Information Retrieval in High Dimensional Data

1. Basics

Observation Matrix: $\boldsymbol{X} \subset \mathbb{R}^{pxn}$ with p-dimensional random variable and n obeservations (i: variable, j: observation)
Expected Value $\mu = E[X] \in \mathbb{R}^p$
Estimate mean: $\hat{\mu}_i = \frac{1}{n}\sum_{j=1}^n x_{ij}$ (Sum row wise!)
Variance $\Sigma = Var(X) = E[(X-\mu)(X-\mu)^T] \in \mathbb{R}^{pxp}$
Σ is symmetric and positive semidefinite: $x^T\Sigma x \geq 0\ \forall x$ or $\Sigma \geq 0$
Centered Observation Matrix: $\hat{X}_{ij} = x_{ij} - \hat{\mu}_i$ Transpose Matrix: $(XY)^T = Y^T X^T$

1.1. Random Variables

Probability: $Pr(X \in \mathcal{X}) = \int_{\mathcal{X}} p_X(x)dx = \sum_{\{i|x_i \in \mathcal{X}\}} p_i$
Marginal Density: $p_X(x) = \int_{\mathbb{R}^k} p_{X,Y}(x,y)dy$
Conditional Density: $p_{Y|X=x(y)} = \frac{p_{X,Y}(x,y}{p_X(x}$ Expectation Value: $E[X] = \int_{\mathbb{R}^p} x p_X(x)dx = \mu_X$
(Co)Variance: $Var[X] = E[(X-\mu_X)(X-\mu_X)^T$

2. Statistical Decision Making

2.1. Loss Function

- Quadratic Loss Function: $L(Y, f(X) = (Y - f(X))^2$
- Minimize Expected Prediction Error: $EPE(f) = \mathrm{E}[L(Y, f(X)]$
- If using quadratic loss function, conditional mean is best
- If using absolute loss, conditional median is best

2.2. Decision Making

- *Global Methods*: Find best explicit global function $\hat{f}$
  $\hat{f} = argmin_{f \in \mathcal{F}} EPE(f)$, incorporate all points
  reduce complexity by learning parametrize decision function
- *Local Methods*: Find best local value $\hat{c}$ of given realization x
  $\hat{c} = argmin_{c \in \mathbb{R}} \mathrm{E}_{Y|X=x} L(Y,c)$, only samples in region of interest

2.3. Curse of Dimensionality

With increasing dimension p:

- Noise increases (accumulates over dimensions
- Number of observations for same accuracy increases exponentially
- *Empty Space Phenomenon*: High dimensional spaces are sparse
- Bigger space = more samples in the tail of the distribution
- Samples are equidistant to each other

Therefore, difficult to estimate underlying distribution

2.4. Data Preparation

- **No**minal Categories - **No** ordering, **Or**dinal Categories: Ordering
- Num to Cat: Discretization, Cat to Num: Binarization
- *Bag of Words*: Frequent and Distinct = strong Weight
- Frequent: Term Frequency
- Distinct: Inverse Document Frequency
- Text Prep: Remove HTML, lower case, Remove punctuation/numbers/common words, split into words

3. Logistic Regression

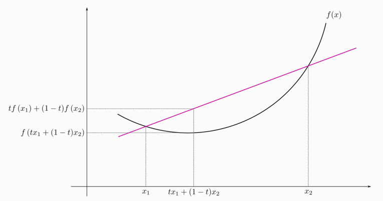
3.1. Binary Classification

Input Data: $X \in \mathbb{R}^p$, Output Variables: $Y \in \{-1, +1\}$
$L_{0,1}(Y, f(X) = 1 \quad$ if $Y \cdot sign(f(X)) \leq 0$, numerically infeasible

3.2. Convexity

$$\boxed{\text{in convex set } \mathcal{C} \subset \mathbb{R},\ f \text{ is convex, if } tf(\boldsymbol{x_2} + (1-t)f(\boldsymbol{x_1}) \geq f(t\boldsymbol{x_2} + (1-t)\boldsymbol{x_1} \quad \forall \boldsymbol{x_1}, \boldsymbol{x_2} \in \mathcal{C}, t \in [0,1]}$$

if f, g convex, then: $max(f,g)$, $f+g$, $g \circ h$ (if g non-decreasing) are convex. Local minimum of *strictly* convex is global minimum and unique



3.3. Logistic Regression

Choose $f(\boldsymbol{x}) = \boldsymbol{w^T x} + b$ and log-loss $l(t) = log(1 + e^{-t})$:

$$\boxed{\min_{\boldsymbol{w} \in \mathbb{R}^n, b \in \mathbb{R}} \frac{1}{n}\sum_{i=1}^n log(1 + exp(-y_i(\boldsymbol{w^T x_i} + b)))}$$

$P(Y = y|x) = \frac{1}{1+exp(-y(\boldsymbol{w^T x}+b))}$, Prob. that y is correct
Find $\boldsymbol{w^*}, b^*$ by gradient descent
Classify new sample: $sign(\boldsymbol{w^{*T} x_{new}} + b^*)$

3.4. Overfitting

For linearly separable non empty training sets, the loss function has no global minimum in $\mathbb{R}^{p+1}$. If there is a dividing hyperplane (=lin. separable), we can scale its parameters and increase the value of the loss function. Dies does not need to be the best dividing hyperplane! Fixed by regularizer, e.g. adding $\lambda||\boldsymbol{w}||^2 + b^2$

3.5. Alternative Approach (Statistics)

Linear Model: $a = w_0 + w_1x_1 + \ldots + w_nx_m = \boldsymbol{w^T x}$
Probability: $\sigma(a) = \frac{1}{1+e^{-a}}$ $\qquad D = \{(\boldsymbol{x_i}, z_i)\}$
Find $\boldsymbol{w}_{MLE} = arg\max_{\boldsymbol{w}} P(D|\boldsymbol{w})$ $\qquad \boldsymbol{x_i} \in \mathbb{R}^d, z_i \in \{0,1\}$

$P(D|\boldsymbol{w}) = \prod_{i=1}^n \sigma(\boldsymbol{w^T x_i})^{z_i}(1 - \sigma(\boldsymbol{w^T x_i})^{1-z_i}$

$L(\boldsymbol{w} = -log(P(D|\boldsymbol{w}), \nabla_{\boldsymbol{w}} L(\boldsymbol{w}) = \boldsymbol{X}(\sigma(\boldsymbol{X^T w}) - \boldsymbol{z})$
(Requires Hessian Matrix, which is positive semi-definite)
Results are identical up to the factor 1/n

4. Kernels

Kernels replace the standard inner products with some function that is *a suitable generalization of an inner product* to allow for nonlinear behavior.

