SOME TYPES OF MACHINE LEARNING

Supervised learning – have inputs and corresponding targets

(May want to make predictions with new input data, or just understand relationship!)

- Regression: continuous target variables
- Classification: target variable is discrete category

Unsupervised learning – do not have target values

Typically want to discover structure in the data. Examples goals:

- Clustering
- Dimensionality reduction
- Density estimation

Reinforcement learning

PCA (PRINCIPAL COMPONENT ANALYSIS)

Data is high dimensional

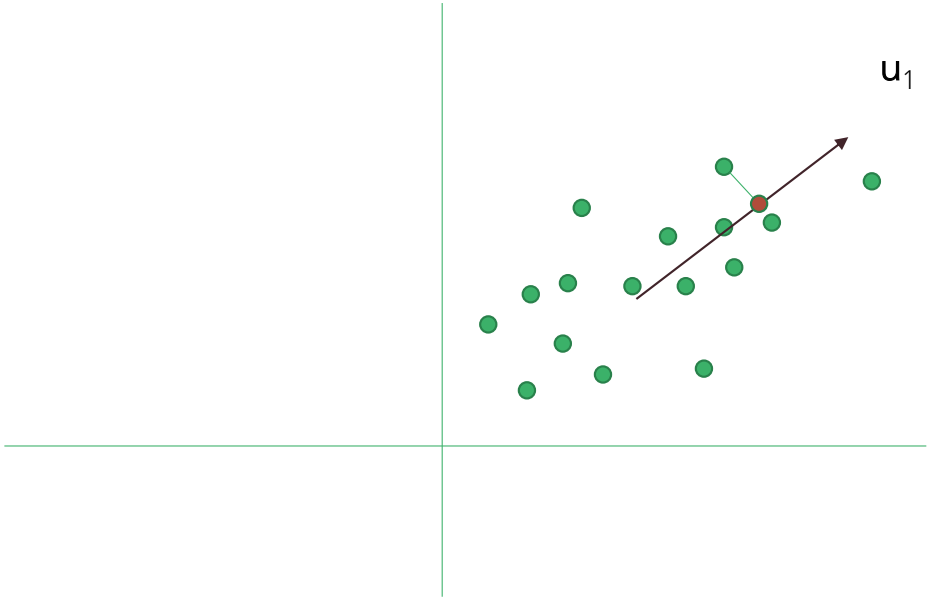
Want to project to a lower dimensional space but still keep relevant information

$$\{\mathbf{x}_n\} \in \mathbb{R}^D \rightarrow \{\tilde{\mathbf{x}}_n\} \in \mathbb{R}^M$$

M<D

PCA (PRINCIPAL COMPONENT ANALYSIS)

First, find u_1 (vector of length1) to minimize variance of the projection onto u_1

