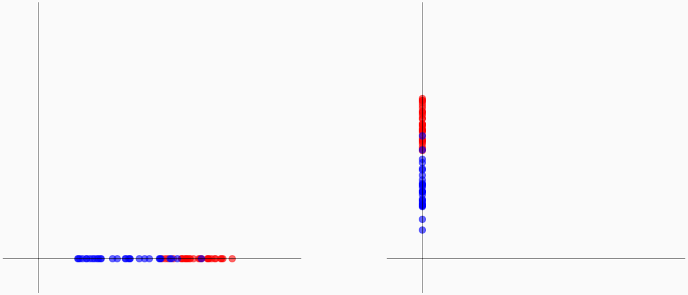
Benefits of Dimensionality Reduction

- Reduce variance of the model.
- Reduce model complexity and training time.
- Find only the most relevant features.
- Data compression.
- Easier to visualize what the model is doing.

Naive Solution: Remove Features

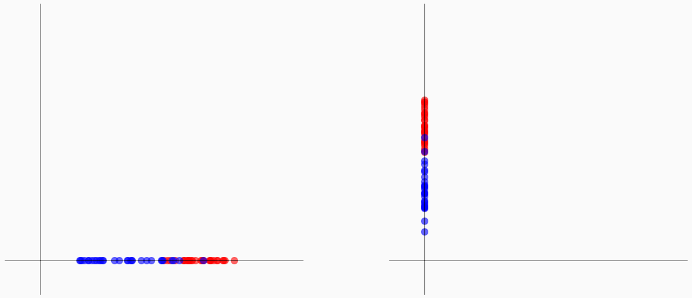


Naive Solution: Remove Features



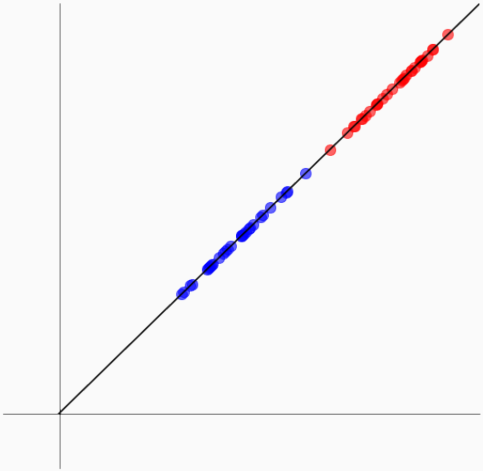
Issue: ???

Naive Solution: Remove Features



Issue: We're losing information!

Informed Solution: Project onto Subspaces



PCA Motivation

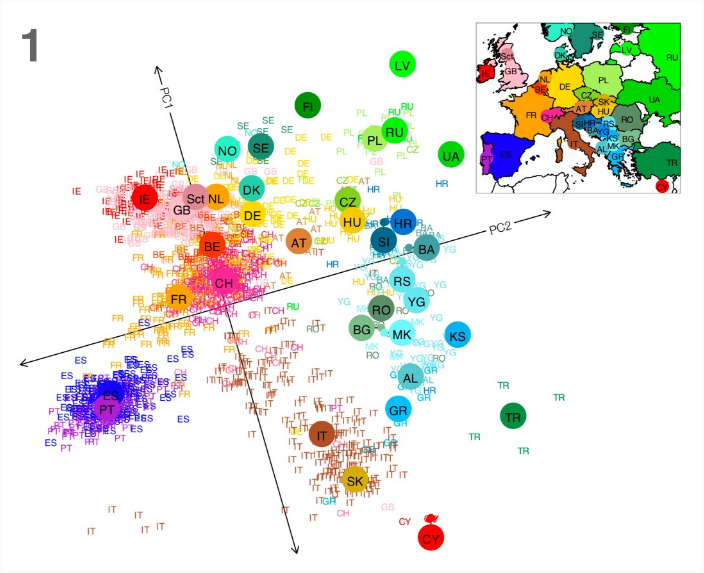
Find the ‘best’ projection.

PCA Motivation

Find the 'best' projection.

The 'best' projection is the one that maximizes the spread, or variance, of the projected data.