$$\boxed{\text{A positive semidefinite Kernel is a function } \kappa : \mathbb{R}^{p\times} \to \mathbb{R} \text{ so that for all sets } \mathbf{X} = \{\mathbf{x}_1, \cdot, \mathbf{x_n}\} \text{ the } \textit{Gram-Matrix } \mathbf{K}_{ij} = \kappa(\mathbf{x_i}, \mathbf{x_j}) \text{ is symmetric and positive semi definite}}$$

- Symmetric: $\kappa(\mathbf{x_i}, \mathbf{x_j}) = \kappa(\mathbf{x_j}, \mathbf{x_i})\ \forall i, j$
- positive semi definite: $\mathbf{x^T K x} \geq 0\ \forall \mathbf{x}$ or all eigenvalues ≥ 0

4.1. Common Kernels and Rules

- Linear kernel: $\kappa(\mathbf{x}, \mathbf{y} = \mathbf{x^T y} + c,\ c \geq 0$
- Polynomial Kernel $\kappa(\mathbf{x}, \mathbf{y} = (a\mathbf{x^T y} + c)^d,\ a, c, d \geq 0$
- Gaussian Kernel $\kappa(\mathbf{x}, \mathbf{y} = exp(-\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2}$
- If κ_1, κ_2 are Kernels and $c \geq 0$, then $c\kappa_1$, $c + \kappa_1$, $\kappa_1 + \kappa_2$ and $\kappa_1 \cdot \kappa_2$ are kernels as well

5. Principal Component Analysis

Unsupervised Learning: Structure unlabeled data. Here: reduce dimension of input without loosing to much information

Goal: Find $\boldsymbol{U}_k \in \mathbb{R}^{pxk}$ that minimizes $\sum_{i=1}^n ||\boldsymbol{x}_i - \boldsymbol{U_k U_k^T x_i}||_2^2$

5.1. PCA - Singular Value Decomposition

Important: $\boldsymbol{X}$: centered observation matrix SVD returns: $\boldsymbol{X} = \boldsymbol{XDV^T}$ with singular values $d_1 \geq d_2 \geq \ldots \geq d_n$ on the diagonal of **D**
$\boldsymbol{U_k}$ satisfies goal with diagonal covariance matrix $\boldsymbol{S} = \boldsymbol{U_k^T X}$
s_{ij}: j-th score of the i-th principal component. S: score matrix, U: loadings matrix
To reduced variables: $\boldsymbol{S} = \boldsymbol{D_k V_k^T}$, new sample: $s_{new} = \boldsymbol{U_k^T} s_{new}$
Cheaper than using $\boldsymbol{U_k^T X}$ (nk operations instead $(2p-1)nk$
Dimensions:
$X \in \mathbb{R}^{m\times n}$, $U \in \mathbb{R}^{m\times m}$, $D \in \mathbb{R}^{m\times n}$, $V^T \in \mathbb{R}^{n\times n}$, $U_k \in \mathbb{R}^{k\times n}$, $X_k \in \mathbb{R}^{k\times n}$, m: Dimension variable, n: samples
The eigenvectors of $\mathbf{X_a X_a^T}$ are the loadings of $\mathbf{X_a}$ only if X_a is centered

5.2. Statistical Interpretation

Goal: Find $Y = \boldsymbol{U^T} X$ with $\boldsymbol{U^T U} = \boldsymbol{I}$ and declining variance of the components

5.3. Autoencoders

Autoencoder: Form of neural networks, where input is first mapped to a lower dimension k by f and then back to input g: $g \circ f(x_i) \approx x_i$. If f and g are linear and represented by $\boldsymbol{V} \in \mathbb{R}^{k\times p}$ and $\boldsymbol{W} \in \mathbb{R}^{p\times k}$ then $g \circ f(x_i) = \boldsymbol{WV}x_i$ and the error is measuered by the sum of squared distances $\sum_{i=1}^n ||x_i - \boldsymbol{XV}x_i||^2$, then the first k singular vectors of **X** are optimal.
WV has at most rank k
If f, g not linear: only approximation possible

6. Kernel-PCA

6.1. Linear PCA by inner products

If $\mathbf{K} = \mathbf{X^T X}$ the inner product matrix and $\mathbf{X} = \mathbf{U^T}$ the SVD of **X**, we can write $\mathbf{K} = \mathbf{V\Sigma^T \Sigma V^T}$ with $\Sigma^T\Sigma$ being diagonal. This is the eigenvalue decomposition of **X**
Let $\mathbf{V_k}$ and $\mathbf{\Sigma_k} = diag(\sigma_1, \cdots \sigma_k$ the first k eigenvectors and eigenvalues

$$\boxed{\begin{gathered}\text{Therefore: } \mathbf{U_k^T} = \mathbf{\Sigma_k^{-1} V_k^T X^T} \text{ and for a new sample } \mathbf{y}\text{:} \\ \mathbf{U_k^T y} = \mathbf{\Sigma_k^{-1} V_k^T X^T y} = \mathbf{\Sigma_k^{-1} V_k^T}\begin{bmatrix}\mathbf{x_1^T y} & \cdots \mathbf{x_n^T y}\end{bmatrix}^T \\ \text{This only requieres the inner product } \mathbf{x_n^T y} = \mathbf{k_y}\end{gathered}}$$

If **X** is not centered, we can center **K** without centering **X**:

$$\tilde{\mathbf{K}} = \mathbf{HKH} \text{ with } \mathbf{H} = (\mathbf{I}_n \cdot \tfrac{1}{n}\mathbb{1}_n\mathbb{1}_n^T)$$

For new samples, we have to replace $\tilde{\mathbf{y}} = \mathbf{y} - \frac{1}{n}\mathbf{X}\mathbb{1}_n$ and $\tilde{\mathbf{X}} = \mathbf{XH}$

$$\mathbf{U_k^T}(\tilde{\mathbf{y}}) = \mathbf{\Sigma_k^{-1} V_k^T}\tilde{\mathbf{k}}_\mathbf{y}, \quad \tilde{\mathbf{k}}_\mathbf{y} = \mathbf{Hk_y} - \tfrac{1}{n}\mathbf{HK}\mathbb{1}_n$$

6.2. Transition to Kernel PCA

Instead of replacing the inner product $\mathbf{x^T y}$ by $\langle\phi(\mathbf{x}), \phi(\mathbf{y})\rangle$, we substitute $\mathbf{x^T y} \to \kappa(\mathbf{x}, \mathbf{y})$:

$$\boxed{\begin{gathered}\mathbf{k}^{new} = \begin{bmatrix}kappa(\mathbf{x_1}, \mathbf{y}) & \cdots & kappa(\mathbf{x_n}, \mathbf{y})\end{bmatrix}^T \\ \mathbf{k}_{cent}^{new} = \mathbf{Hk}_{cent}^{new} - \tfrac{1}{n}\mathbf{HK}\mathbb{1}_n\end{gathered}}$$

