SUPERVISED LEARNING - (Brief)

- Supervised learning; basics; labeled data
- Classification problems; KNN classification
- Linear Classifiers; Fisher Lin. Discrimants
- Support Vector Machines; Deep Neural Networks

Major tool of Data Mining: Dimension reduction

Goal is not as much to reduce size (& cost) but to:

- Reduce noise and redundancy in data before performing a task [e.g., classification as in digit/face recognition]
- Discover important ‘features’ or ‘parameters’

The problem: Given: \( X = [x_1, \cdots, x_n] \in \mathbb{R}^{m \times n} \), find a low-dim. representation \( Y = [y_1, \cdots, y_n] \in \mathbb{R}^{d \times n} \) of \( X \)

Achieved by a mapping \( \Phi : x \in \mathbb{R}^m \rightarrow y \in \mathbb{R}^d \) so:
\[
\phi(x_i) = y_i, \quad i = 1, \cdots, n
\]

Basics: Principal Component Analysis (PCA)

**PCA:** Compute \( W \) to maximize variance of projected data:

\[
\text{max}_{W \in \mathbb{R}^{m \times d}, W^T W = I} \sum_{i=1}^{n} \left\| y_i - \frac{1}{n} \sum_{j=1}^{n} y_j \right\|^2_2, \quad y_i = W^T x_i.
\]

 Leads to maximizing
\[
\text{Tr} \left[ W^T (X - \mu e^T)(X - \mu e^T)^T W \right], \quad \mu = \frac{1}{n} \sum_{i=1}^{n} x_i
\]

Solution \( W = \{ \text{dominant eigenvectors} \} \) of the covariance matrix \( \equiv \) Set of left singular vectors of \( \bar{X} = X - \mu e^T \)
**SVD:**

\[ \bar{X} = U \Sigma V^T, \quad U^T U = I, \quad V^T V = I, \quad \Sigma = \text{Diag} \]

- Optimal \( W = U_d \equiv \text{matrix of first} \ d \ \text{columns of} \ U \)
- Solution \( W \) also minimizes ‘reconstruction error’ ..

\[ \sum_i \|x_i - WW^T x_i\|^2 = \sum_i \|x_i - Wy_i\|^2 \]

- In some methods recentering to zero is not done, i.e., \( \bar{X} \) replaced by \( X \).

**Unsupervised learning**

"Unsupervised learning": methods do not exploit labeled data

- Example of digits: perform a 2-D projection
- Images of same digit tend to cluster (more or less)
- Such 2-D representations are popular for visualization
- Can also try to find natural clusters in data, e.g., in materials
- Basic clustering technique: K-means

**Example: Digit images (a random sample of 30)**

**2-D 'reductions':**

PCA - digits: 0 −− 4
LLE - digits: 0 −− 4
K-PCA - digits: 0 −− 4
ONPP - digits: 0 −− 4
Supervised learning

- We now have data that is 'labeled'

**Examples:** Health Sciences (‘malignant’- ‘non malignant’); Materials (‘photovoltaic’, ’hard’, ’conductor’, ...) ; Digit Recognition (‘0’, ‘1’, ...., ‘9’)

**Supervised learning: classification**

- Best illustration: written digits recognition example

  **Given:** set of labeled samples (training set), and an (unlabeled) test image $x$.

  **Problem:** label of $x = ?$

- Roughly speaking: we seek dimension reduction so that recognition is 'more effective' in low-dim. space
**Basic method: K-nearest neighbors (KNN) classification**

- Idea of a voting system: get distances between test sample and training samples
- Get the \( k \) nearest neighbors (here \( k = 8 \))
- Predominant class among these \( k \) items is assigned to the test sample ("∗" here)

**Supervised learning: Linear classification**

- **Linear classifiers:** Find a hyperplane which best separates the data in classes A and B.
- Example of application: Distinguish between SPAM and non-SPAM e-mails
- Note: The world in non-linear. Often this is combined with Kernels — amounts to changing the inner product

**A harder case:**

- Use kernels to transform

**Transformed data with a Gaussian Kernel**
**Simple linear classifiers**

- Let \( X = [x_1, \ldots, x_n] \) be the data matrix.
- and \( L = [l_1, \ldots, l_n] \) == labels. \( l_i = \pm 1 \)
- 1st Solution: Find a vector \( u \) such that \( u^T x_i \) close to \( l_i \), \( \forall i \)
- Common solution: SVD to reduce dimension of data \([\text{e.g. } 2-D]\) then do comparison in this space. e.g.

\[ A: u^T x_i \geq 0, \quad B: u^T x_i < 0 \]

[For clarity: principal axis \( u \) drawn below where it should be]

**Fisher’s Linear Discriminant Analysis (LDA)**

**Principle:** Use label information to build a good projector, i.e., one that can ‘discriminate’ well between classes

- Define “between scatter”: a measure of how well separated two distinct classes are.
- Define “within scatter”: a measure of how well clustered items of the same class are.
- Objective: make “between scatter” measure large and “within scatter” small.