PCA Motivation



Deriving PCA

Math Review: Matrices as Maps

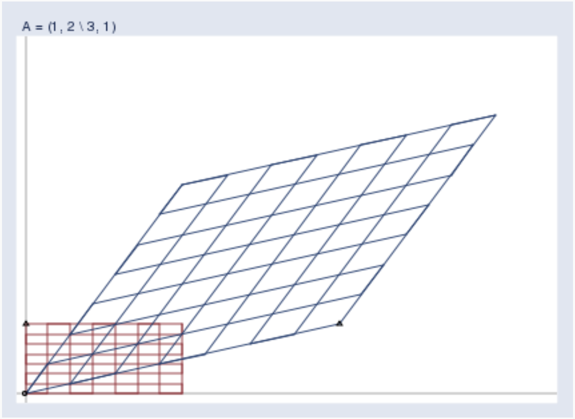
We typically view matrices as data storage containers. They can also be viewed as maps from vectors to vectors.

$$A = \begin{bmatrix} 1 & 2 \\ -2 & 3 \end{bmatrix}, v = \begin{bmatrix} -1 \\ 4 \end{bmatrix}$$

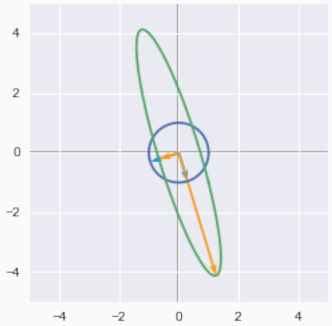
$$Av = \begin{bmatrix} 1 & 2 \\ -2 & 3 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 4 \end{bmatrix} = \begin{bmatrix} 7 \\ 14 \end{bmatrix}$$

Math Review: Matrices as Maps

When taken over every possible input vector, matrices squash and stretch the input space.



Math Review: Eigenvalues and Eigenvectors



- Every vector is affected after a matrix is applied, but certain vectors retain their direction, while only changing magnitude. These are the **eigenvectors**.
- For each eigenvector, the corresponding **eigenvalue** is the factor by which the eigenvector is stretched or shrunk.

Math Review: Eigenvalues and Eigenvectors

In math,

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}.$$

Here, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a $n \times n$ matrix, $\mathbf{v} \in \mathbb{R}^{n}$, and $\lambda \in \mathbb{R}$.

PCA Derivation: The First Principle Component

Let $\mathbf{X}$ be our data matrix of n samples, each of which is d-dimensional, so $\mathbf{X} \in \mathbb{R}^{n \times d}$.

Recall: We seek to find the vector $\mathbf{v} \in \mathbb{R}^{n}$ such that the variance of the projection of the rows of $\mathbf{X}$ onto $\mathbf{v}$ is maximized.

PCA Derivation: The First Principle Component

First, our math will be cleaner if we center our data, as variance is always defined relative to the mean. Define $\overline{\mathbf{X}} = \mathbf{X} - \mu(\mathbf{X})$, where each row of $\mu(\mathbf{X}) \in \mathbb{R}^{n \times d}$ is equal to the average of the rows of $\mathbf{X}$.

If we constrain $\mathbf{v}$ to have norm one, then $\overline{\mathbf{X}}\mathbf{v} \in \mathbb{R}^n$ contains the projections of the rows of $\overline{\mathbf{X}}$ onto $\mathbf{v}$. The variance we seek to maximize is thus

$$\max_{||\mathbf{v}||=1} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{v})^2 = \max_{||\mathbf{v}||=1} \frac{1}{n} \cdot (\overline{\mathbf{X}}\mathbf{v})^\top (\mathbf{X}\mathbf{v}) = \max_{\mathbf{v}^\top \mathbf{v}=1} \frac{1}{n} \cdot \mathbf{v}^\top \overline{\mathbf{X}}^\top \overline{\mathbf{X}}\mathbf{v}.$$

PCA Derivation: The First Principle Component

To solve this problem, we will use the method of Lagrange Multipliers:

$$\nabla_{\mathbf{v}} \left(\mathbf{v}^\top \overline{\mathbf{X}}^\top \overline{\mathbf{X}} \mathbf{v} \right) = \lambda \nabla_{\mathbf{v}} \left(\mathbf{v}^\top \mathbf{v} \right)$$

Taking the derivatives, we have that $2 \cdot \overline{\mathbf{X}}^\top \overline{\mathbf{X}} \mathbf{v} = 2 \cdot \lambda \mathbf{v}$ Thus, $\mathbf{v}$ is an eigenvector of $\overline{\mathbf{X}}^\top \overline{\mathbf{X}}$!

