Part I
Course recap

In this part of the lesson we tried to cover the list of all the topics that were learnt in ths course. Here is the list we covered:

Algorithms

Unsupervised

- Clustering: Group ("cluster") points into clusters
  - k-Means - both the hard version (point in cluster) and the soft version (probability per cluster)
  - Gaussian Mixture Model (GMM)
  - Spectral Clustering
- Dimensionality Reduction
  - Principal Component Analysis (PCA)
    - Including the probabilistic interpretation

Supervised

- Binary Classification
  - Naive Bayes
  - k-Nearest Neighbors (k-NN)
  - Separating Hyperplanes:
    - SVM
      - Regular (Hard Margin)
      - Soft Margin
      - Kernels - Polynomial, Gaussian, Linear, and more isoteric kernels
    - Perceptron
      - Regular
      - Margin Perceptron (finding a higher margin)
    - Winnow
    - Logistic regression
      - Need to add threshold to move from probabilities to classification
  - Decision trees
  - Boosting methods
    - Random forest
    - AdaBoost
    - Majority
    - Bagging / Stacking
- Multiclass (categorical) classification
  - Conversion from binary

        * one vs. one - train classifiers for each pair, and somehow sum all decisions
        * one vs. all - train classifier for each "one" and "rest", and then somehow combine all decisions
    - Decision tree
    - Naive Bayes
    - (k-)Nearest Neighbor
- Regression
    - Linear Regression
        * Lasso / Ridge
    - Logistic Regression
    - k-Nearest Neighbors
        * Use mean instead of majority
    - SVR (similar to SVM)

Tools

Optimization

- Estimation Maximization (EM)
- Linear programming (linear target and linear constraints)
- Quadratic programming (same, with qudratic equations)
- Gradient Descent
- Coordinate Gradient Descent (go in lines on one coordinate at a time)

Statistics

- Machine Learning
- Maximal A-postriori (MAP), Bayseian

General

- Singular Value Decomposition (SVD)
- Penalty / Regularization
- Probably Approximate Classification (PAC)

Part II
More insights into PCA

Given a set of points $\bar{x}_1, ..., \bar{x}_n \in \mathbb{R}^d$, let X be a matrix with those points as it's columns:

$$X = \begin{pmatrix} | & & | \\ \bar{x}_1 & \dots & \bar{x}_n \\ | & & | \end{pmatrix}$$

Assuming these points are centered around the origin (i.e. $\sum_{i=1}^{n} \bar{x}_i = \bar{0}$), PCA is doing a spectral decomposition (finding eigenvalues and eigenvectors) of XX^T of size $d \times d$.

1 Complexity

Computing XX^T is $O\left(d^2 \cdot n\right)$, and the spectral decomposition (diagonalization) is $O\left(d^3\right)$. But, what happens if $n \ll d$?

- For example, in images we may have millions of pixels but we don't necessarily have millions of images
- Other examples for such cases are when dealing with medical data (genetics, MRI, and other checks)
- Finally, another example from the Natural Language Processing (NLP) field is the "Bag of words" descriptor. Assuming we have a big corpus of text, and we create a vector counting the occurences of each word with a text sample. It won't be suprising if the amount of words would be significantly larger than the amount of text samples we have, and in that case we'd have $n \ll d$.

So let's look at the SVD of X:

$$X_{d\times n} = U_{d\times d}\Sigma_{d\times n}V^T \qquad (V_{n\times n})$$

And then

$$XX^T = \left(U\Sigma V^T\right)\left(V\Sigma^T U^T\right)$$

Since V is an orthonormal matrix, $V^TV = I$. Furthermore, remember that Σ is essentially a really high matrix $(d \times n)$ with values on it's "diagonal"

$$\Sigma = \begin{pmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \\ & & & \\ 0 & & & \end{pmatrix}$$

Combining those, we get the following:

$$XX^T = \left(U\Sigma V^T\right)\left(V\Sigma^T U^T\right) = U\Sigma\Sigma^T U^T = U \cdot diag\left(\sigma_1^2, \dots \sigma_n^2\right) \cdot U^T$$