**Idea:** Find projector that maximizes the ratio of the “between scatter” measure over “within scatter” measure

\[
S_B = \sum_{k=1}^c n_k (\mu(k) - \mu)(\mu(k) - \mu)^T,
\]

\[
S_W = \sum_{k=1}^c \sum_{x_i \in X_k} (x_i - \mu(k))(x_i - \mu(k))^T
\]

where:
- \( \mu = \text{mean} (X) \)
- \( \mu(k) = \text{mean} (X_k) \)
- \( X_k = k\text{-th class} \)
- \( n_k = |X_k| \)

**Consider 2nd moments for a vector** \( a \):

\[
a^T S_B a = \sum_{i=1}^c n_k |a^T(\mu(k) - \mu)|^2,
\]

\[
a^T S_W a = \sum_{k=1}^c \sum_{x_i \in X_k} |a^T(x_i - \mu(k))|^2
\]

\( a^T S_B a \equiv \text{weighted variance of projected } \mu_j's \)

\( a^T S_W a \equiv \text{w. sum of variances of projected classes } X_j's \)

LDA projects the data so as to maximize the ratio of these two numbers:

Optimal \( a \) = eigenvector associated with top eigenvalue of:

\[
S_B u = \lambda_i S_W u_i.
\]
LDA – Extension to arbitrary dimensions

- Criterion: maximize the ratio of two traces:
  \[ \frac{\text{Tr}[U^T S_B U]}{\text{Tr}[U^T S_W U]} \]
- Constraint: \( U^T U = I \) (orthogonal projector).
- Reduced dimension data: \( Y = U^T X \).

**Common viewpoint:** hard to maximize, therefore ...

- ... alternative: Solve instead the (‘easier’) problem:
  \[ \max_{U^T S_W U = I} \text{Tr}[U^T S_B U] \]
- Solution: largest eigenvectors of \( S_B u_i = \lambda_i S_W u_i \).

In Brief: Support Vector Machines (SVM)

- Similar in spirit to LDA. Formally, SVM finds a hyperplane that best separates two training sets belonging to two classes.
- If the hyperplane is:
  \[ w^T x + b = 0 \]
  \[ y_i (w^T x_i + b) \geq 1, \ \forall x_i. \]
- Need to solve the constrained quadratic programming problem:
  \[ \min_{w,b} \frac{1}{2} ||w||^2 \]
  \[ \text{s.t. } y_i (w^T x_i + b) \geq 1, \ \forall x_i. \]

**Modification 1:** Soft margin. Consider hinge loss: \( \max \{0, 1 - y_i [w^T x_i + b]\} \)

- Zero if constraint satisfied for pair \( x_i, y_i \). Otherwise proportional to distance from corresponding hyperplane. Hence we can minimize
  \[ \lambda ||w||^2 + \frac{1}{n} \sum_{i=1}^{n} \max \{0, 1 - y_i [w^T x_i + b]\} \]

**Modification 2:** Use in combination with a Kernel to improve separability.
A few words on Deep Neural Networks (DNNs)

- Ideas of neural networks goes back to the 1960s - were popularized in early 1990s – then laid dormant until recently.
- Two reasons for the come-back:
  - DNN are remarkably effective in some applications
  - big progress made in hardware [→ affordable ‘training cost’]

Training a neural network can be viewed as a problem of approximating a function $\phi$ which is defined via sets of parameters:

Problem: find sets of parameters such that $\phi(x) \approx y$

Input: $x$, Output: $y$
Set: $z_0 = x$
For $l = 1 : L+1$ Do:
  $z_l = \sigma(W^T_l z_{l-1} + b_l)$
End
Set: $y = \phi(x) := z_{L+1}$

- layer # 0 = input layer
- layer # ($L + 1$) = output layer

- A matrix $W_i$ is associated with layers 1, 2, $L + 1$.
- Problem: Find $\phi$ (i.e., matrices $W_i$) s.t. $\phi(x) \approx y$

DNN (continued)

- Problem is not convex, highly parameterized, ...,
- .. Main method used: Stochastic gradient descent [basic]
- It all looks like alchemy... but it works well for certain applications
- Training is still quite expensive – GPUs can help
- *Very* active area of research