PCA Derivation: The First Principle Component

Substituting this observation into our original objective function, we have that

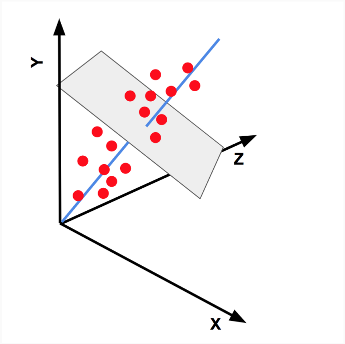
$$\max_{\mathbf{v}^\top \mathbf{v}=1} \frac{1}{n} \cdot \mathbf{v}^\top \overline{\mathbf{X}}^\top \overline{\mathbf{X}} \mathbf{v} = \max_{\mathbf{v}^\top \mathbf{v}=1} \frac{1}{n} \cdot \mathbf{v}^\top (\lambda \mathbf{v}) = \frac{\lambda}{n}$$

This tells us that we should project onto the eigenvector of $\overline{\mathbf{X}}^\top \overline{\mathbf{X}}$ with the largest eigenvalue.

PCA Derivation: More Principle Components

We have found the direction that explains the most variance of our data. How do we find more directions?

Solution: For each of the points, remove the portion corresponding to the direction that we found. Then, we can repeat the process.



PCA Derivation: More Principle Components

If we perform a similar mathematical analysis, we will find that the k-th principle component is equal to the eigenvector of $\overline{\mathbf{X}}^\top \overline{\mathbf{X}}$ with the k-th largest eigenvalue.

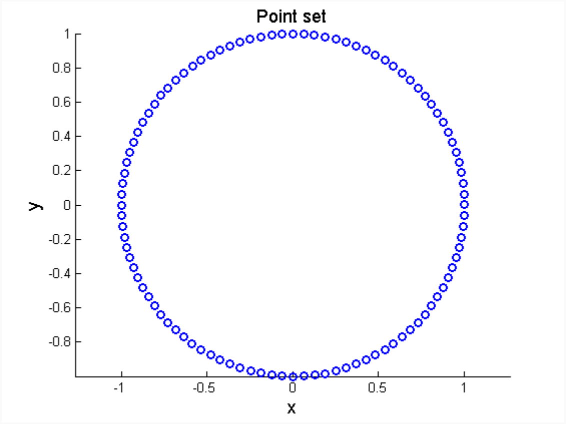
This allows us to choose the level of compression.

PCA Demo

Autoencoders

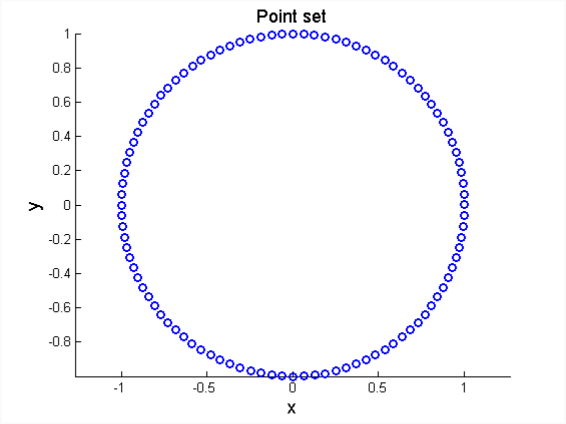
Motivation

What's the problem with applying PCA to this data?



Motivation

A potential issue: Certain datasets are clearly one-dimensional, yet PCA will fail!