Kernel Principal Component Analysis

1. for training set $\mathbf{X} = \begin{bmatrix}\mathbf{x}_1 & \cdots & \mathbf{x}_n\end{bmatrix}$, $\mathbf{x}_1 \in \mathbb{R}^p$
2. Find suitable Kernel function $\kappa(\cdot)$ and compute Gram Matrix $\mathbf{K}$
3. Compute centered Gram Matrix: $\tilde{\mathbf{K}} = \mathbf{HKH}$ with $\mathbf{H} = \mathbf{I}_n - \frac{1}{n}\mathbb{1}_n\mathbb{1}_n^T$
4. Compute Eigenvalue Decomposition: $\tilde{\mathbf{K}} = \mathbf{V\Lambda V^T}$. Because **K** is positive semi-definite and therefore the diagonal entries of $\mathbf{\Lambda}$ are non-negative, we write $\mathbf{\Lambda} = \mathbf{\Sigma}^2 = diag(\sigma_1^2, \cdot, \sigma_n^2)$
5. Reduced Matrices: $\mathbf{\Sigma_k} = diag(\sigma_1, \cdots, \sigma_k)$, $\in \mathbb{R}^{k\times k}$ and $\mathbf{V}_k \in \mathbb{R}^{n\times k}$
6. Reduced Training Data: $\mathbf{S} = \mathbf{\Sigma_k V}_k^T$
7. For new datapoint $\mathbf{y} \in \mathbb{R}^p$, compute new components:

$$\mathbf{s}_n ew = \mathbf{\Sigma}_k^{-1}\mathbf{V}_k^T\mathbf{k}_{cent}^{new}$$
$$\mathbf{k}_{cent}^{new} = \mathbf{Hk}_{cent}^{new} - \tfrac{1}{n}\mathbf{HK}\mathbb{1}_n$$
$$\mathbf{k}^{new} = \begin{bmatrix}kappa(\mathbf{x_1}, \mathbf{y}) & \cdots & kappa(\mathbf{x_n}, \mathbf{y})\end{bmatrix}^T$$

7. Feedforward Neural networks

Minimize any expected loss at the price of training many parameters
Find $f \in \mathcal{F}$ that minimizes loss. If all functions in $\mathcal{F}$ can be described by a set of k parameters $\Theta \in \mathbb{R}^k$, then the goal is for N samples:

$$\hat{\Theta} = arg\min_{\Theta \in \mathbb{R}^k} = \frac{1}{N}\sum_i L(f_\Theta(x_i))$$

One layer of a FNN consists of a linear function $\varphi_{\boldsymbol{W}}(\boldsymbol{h}) = \boldsymbol{Wh}$ and a nonlinear *activation function*, e.g. the ReLU: $\sigma(t) = max\{0, t\}$:

$$f : \mathbb{R}^p \to \mathbb{R}^k : \sigma_l \circ \varphi_{\boldsymbol{W_l}} \circ \cdots \circ \sigma_1 \circ \varphi_{\boldsymbol{W_i}}(\boldsymbol{X})$$

With p/k Input/Output dimension and l layers, *Deep* FNN: $l > 3$
k depends on loss function (Output of FNN is input of Loss)

7.1. Training FNNs (Backpropagation)

$$\boxed{\begin{gathered}\text{Functions } g : \mathbb{R}^k \to \mathbb{R}^l \text{ and } h : \mathbb{R}^l \to \mathbb{R}^m \text{ with } g \text{ being diff'able at } \mathbf{x} \text{ and } h \text{ at } \boldsymbol{y} = g(\boldsymbol{x}) \text{ with Jacobi matrices } \boldsymbol{J}_g(\boldsymbol{x}) \text{ and } \boldsymbol{J}_h(\boldsymbol{y}) \text{ then} \\ h \circ g : \mathbb{R}^k \to \mathbb{R}^m \text{ is diff'able at } \mathbf{x} \text{ with} \\ \boldsymbol{J}_{h\circ g}(\boldsymbol{x}) = \boldsymbol{J}_h(g(\boldsymbol{x}) \cdot \boldsymbol{J}_g(\boldsymbol{x})\end{gathered}}$$

8. Support Vector Machine (Basic Linear SVM)

- Idea: Find hyperplane to divide $\boldsymbol{x}, y$ into two subspaces
- Hyperplane: $\mathcal{H}_{\boldsymbol{w},b} = \{x \in \mathbb{R}|\boldsymbol{w^T x} - b = 0\}$, $\boldsymbol{w}$ normal to $\mathcal{H}$
- Euclidean distance from $\boldsymbol{x}$ to $\mathcal{H}_{\boldsymbol{w},b}$: $\delta(\boldsymbol{x}, \mathcal{H}_{\boldsymbol{w},b}) = \frac{\boldsymbol{w^T x} - b}{||\boldsymbol{w}||}$
- Goal: Find $y_i(\boldsymbol{w^T x_i} - b) \geq 1 \quad \forall i \in 1, \ldots, N$
- Optimization Problem: $min\frac{1}{2}||\boldsymbol{w}||^2$ that fulfills Goal

8.1. Karush Kuhn Tucker Conditions and Linear SVMs

$$\boxed{\begin{gathered}\min_{\boldsymbol{w}, b, \boldsymbol{\lambda} \geq 0} L(\boldsymbol{w}, b, \boldsymbol{\lambda}) \\ L(\boldsymbol{w}, b, \boldsymbol{\lambda}) = \tfrac{1}{2}||\boldsymbol{w}||^2 - \sum_i \lambda_i y_i(\boldsymbol{w^T x_i} - b) + \lambda_i \\ \nabla_{(\boldsymbol{w},b)} L(\boldsymbol{w}, b, \boldsymbol{\lambda}) = \begin{bmatrix}\boldsymbol{w} - \sum_i \lambda_i y_i \boldsymbol{x}_i \\ \sum_i \lambda_i y_i\end{bmatrix} \\ \boldsymbol{w^*} = \sum_i \lambda_i^* y_i \boldsymbol{x}_i \quad \sum_i \lambda_i^* y_i = 0 \\ L_D(\boldsymbol{\lambda}) = \sum_i \lambda_i - \tfrac{1}{2}\boldsymbol{\lambda^T H \lambda} \quad \text{With } h_{ij} = y_i y_x \boldsymbol{x_i^T x_j} \\ \text{New convex quadratic opt problem: } \boldsymbol{\lambda^*} = \max_{\boldsymbol{\lambda}} L_D(\boldsymbol{\lambda})\end{gathered}}$$