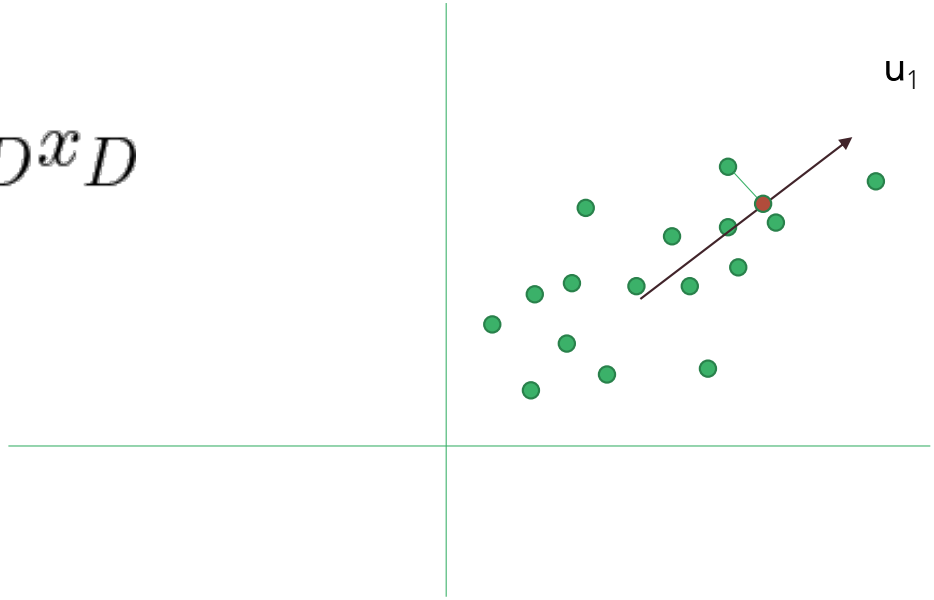


PCA (PRINCIPAL COMPONENT ANALYSIS)

First, find $\mathbf{u}_1$ (vector of length1) to minimize variance of the projection onto $\mathbf{u}_1$

Projection of $\mathbf{x}_n$ onto $\mathbf{u}_1$:

$$\mathbf{u}_1^T \mathbf{x}_n = u_1 x_1 + ... u_D x_D$$

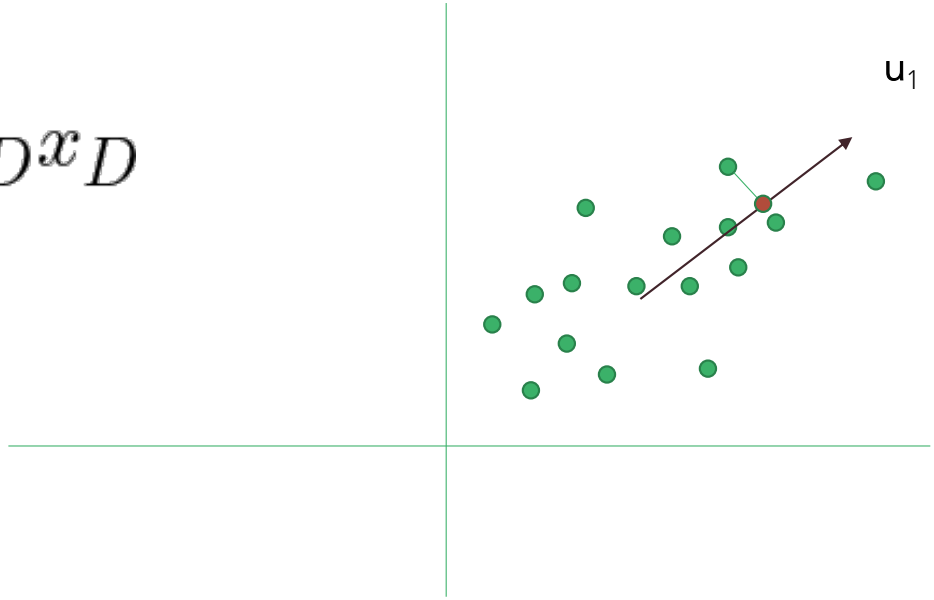


PCA (PRINCIPAL COMPONENT ANALYSIS)

First, find $\mathbf{u}_1$ (vector of length1) to minimize variance of the projection onto $\mathbf{u}_1$

Projection of $\mathbf{x}_n$ onto $\mathbf{u}_1$:

$$\mathbf{u}_1^T \mathbf{x}_n = u_1 x_1 + ... u_D x_D$$



PCA (PRINCIPAL COMPONENT ANALYSIS)

First, find u_1 (vector of length1) to minimize variance of the projection onto u_1