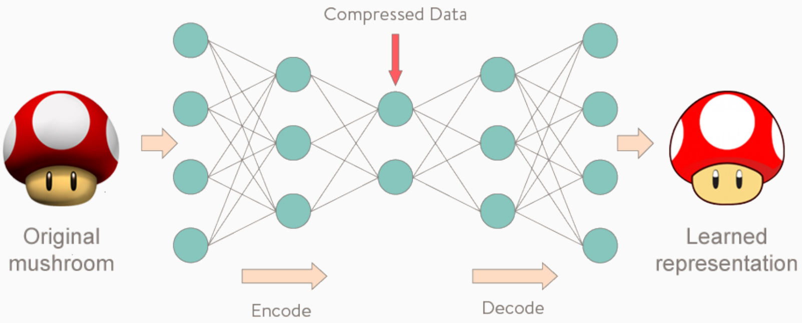


The Solution: Neural Networks

Luckily, neural networks can learn arbitrary, non-linear functions.

The Autoencoder Setup

How do we use neural networks for data compression? Learn the identity function with a bottleneck!



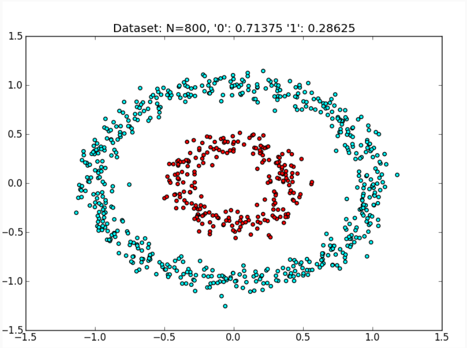
Autoencoder Demo

Kernels

Motivation

We mentioned that dimensionality reduction is useful when we have too many features (too high a variance).

If we have too high a *bias*, we may need to instead increase the number of features.



Kernels with Linear Regression

Recall linear regression:

$$\arg\min_{\mathbf{w}} ||\mathbf{X}\mathbf{w} - \mathbf{y}||^2 = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$$

We previously saw that we can add polynomial features to our data by replacing $\mathbf{X} \in \mathbb{R}^{n \times d}$ with $\mathbf{K} \in \mathbb{R}^{n \times d'}$, where the rows of $\mathbf{K}$ contain the d' features generated from the rows of $\mathbf{X}$. We call this data augmentation process **kernelization**.

Kernels with Linear Regression

For example, if our original data matrix is

$$\mathbf{X} = \begin{bmatrix} 4 & 3 \\ 2 & 5 \end{bmatrix},$$

then our kernel matrix with quadratic features is

$$\mathbf{K} = \begin{bmatrix} \phi([4,3])^\top \\ \phi([2,5])^\top \end{bmatrix} = \begin{bmatrix} 1 & 4 & 3 & 4 \cdot 3 & 4^2 & 3^2 \\ 1 & 2 & 5 & 2 \cdot 5 & 2^2 & 5^2 \end{bmatrix}.$$

Kernels with Linear Regression

When we make this substitution, our linear regression prediction for a new point $\mathbf{z}$ is equal to

$$\begin{aligned}\phi(\mathbf{z})^\top \mathbf{w} &= \phi(\mathbf{z})^\top \left(\mathbf{K}^\top \mathbf{K}\right)^{-1} \mathbf{K}^\top \mathbf{y} \\ &= \phi(\mathbf{z})^\top \mathbf{K}^\top \left(\mathbf{K}\mathbf{K}^\top\right)^{-1} \mathbf{y}.\end{aligned}$$

Kernels with Linear Regression

So, our goal is to find $\phi(\mathbf{z})^\top \mathbf{K}^\top \left(\mathbf{K}\mathbf{K}^\top\right)^{-1} \mathbf{y}$. Let's first find $\mathbf{K}\mathbf{K}^\top$.

We see that

$$\begin{aligned}(\mathbf{K}\mathbf{K}^\top)_{ij} &= \left(\begin{bmatrix} \phi(\mathbf{x}_1)^\top \\ \vdots \\ \phi(\mathbf{x}_n)^\top \end{bmatrix} \cdot \begin{bmatrix} \phi(\mathbf{x}_i) & \cdots & \phi(\mathbf{x}_n) \end{bmatrix} \right)_{ij} \\ &= \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle .\end{aligned}$$

This is the Kernel Trick!