- Problem is strictly convex, therefore solution is unique
- If $\lambda_i \neq 0$, then $\boldsymbol{x_i}$ is support vector (Lies in or on margin)
- Only works, if classes can be linearly separated
- Else: Kernel/Soft Margin SVM required
- SVM works better when (nearly) separable than LR
- SVM is preferred when using kernels compared to LR

9. Decision Making

- **Decision making** is about mapping an input vector to an output value.
- **Binary decision making** can be visualised via a joint pdf of X and Y, where Y takes on -1 or 1 and X is a continuous random variable whose distribution changes conditioned on Y.
- **Loss** is a positive, real measure of the deviation between a target and an output from a decision function.
- **Expected prediction error** as a function of f is formulated by the expectation of the loss function, whose values change depending on the decision function f: $EPE(f) = E[L(Y, f(X))]$.
- The aim of **global methods** is to find the best global decision function f out of class F that minimises the expected prediction error.
- The aim of **local methods** is to find the best output which minimizes the expectation of loss on the distribution of Y conditioned on known X
- We can obtain expected prediction error and expected loss due to the assumption that we *know the joint pdf* of X and Y, i.e. the stochastic behaviour of the system.
- For local methods with loss specified as the squared loss, expected loss is found to be the conditional expectation of of Y on the distribution of Y conditioned on X = x: $f(X) = E_{Y|X=x}[Y]$.
- The problem with local methods is most of the time X is a continuous random variable and thus values here X=x are impossible, and so we take the **k-nearest neighbours** to x as approximations of x, thus giving us a set of samples representing the distribution of values X=x such that local method for finding output can now be applied to these samples.

10. Curse of Dimensionality

- Most of the time it is desired that the dimensionality of samples are decreased due to various problems involved with working in high dimensions.
- As the number of dimensions increases, the number of samples required to obtain an accurate representation of the probability distribution also increases a lot.
- The **empty space theorem** says that as dimensionality increases, samples tend to be located at the tail end of their distributions.
- As the number of dimensions increases, the distance between a point and any possible samples becomes more and more equidistant.
- For a random vector $X \in \mathbb{R}^p$, the probability of at least one X_i being further than β away from the center: (For large p this becomes 1)

$$Pr(||X||_2^2 \geq \beta) = 1 - Pr(X_1^2 < \beta)^p.$$

11. Convex Functions

Definition Convexity: $\mathcal{C} \subset \mathbb{R}^n$ a convex set, i.e for any pair $\boldsymbol{x_1}, \boldsymbol{x_2} \in \mathcal{C}$ the point $t\boldsymbol{x_2} + (1-t)\boldsymbol{x_1} \in \mathcal{C}\ \forall t \in [0,1]$. A function is called *convex* if $\forall \boldsymbol{x_1}, \boldsymbol{x_2} \in \mathcal{C}, t \in [0,1]$:
$tf(\boldsymbol{x_2}) + (1-t)f(\boldsymbol{x_1}) \geq f(t\boldsymbol{x_2} + (1-t)\boldsymbol{x_1})$

- By definition, a function is convex if a line can be drawn between any two points in a function, and all vertical values of the line are greater than or equal to all vertical values of the function between the same two points.
- A function is also convex if its second derivative is positive, or equivalently for vector inputs its Hessian matrix contains all non-negative eigenvalues.
- Several operations between convex functions also result in another convex function, including the max between two convex functions:

if f, g convex, then: $max(f, g)$, $f + g$, $g \circ h$ (if g non-decreasing) are convex. Local minimum of *strictly* convex is global minimum and unique

12. Logistic Regression

12.1. Formulation via Optimization

- The aim of logistic regression to perform binary classification on a random vector, that is, map a random vector to either -1 or 1.
- In order to ensure that the outputs in logistic regression are either -1 or 1, the output in logistic regression is taken to be the sign of the output from a prediction function f which contributes to the decision.
- The loss function in logistic regression is taken to be the number of misclassifications of a prediction, that is, 1 if the decision from logistic regression does not match the target, and 0 if it does match.
$$f(z) = \begin{cases} 1 & \text{if} \quad Y \cdot sign(f(x)) \leq 0 \\ 0 & \quad \text{otherwise} \end{cases}$$
- It is clear that we must find an appropriate f to complete the decision function, and thus we define the global method with the **misclassification loss** as the problem we need to solve. $\frac{1}{n}\sum\limits_{i=1}^{n} L_{0,1}(y_i, f(x_i))$
- The misclassification loss is extremely difficult to solve due to being non-continuous and non-convex, and so first we need to approximate this function as a convex function since they are easy to optimize.
- A convex approximation of misclassification loss is $log(1 + exp(-t))$, which when concatenated with an affine function $f(\boldsymbol{x}) = \boldsymbol{w^T x} + b$ is also convex, and can therefore be easily optimized via minimization.

$$\min_{\boldsymbol{w}\in\mathbb{R}^n, b\in\mathbb{R}} \frac{1}{n}\sum_{i=1}^{n} log(1 + exp(-y_i(\boldsymbol{w^T x}_i + b)))$$

- Noting that the loss is in **negative log-likelihood** form, we can convert this to a probability by taking the exponential of likelihood.

$$Pr(Y = y|x) = \frac{1}{1+exp(-y(\boldsymbol{w^T x}+b}$$

Gradient-based methods can be used to find affine weights and bias that optimize 3.3.

12.2. Overfitting on Logistic Regression

- When the samples are linearly separable, there is a constraint that the target times the affine output is always greater than zero Under this constraint, no global minimum exists.

If the data is linearly separable, we can find $(\boldsymbol{w_s}, b_s)$ so that:
$$y_i(\boldsymbol{w_s^T x_i} + b_s) < 0\ \forall i$$
then the lossfunction has no global minimum in $\mathbb{R}^{p+1}$

12.2.1. Proof

This is proven by noting that for a global minimum to exist, a point with small, positive loss exists (3.17) such that all points have losses greater than or equal to this point (3.16). This has to be satisfied along the linearly seperable constraint (3.18). If this constraint is always satisfied, then (3.19) is always positive, which when subbed into (3.15) and scaling weights and bias to infinity we find that the loss approaches zero (3.21), thus violating the global minimum condition (3.18). This is prevented via regularization (3.23).

13. Principal Component Analysis

PCA represents the data projected onto a lower-dimensional plane which captures most of the variance of the dataset.