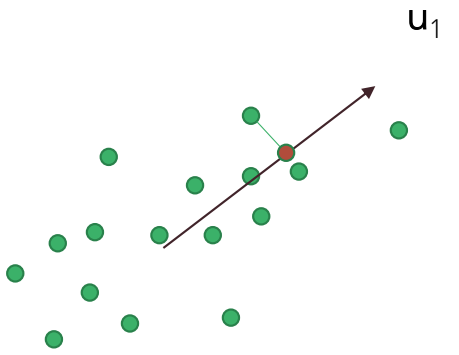
Projection of x_n onto u_1:

$$\mathbf{u}_1^T \mathbf{x}_n = u_1 x_1 + \ldots u_D x_D$$

variance of projection: $\frac{1}{N}\sum_{n=1}^{N}(\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\bar{\mathbf{x}})^2 = \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1$

where $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$, $\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$

data covariance



PCA (PRINCIPAL COMPONENT ANALYSIS)

First, find u_1 to minimize variance of the projection onto u_1

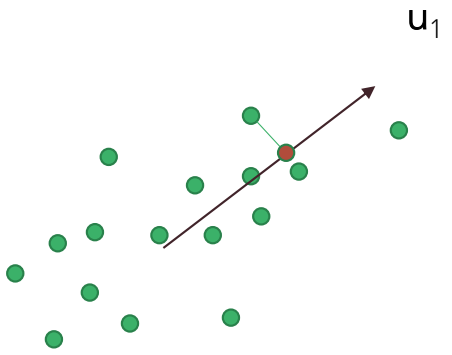
variance of projection: $\frac{1}{N}\sum_{n=1}^{N}(\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\bar{\mathbf{x}})^2 = \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1$

where $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$, $\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$

maximizing subject to $||\mathbf{u}_1|| = 1$, we get:

$\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1$

$\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 = \lambda_1$



PCA (PRINCIPAL COMPONENT ANALYSIS)

First, find u_1 to minimize variance of the projection onto u_1

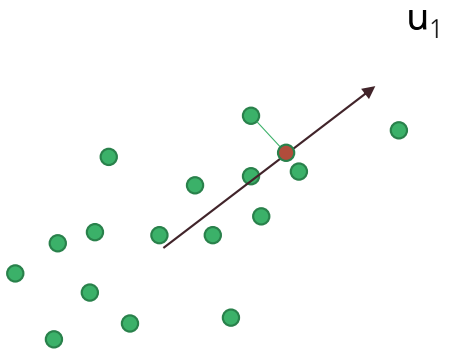
$$\text{variance of projection: } \frac{1}{N}\sum_{n=1}^{N}(\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\bar{\mathbf{x}})^2 = \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1$$

$$\text{where } \mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T,\ \bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

$$\text{maximizing subject to } ||\mathbf{u}_1|| = 1, \text{ we get:}$$

$$\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1$$

$$\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 = \lambda_1$$



So choose u_1 to be unit-length eigenvector corresponding to largest eigenvalue of S

PCA (PRINCIPAL COMPONENT ANALYSIS)

Can imagine continuing iteratively, e.g. next choosing $\mathbf{u}_2$ to minimize the covariance of the projections of the $\mathbf{x}_n - \mathbf{u}_1^T\mathbf{x}_n$ onto $\mathbf{u}_2$