In PCA we usually compute the eigenvectors $\bar{u}_1, ..., \bar{u}_n \in \mathbb{R}^d$ and the eigenvalues $\sigma_1^2, ..., \sigma_n^2$ from XX^T. But, for our case of $d \ll n$, let's try doing so for the transpose - X^TX:

$$X^TX = V \cdot diag(\sigma_1^2, ..., \sigma_n^2) \cdot V^T$$

The resulting eigenvectors are $\bar{v}_1, ..., \bar{v}_n \in \mathbb{R}^n$ and the matching eigenvalues are $\sigma_1^2, ..., \sigma_n^2$. But, note that for $i = 1, ..., n$ we have:

$$X\bar{v}_i = \left(U\Sigma V^T\right)\bar{v}_i = U\Sigma\bar{e}_i^n$$

Let's explain the above:

- Multiplying a vector by V^T means to transfer it to be respresented over the basis spanned by the columns of V
- In that case, the i-th column $(\bar{v}_i)$ would be represented by a vector of length n with 1 at the i-th coordinate and zeros elsewhere (we denoted that as $\bar{e}_i^n$)

Continuing the above computation

$$\dots = U \cdot \left(\sigma_i \cdot \bar{e}_i^d\right) = \sigma_i U\bar{e}_i^d = \sigma_i\bar{u}_i \qquad \Rightarrow \qquad X\bar{v}_i = \sigma_i\bar{v}_i$$

So how do we do PCA for $n \ll d$?

1. Compute $X^T X$ (size $n \times n$)
2. Compute the eigenvectors and eigenvalues of the above - $\bar{v}_1, ..., \bar{v}_n$ and $\sigma_1^2, ..., \sigma_n^2 = \lambda_1, ..., \lambda_n$
3. Compute $\bar{u}_i = \frac{1}{\sigma_i} \cdot X\bar{v}_i$

Complexity:

1. Matrix mulutiplication - computing $n \cdot n$ dot products of length d - $O\left(n^2 d\right)$
2. Spectral clustering - moved from $O\left(d^3\right)$ to $O\left(n^3\right)$
3. Vector computation - $n$vectors of size d, computed by a dot product of length n - $O\left(n^2 d\right)$

And so the overall complexity we get is $O\left(n^2 d + n^3\right)$ instead of $O\left(d^2 n + d^3\right)$, and that's a significant improvement when $n \ll d$!

Part III
Question

We have a regression problem that solving it is the optimization problem

$$\min_{a,b \in \mathbb{R}} \sum_{i=1}^{n} |y_i - ax_i - b|$$

For $y, x_1, ..., x_n \in \mathbb{R}$

1. Which probablistic model's maximal likelihood would be optimized by solving the above problem?
2. Solve this problem with linear programming

Answer 1

Assume that $y_i = ax_i + b + \varepsilon_i$ and let's assume that $\Pr\left(\varepsilon_i = x\right) = f\left(x\right)$. Then, computing the (log) likelihood we get

$$l\left(a, b; \{x_i, y_i\}\right) = \sum_{i=1}^{n} \log \Pr_{a,b}\left(y_i | x_i\right) = \sum_{i=1}^{n} \log \Pr_{a,b}\left(\varepsilon_i = y_i - ax_i - b\right) = \sum_{i=1}^{n} \log f\left(y_i - ax_i - b\right)$$

So, if $f\left(x\right) = C \cdot e^{-|x|}$ then

$$= \sum_{i=1}^{n} \log C - |y_i - ax_i - b|$$

So maximizing the log-likelihood is equivalent to solving the original minimzation problem.

Answer 2

The linearity problem comes from the absolute values, so we'll "invent" new variables

$$\min_{a,b,\bar{\xi}} \sum_{i=1}^{b} \xi_i$$

Under the constraints

$$\xi_i \geq y_i - ax_i - b \qquad \xi_i \geq -\left(y_i - ax_i - b\right)$$