13.1. Geometric Interpretation

- Imagining that x is a vector in p dimensions, and U_k describes a lower dimensional subspace, then the coordinates of x projected onto U_k is found by first obtaining the scores of x along U_k and then taking the linear combination of these scores and the U_k vectors: $\pi_{\mathcal{U}}(\boldsymbol{x}) = \boldsymbol{U_k U_k^T x},\ \boldsymbol{U_k} \in \mathbb{R}^{p\times k}$
- We want to find a U_k that captures most of the variance in the dataset, which is equivalent to finding a hyperplane where the difference between the original data points and their projections are minimized, thus forming our problem for optimization.

$$J(\boldsymbol{U_k}) = \sum_{i=1}^{n} ||\boldsymbol{x_i} - \boldsymbol{U_k U_k^T x_i}||_2^2$$

- The dataset can be represented as the dot product of an orthonormal matrix, a diagonal matrix and another orthonormal matrix via singular value decomposition: $\boldsymbol{X} = \boldsymbol{UDV}^T$.

The first k columns of the first orthonormal matrix via SVD $\boldsymbol{U_k}$ minimizes the the difference between data points and their projections, and the corresponding covariance matrix of the scores is diagonal (= features are uncorrelated).

13.2. Proof

- Theorem 4.1 is proved by first reframing the minimization problem (4.2) into a maximization problem, reframing this again into a problem involving the trace of a rank k projection matrix, then noting that the maximum value along the diagonals is 1 (4.5) of which subbing in U_k achieves due to its resulting in a identity matrix. We can show that the scores are uncorrelated by showing that the dot product between the scores and its transpose scaled by 1/n is a diagonal matrix using the SVD. (4.6)
- If we have the product between two matrices where the first matrix is is comprised of a diagonal matrix as well as a zero matrix, then we can get rid of the zero columns in the first matrix and also get rid of the corresponding rows in the left matrix.
- A computationally inexpensive way of obtaining the scores using the singular values and right singular vector can be derived by substituting the SVD into the formula for the calculating the scores. (4.7)

13.3. Statistical Interpretation

The idea behind the statistical interpretation is that covariance matrices are symmetric positive semidefinite, and therefore there exists some matrix U which can diagonalize it (clearly illustrated using the eigenvalue decomposition). Thus we can push U inside the covariance formula (4.8) to show that the scores with uncorrelated dimensions are obtained by projecting X onto U. (4.9)

$$\boldsymbol{D} = \boldsymbol{U^T} var(X)\boldsymbol{U} = E[\boldsymbol{U^T(x - \mu_x)(X - \mu_x)^T U}] =$$
$$E[(\boldsymbol{U^T}X - \boldsymbol{U^T}\mu_x)(\boldsymbol{U^T}X - \boldsymbol{U^T}\mu_x)^T] =$$
$$E[(\boldsymbol{Y} - \mu_y)(\boldsymbol{Y} - \mu_y)^T = var(\boldsymbol{Y})$$

13.4. Error Model Interpretation

- The error model interpretation is that the samples $\boldsymbol{X}$ is obtained via the addition between a signal matrix $\boldsymbol{L}$ and a noise matrix $\boldsymbol{N}$, where $\boldsymbol{L}$ is lying on a lower dimensional subspace than $\boldsymbol{X}$, and the goal is to try and find $\boldsymbol{L}$. Theorem 4.2
- Setting $\boldsymbol{L}$ equal to the SVD (p dimensions with k) minimizes the Frobenius norm of the difference between $\boldsymbol{X}$ and $\boldsymbol{L}$ given that the dimension k of the subspace which $\boldsymbol{L}$ lies in is known beforehand.

13.5. Relation to Autoencoders

- Autoencoders map an input to low dimensional space using a function f, which is then mapped back to a higher dimensional space using a function g and can approximate the input $g \circ f(\boldsymbol{x_i}) \approx \boldsymbol{x_i}$
- Letting f and g be linear mappings represented by matrices, the reconstruction error $J(\mathbf{W},\mathbf{V}) = \sum_{i=1}^{n} ||\boldsymbol{x_i} - \boldsymbol{WVx_i}||^2$ is minimised by setting f as the transpose of U_k and g as U_k.

Let U_k be the first k left singular vectors of observation matrix $\mathbf{X}$, the $\mathbf{V} = \mathbf{U}_k^T$ and $\mathbf{W} = \mathbf{U}_k$ minimize the reconstruction error of the linear autoencoder.

- This is proven by noting that in Theorem 4.2 which has a problem of the same form, the error is minimized using the projections of the points on the subspace of interest, thus leading to f and g as the transpose of U_k and U_k itself respectively as they cause this projection.

14. Feed forward Neural Networks

14.1. Definitions and Motivation

The power behind an FNN is its ability to minimize any kind of expected prediction error, that is, find model parameters that minimize expected loss (5.1). Feed forward neural networks are concatenations of linear functions (5.2) with activation functions (e.g. 5.3), which can be denoted in vector form. This results in the generalized form of an FNN. (5.5)

Given function class $\mathcal{F}$ described by set of k parameters $\Theta \in \mathbb{R}^< k$:
Solve: $\hat{\Theta} = arg\min_{\Theta\in\mathbb{R}^k} = \frac{1}{N}\sum_i L(f_\Theta(\mathbf{x_i}))$ (5.1)
$\varphi_\mathbf{W} : \mathbb{R}^p \to \mathbb{R}^m, \varphi_\mathbf{W}(\mathbf{h}) = \mathbf{Wh}$ (5.2)
$\sigma(t) = max\{0, t\}$ (ReLU, 5.3)
$f : \mathbb{R}^p \to \mathbb{R}^k : \sigma_l \circ \varphi_{\boldsymbol{W_l}} \circ \cdots \circ \sigma_1 \circ \varphi_{\boldsymbol{W_1}}(\boldsymbol{X})$(5.5)

14.2. Training FNN

The gradients can be calculated via chain rule through each subsequent layer, or in this case due to vectors via the Jacobian. From (5.5) the key stages include matrix multiplication (5.2) and nonlinear activation (5.3). The corresponding Jacobians w.r.t. inputs in each stage is seen in (5.6) and (5.7) respectively. However, for the Jacobian of matrix multiplication w.r.t weights, the weights need to be reshaped into a vector (5.9) which will then allow us to formulate the Jacobian (5.10). Then the gradients w.r.t loss (5.11) is simply (5.31a). The gradient can then be used to update the weights for the next time step, making sure to apply an inverse reshaping of the gradient vector into a matrix (5.14).