Stop at desired number of components M – can use variance explained to decide

PROBABALISTIC PCA

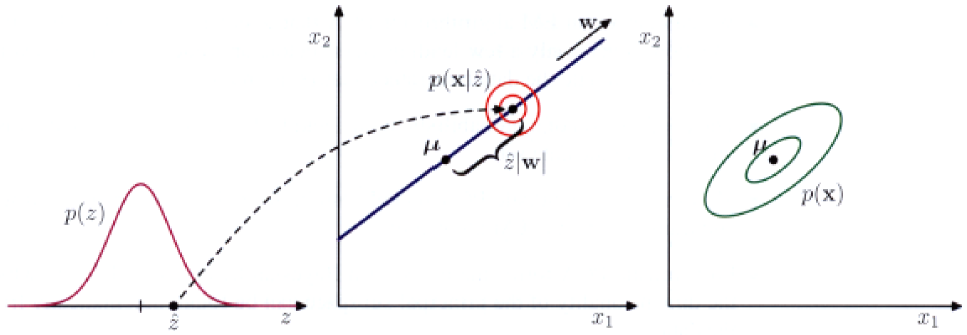


Figure 12.9 from Pattern Recognition and Machine Learning, by Christopher Bishop

PROBABALISTIC PCA

$$p(\mathbf{z}) \sim N(0, \mathbf{I})$$
$$p(\mathbf{x}|\mathbf{z}) = N(\mathbf{Wz} + \mathbf{mu}, \sigma^2\mathbf{I})$$
$$\implies p(\mathbf{x}) = N(\mathbf{x}, \mu, \mathbf{WW^T} + \sigma^2\mathbf{I}),$$
$$\text{can compute } p(\mathbf{z}|\mathbf{x})$$

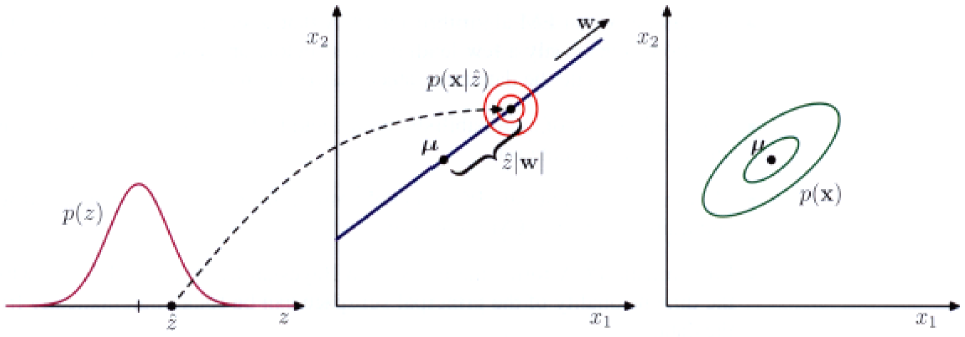


Figure 12.9 from Pattern Recognition and Machine Learning, by Christopher Bishop

z are latent variables

PROBABALISTIC PCA

Can also put a prior over W!

$$p(\mathbf{z}) \sim N(0, \mathbf{I})$$
$$p(\mathbf{x}|\mathbf{z}) = N(\mathbf{Wz} + \mathbf{mu}, \sigma^2\mathbf{I})$$
$$\implies p(\mathbf{x}) = N(\mathbf{x}, \mu, \mathbf{WW^T} + \sigma^2\mathbf{I}),$$
$$\text{can compute } p(\mathbf{z}|\mathbf{x})$$

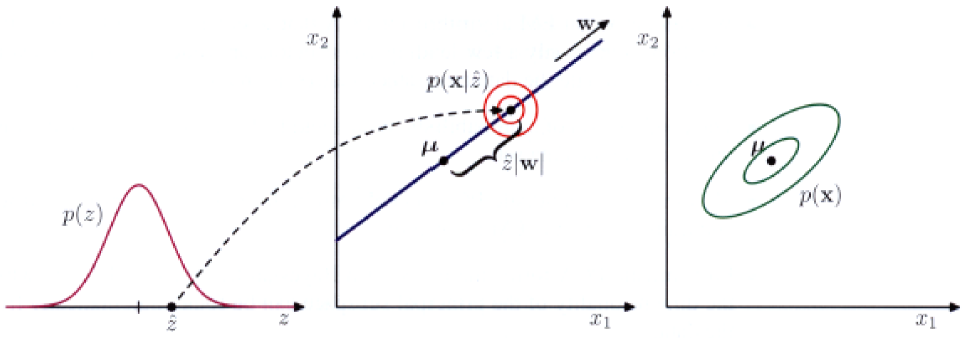


Figure 12.9 from Pattern Recognition and Machine Learning, by Christopher Bishop

z are latent variables

T-DISTRIBUTED STOCHASTIC NEIGHBOR EMBEDDING (T-SNE)