Kernels with Linear Regression

Example of the kernel trick: Consider the kernel given by

$$\phi\left([x, y]^\top\right) = \left[x^2,\, \sqrt{2} \cdot xy,\, y^2\right]^\top .$$

We can rewrite the desired inner product as:

$$\begin{aligned}
&\left\langle \phi\left([x_1, y_1]^\top\right), \phi\left([x_2, y_2]^\top\right) \right\rangle \\
&= \left\langle \left[x_1^2,\, \sqrt{2} \cdot x_1 y_1,\, y_1^2\right]^\top, \left[x_2^2,\, \sqrt{2} \cdot x_2 y_2,\, y_2^2\right]^\top \right\rangle \\
&= x_1^2 x_2^2 + 2 x_1 y_1 x_2 y_2 + y_1^2 y_2^2 \\
&= (x_1 x_2 + y_1 y_2)^2 \\
&= \left\langle [x_1, y_1]^\top, [x_2, y_2]^\top \right\rangle^2 .
\end{aligned}$$

This is much easier to compute, especially for higher dimensions.

Kernels with Linear Regression

Key Takeaway

Kernels allow for complex, expressive models, without too much additional computation.

SVMs

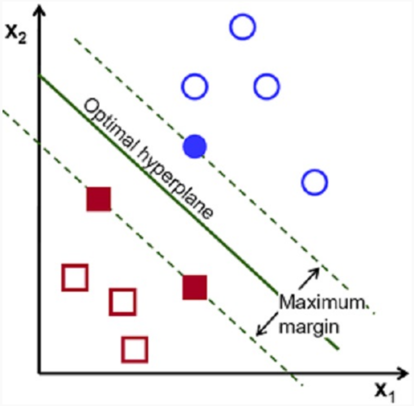
Introduction

We have seen how kernels can be applied to linear regression.

We now study a classification method that can utilize kernelization, **support vector machines**.

Support Vector Machines

SVMs are a *classification* tool.



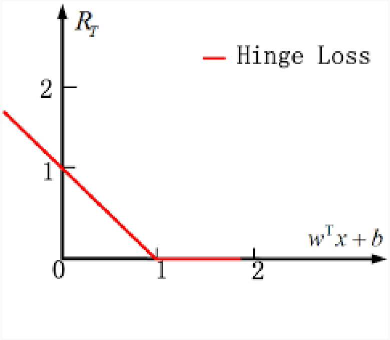
Support Vector Machines

How do we represent this idea of a margin using a loss function?

$$\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \sum_{i=1}^{n} \max(0, 1 - y_i \cdot (\mathbf{w}^\top \mathbf{x}_i + b))$$

Support Vector Machines

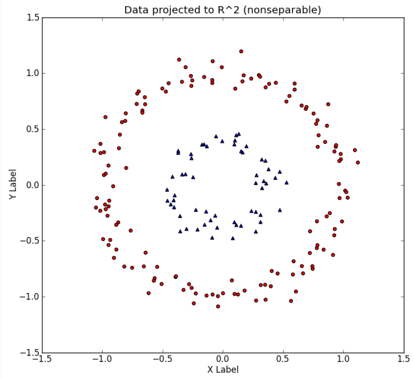
This loss is called the *hinge loss*.

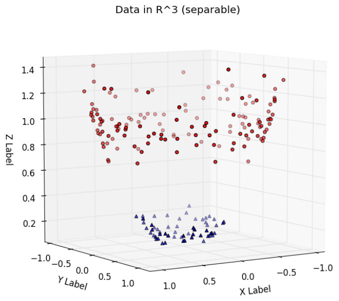


$$f(x) = \min(0, 1 - x)$$

Support Vector Machines

SVMs heavily utilize kernels!





Support Vector Machines

Note that, due to our objective function, most of our data does not contribute to the overall loss. This means that we can randomly throw away data points and still get a very accurate model!