$\mathbf{J_{W}}_g(\mathbf{x}) = \mathbf{W} \cdot \mathbf{J_g}(\mathbf{x})$ (5.6)
$$\mathbf{j}_{(ij),\sigma}(\mathbf{x}) = \begin{cases} \sigma'(x_i) & \text{if} \quad i = j \\ 0 & \quad \text{else} \end{cases} \qquad (5.7)$$
$\mathbf{J}_{mult}(x) = diag(\mathbf{x}^T) \in \mathbb{R}^{m\times mn}$ (5.10), doesn't depent on
With $\mathbf{h}_j = \sigma_j \circ \varphi_{\mathbf{W_j}} \circ \cdots \circ \sigma_1 \circ \varphi_{\mathbf{W_1}}(\mathbf{x})$ the j-th output
We get for the output-layer (L):
$\frac{\partial}{\partial \mathbf{W}_l}F = \mathbf{J}_L(\mathbf{h}_l) \cdot \mathbf{J}_{\sigma_l}(\mathbf{W_l h_{l-1}}) \cdot \mathbf{J}_{mult}(\mathbf{h}_{l-1})$
$\frac{\partial}{\partial \mathbf{W}_j}F = \mathbf{J}_L(\mathbf{h}_l) \cdot \mathbf{J}_{\sigma_l}(\mathbf{W_l h_{l-1}}) \cdot$
$\mathbf{W_l J}_{\sigma_{l-1}}(\mathbf{W_{l-1} h_{l-2}}) \cdot \mathbf{J}_{l-1} \cdots \mathbf{J}_{mult}(\mathbf{h}_{l-1})$ for the j-th layer

15. Kernel Trick

- A kernel is a function that maps two sets containing real numbers (representing a sample) to a single scalar, such that for all possible sets of samples (contained in a data matrix with columns corresponding to samples) the Gram-Matrix (6.1) is symmetric and positive semidefinite.

A positive semidefinite Kernel is a function $\kappa : \mathbb{R}^{p\times} \to \mathbb{R}$ so that for all sets $\mathbf{X} = \{\mathbf{x}_1, \cdot, \mathbf{x_n}\}$ the *Gram-Matrix* $\mathbf{K}_{ij} = \kappa(\mathbf{x_i}, \mathbf{x_j})$ is symmetric and positive semi definite

- We can determine whether a particular function is a kernel by testing for violations of symmetry or positive semidefiniteness. Symmetry can easily be tested by substituting named variables into the kernel, then flipping around the variables and seeing whether the kernel expression is the same.
- Positive semidefiniteness can be tested as violated by definition, or if any diagonal values are ever negative. Furthermore, if the determinant of a matrix is negative it is definitely not psd (since psd eigenvalues are nonnegative and the determinant is the product of the eigenvalues), but we cannot say anything if the determinant is positive.

16. Kernel PCA

16.1. Linear PCA with Inner Products

- Scores of the input data can be calculated using singular values and the right eigenvector (4.7), which we can obtain using the eigenvalue decomposition of the Gram-Matrix (7.2) and therefore giving us a way to calculate linear PCA scores using the inner product. If we forgot to center the data, use the centred Gram-Matrix instead (7.8).
- If the input data has not been centred beforehand, we can find the Gram-Matrix for the centred data by noting how input data is usually centred (7.6), factoring out the input matrix (7.7) and then using inner product to compute the Gram-Matrix corresponding to centred input. (7.8)
- Scores for new observations are usually calculated by projecting them onto U_k, but if we want to reformulate this to be in terms of the singular value and right singular vectors we can isolate U in the SVD and retain only k columns (7.4a) to arrive at the formulation. (7.4)
- If we want to calculate scores for new observations considering centering, we take the formula for raw new observations (7.4), replace the input matrix with a centred input matrix and the observation with an approximated centred observation, resulting in a nice formula involving the Gram-Matrix. (7.9)

EVD of Gram Matrix (7.2): $\mathbf{K} = \mathbf{V\Sigma^T\Sigma V^T}$
Centered Gram Matrix (7.8): $\tilde{\mathbf{K}} = \mathbf{HKH}$ with
$\mathbf{H} = (\mathbf{I}_n . \frac{1}{n}\mathbb{1}_n\mathbb{1}_n^T)$
Computing scores for new sample $\mathbf{y}$: $\mathbf{U_k^T}(\tilde{\mathbf{y}}) = \mathbf{\Sigma_k^{-1}V_k^T\tilde{k}_y}$,
$\mathbf{\tilde{k}_y} = \mathbf{Hk_y} - \frac{1}{n}\mathbf{HK}\mathbb{1}_n$

16.2. Transition to Kernel PCA

- Kernel PCA requires us to first compute the Gram-Matrix via the kernel (6.1). This can then be substituted for the Gram-Matrix via inner product at various places, such as during centering (7.8) and when performing eigenvalue decomposition (7.2) to calculate matrices for scores (4.7).
- The scores for new observations via Kernel PCA is based off the Linear PCA version (7.4), and can be obtained by replacing the linear observation inner products with the kernel observation inner products (as well as ensuring that singular values and right singular vectors obtained from eigenvector decomposition of Gram-Matrix via kernel (7.2)).
- Similarly, when considering centering, note the Linear PCA version of scores which consider centering (7.9) and replace the linear Gram-Matrix with the kernel Gram-Matrix as well as the linear observation inner products with the kernel observation inner products. (7.12)

17. Support Vector Machines

17.1. Geometry

For some vector w, an affine hyperplane is described as the set of points x that can be projected onto w and then shifted by some constant b to equal zero. (8.1)

- The vector w is normal to the hyperplane (8.1) since a vector of any direction in the hyperplane projected onto w has a magnitude of zero. (8.2)
- The signed distance from any point to a hyperplane is defined as in (8.3), and can be interpreted as the distance from the hyperplane to the point x. It is formulated by first projecting x onto w and then centering the number line at b to obtain the signed value of x in the decision perspective (denote this as the decision distance). Then this value is divided by the magnitude of w to obtain the signed distance. Note that -b denotes the decision distance to the origin, and the magnitude of w is a scaling factor used to shift between the decision distance and signed distance.

Hyperplane:$\mathcal{H}_{\boldsymbol{w},b} = \{x \in \mathbb{R} | \boldsymbol{w^T x} - b = 0\}$ (8.1)
$\mathbf{w}$ is normal to $\mathcal{H}$: $\mathbf{w}^T(\mathbf{x_1} - \mathbf{x_2}) = 0$ (8.2)
Signed Distance: $\delta(\boldsymbol{x}, \mathcal{H}_{\boldsymbol{w},b}) = \frac{\boldsymbol{w^T x} - b}{||\boldsymbol{w}||}$ (8.3)
Margin:$\mathcal{H}_+ = \{x \in \mathbb{R} | \boldsymbol{w^T x} - b = 1\}$
$\mathcal{H}_- = \{x \in \mathbb{R} | \boldsymbol{w^T x} - b = -1\}$ (8.8-9)

- The decision perspective can be imagined as the perspective where classification decisions are made, all depending on the decision distance.
- The positive margin is defined as the hyperplane with a decision distance of 1 (8.8), and the negative margin is defined as the hyperplane with a decision distance of -1 (8.9).