Again want to represent high dimensional vectors x with low dimensional counterparts y

Aim to transform close points into points that are still close

T-DISTRIBUTED STOCHASTIC NEIGHBOR EMBEDDING (T-SNE)

Again want to represent high dimensional vectors x with low dimensional counterparts y

Aim to transform close points into points that are still close

$$p_{j|i} = \frac{\exp(-||\mathbf{x}_i - \mathbf{x}_j||/\sigma_i^2)}{\sum_{k \neq i} \exp(-||\mathbf{x}_i - \mathbf{x}_k||/\sigma_i^2)} \qquad q_{ij} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||\mathbf{y}_i - \mathbf{y}_k||^2)^{-1}}$$

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

$$\text{minimize } KL(p,q) = \int p_{ij} \log(\frac{p_{ij}}{q_{ij}}) = \int p_{ij}(\log(p_{ij}) - \log(q_{ij}))$$

KL DIVERGENCE

$$\text{minimize } KL(p,q) = \int p_{ij} \log(\frac{p_{ij}}{q_{ij}}) = \int p_{ij}(\log(p_{ij}) - \log(q_{ij}))$$

A way to measure the "distance" two probability distributions, but caution….

Not symmetric! KL(p, q) and KL(q, p) are different in general

An intro blog post on KL divergence: https://karinknudson.com/kl_divergence.html

Useful in variational inference, among other settings

FOR AN EXCELLENT DISCUSSION OF T-SNE, SEE:

For an excellent discussion of t-SNE, see:

https://distill.pub/2016/misread-tsne/

Check out alternative manifold learning techniques! For a brief introduction to those that are implemented in sci-kit learn, see: https://scikit-learn.org/stable/modules/manifold.html#manifol

For a discussion and comparison of t-SNE and PCA (with examples in Python), see e.g.:

https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1

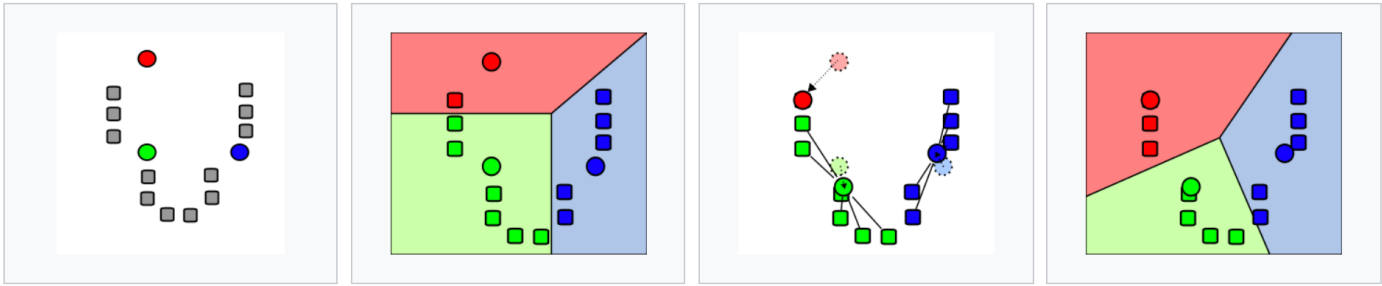
K-MEANS CLUSTERING

Choose the number of clusters you expect: K

Choose a set of K points ("means") then alternately

1. "Assign" each data point to the cluster corresponding to the nearest "mean"

2. Recalculate the means by taking the average of the points assigned to each cluster



MIXTURE OF GAUSSIANS CLUSTERING

Assume data points are generated from a mixture of K Gaussian distributions

Find the mean and variance of those Gaussians that give the highest likelihood

(Can use EM algorithm to optimize the likelihood)