17.2. Basic Linear SVM

- The aim of Linear SVM is to linearly separate data using the margins of the hyperplanes, such that all positively labelled data is bounded by the positive margin and all negatively labelled data is bounded by the negative margin. In other words, all positively labelled data has a decision distance greater than or equal to 1, and all negatively labelled data has a decision distance less than or equal to -1, resulting in a separation constraint for w and b. (8.12)
- We define the best w and b as the ones that have the widest margin, i.e. the widest distance between the hyperplanes. This width his calculated by the sum of the signed distance from the positive margin to the hyperplane and the signed distance from the hyperplane to the negative margin (8.13), which we can then maximize subject to our separation constraint (8.12).
- Flipping around the numerator and denominator of the width, this becomes a minimization problem (8.14).

Conditions: $\mathbf{w^T x}_i - b \geq +1$ for $y_i = 1$ and
$\mathbf{w^T x}_i - b \geq -1$ for $y_i = -1$
Compact: $y_i \cdot (\mathbf{w^T x}_i - b) \geq 1 \; \forall i$ (8.12) Minimization:
$\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||_2^2$ s.t. $y_i(\mathbf{w^T x}_i - b) \geq 1 \; \forall i$

17.3. KKT and Lagrangian Duality

The Karush-Kuhn Tucker conditions says that a minimisation problem with a set of equality and inequality constraints (8.15) can be reformulated as a Lagrangian primal (8.16) with some KKT conditions (Theorem 8.2), such that for convex objective functions with a convex feasible region (like that defined 8.14) minimising the primal (8.16) is equivalent to minimising the original problem (8.15).

- For a optimization problem with $\mathcal{E}$ and $\mathcal{I}$ as *equality and inequality constraints* (8.15):

$\min_{\mathbf{z}\in\mathbb{R}^n} f(\mathbf{z})$ s.t. $c_i(\mathbf{z}) = 0 \, \forall i \in \mathcal{E}$ and s.t. $c_j(\mathbf{z}) \geq 0 \, \forall j \in \mathcal{I}$
Lagrange Function (8.16): $L(\mathbf{z}, \lambda) = f(\mathbf{z}) - \sum_{i \in \mathcal{I} \bigcup \mathcal{E}} \lambda_i c_i(\mathbf{z})$

Karush-Kuhn-Tucker Conditions

For $\mathbf{z}^*$ as a solution to 8.15, there exists a larange multiplier λ^* such that:

- $\Delta_z L(z^*, \lambda^*) = 0$
- $c_i(z^*) = 0 \quad \forall i \in \mathcal{E}$
- $c_i(z^*) \geq 0 \quad \forall i \in \mathcal{I}$
- $\lambda_i^* \geq 0 \quad \forall i \in \mathcal{I}$
- $\lambda_i^* c_i(z^*) = 0 \quad \forall i \in \mathcal{I} \bigcup \mathcal{E}$

- The convex primal function (8.16) can then be reformulated as a concave dual function (8.22) by taking the infimum of the primal function. The infimum is the set of points along which the function is minimized w.r.t. the non-Lagrangian multiplier variables. Then by maximizing the primal subject to its constraints (8.22a) we obtain a lower bound for the solution to the primal (weak duality) or when some conditions are satisfied (such as in SVM) this coincides with the solution to the primal (strong duality).

17.4. Linear SVM via Lagrangian Duality

- SVM via Lagrangian duality follows the process specified above. The original problem is as in (8.14), the Lagrangian primal is as in (8.23 to 8.25) and the KKT conditions are as in (8.26 to 8.29).

Problem: $\min_{\boldsymbol{w},b,\boldsymbol{\lambda}\geq 0} L(\boldsymbol{w}, b, \boldsymbol{\lambda})$

$$L(\boldsymbol{w}, b, \boldsymbol{\lambda}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_i \lambda_i y_i(\boldsymbol{w^T x_i} - b) + \lambda_i$$

$$\nabla_{(\boldsymbol{w},b)} L(\boldsymbol{w}, b, \boldsymbol{\lambda}) = \begin{bmatrix} \boldsymbol{w} - \sum_i \lambda_i y_i \boldsymbol{x}_i \\ \sum_i \lambda_i y_i \end{bmatrix}$$

KKT Conditions:

$$\mathbf{w}^* - \textstyle\sum_i \lambda_i^* y_i x_i = 0, \quad \sum_i \lambda_i^* y_i = 0$$

$$\lambda_i^*(y_i(w^{*T} x_i - b^*) - 1 = 0$$

Returns:

$$\min_{\boldsymbol{w},b,\boldsymbol{\lambda}\geq 0} L(\boldsymbol{w}, b, \boldsymbol{\lambda}) = L_D(\lambda) =$$

$$\sum_i \lambda_i - \frac{1}{2}\sum_{i,j} {}_i\lambda_j y_i y_j \mathbf{x_i^T x_j} \text{ s.t. } \lambda_i \geq 0, \; \textstyle\sum_i \lambda_i y_i = 0$$

- Then the dual function needs to be calculated by taking the infimum of 8.25, which is accomplished by substituting the gradients w.r.t w and b (8.26) inside, resulting in the dual function (8.30a).
- Maximising this function w.r.t. its constraints is then the dual problem for SVM. (8.30)

18. Useful Facts

The matrix resulting from the dot product of between an R^{nxp} matrix and R^{pxn} has at most a rank of p.

19. Homework and Assignments

Given a prediction table, eg:

$p_X(X_1, X_2)$	$X_2 = 0$	$X_2 = 1$
$X_1 = 0$	$p_X(0,0)$	$p_X(0,1)$
$X_2 = 1$	$p_X(1,0)$	$p_X(1,1)$

Calculate Covariance Matrix from Table

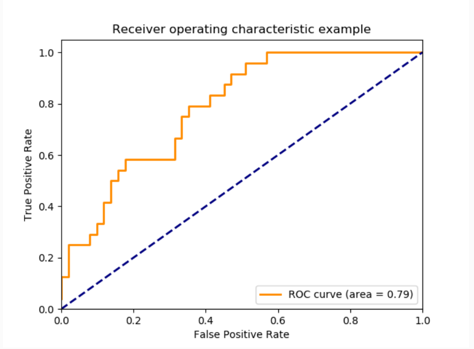
1. Calculate Means for X_1 and X_2: μ_1, μ_2
2. Create X-Matrix, e.g.: $\begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
3. Create p-Matrix: $\begin{bmatrix} \mathrm{p}_X(0,0) & 0 & 0 & 0 \\ 0 & \mathrm{p}_X(1,0) & 0 & 0 \\ 0 & 0 & \mathrm{p}_X(0,1) & 0 \\ 0 & 0 & 0 & \mathrm{p}_X(1,1) \end{bmatrix}$
4. Calculate Covariance: $Cov = XpX^T$

19.1. Classification Analysis

19.1.1. ROC Curve

The ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It shows:

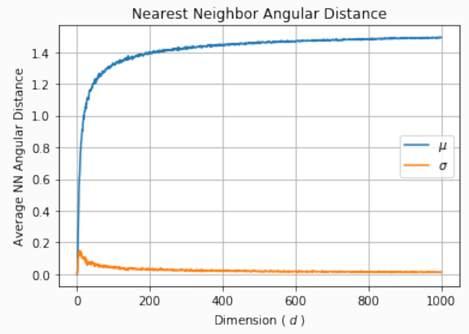
- The relationship between sensitivity and specificity. For example, a decrease in sensitivity results in an increase in specificity.
- Test accuracy; the closer the graph is to the top and left-hand borders, the more accurate the test. Likewise, the closer the graph to the diagonal, the less accurate the test. A perfect test would go straight from zero up the the top-left corner and then straight across the horizontal.
- The likelihood ratio; given by the derivative at any particular cutpoint.



The **Area under the Curve (AUC)** is the integral of the ROC Curve and gives a measure how good the classifier is. An area under the ROC curve of 0.8, for example, means that a randomly selected case from the group with the target equals 1 has a score larger than that for a randomly chosen case from the group with the target equals 0 in 80% of the time. When a classifier cannot distinguish between the two groups, the area will be equal to 0.5 (the ROC curve will coincide with the diagonal). When there is a perfect separation of the two groups, i.e., no overlapping of the distributions, the area under the ROC curve reaches to 1 (the ROC curve will reach the upper left corner of the plot).

19.2. Curse of Dimensionality

- The angular distance between 2 randomly sampled vectors increases with dimension d of the sample space.
- Convergence to $\frac{\pi}{2}$ implies that two randomly sampled vectors are orthogonal to each other in d-dimensional apsce for $d \gg n$.
- Convergence to $\frac{\pi}{2}$ also implies that most samples are concentrated in the 'corners' of the d-dimensional cube $[-1, 1]^d$, i.e. in high dimension, the corners occupy most of the space.
- This convergence also means that 2 randomly sampled vectors are increasingly equidistant (in termas of angular distance) from their respective nearest neighbors in high dimensional space.
- Because the samples are increasingly equidistant from each other, this means that distance-based classifiers (e.g. k-Nearest Neighbors) cannot be used on such data in high-dimensional space.
- Increasing the sample size n decreases the average angular distance between neighbouring vectors in a d-dimensional feature space. The rate of decrease, however, decreases with increasing n.



19.3. Logistic Regression

- With big datasets, standard gradient descent could lead to `Memory Error`
- Use *stochastic gradient descent* instead: Train over epochs instead:
- Each epoch, the training set is divided randomly into equal size subsets (=minibatch). Then the gradient of each subset is calculated and applied only to the samples in the subset
- A epoch is finished when the gradient step was performed on each subset

19.4. Principal Component Analysis

Removing the first n columns from U_k can have different effects on classification:

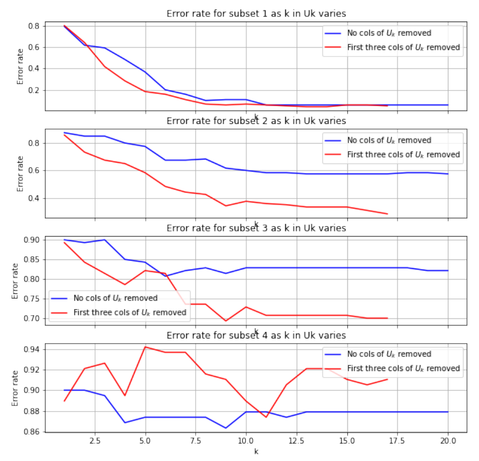
- **Decreased Error Rate**: This may be because even though the first n components capture more variance in the samples, perhaps the other components are better at separating samples by labels, allowing KNN to correctly classify samples (Subset 1 in top plot and second plot)
- **No Effect on Error Rate**: This may be because the first three principal components are as good at separating samples by labels compared to other principal components (Subsets 2+3 in top plot and third plot)
- **Increase Error Rate**: This may be because the first three principal components are better at separating samples by labels compared to the other principal components (Subset 4 top plot and bottom plot)

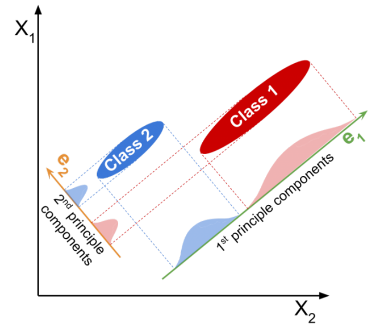
19.4.1. How to choose k?

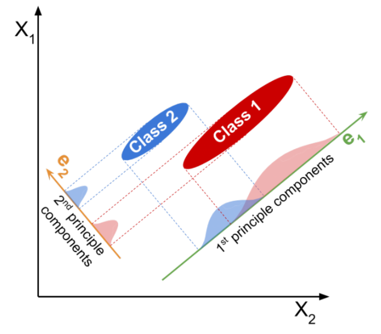
Assuming that $\mathbf{X} \in \mathbb{R}^{p \times N}$ is the centered data matrix and $\mathbf{P} = \mathbf{U}_k \mathbf{U}_k^\top$ is the projector onto the k-dimensional principal subspace, the dimension k is chosen such that the fraction of overall energy contained in the projection error does not exceed ϵ, i.e.

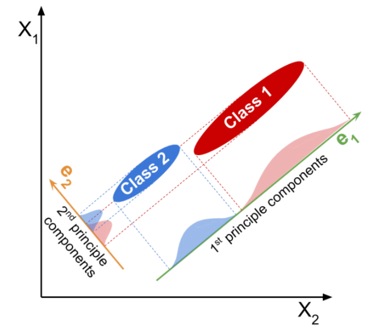
$$\frac{\|\mathbf{X} - \mathbf{P}\mathbf{X}\|_F^2}{\|\mathbf{X}\|_F^2} = \frac{\sum_{i=1}^{M} \|\mathbf{x}_i - \mathbf{P}\mathbf{x}_i\|^2}{\sum_{i=1}^{N} \|\mathbf{x}_i\|^2} \leq \epsilon,$$

where ϵ is usually chosen to be between 0.01 and 0.2. Energy is not always the best way to measure useful information, e.g. when images differ in brightness (=No use full information)


Error rate for subset 1 as k in Uk varies
Error rate for subset 2 as k in Uk varies
Error rate for subset 3 as k in Uk varies
Error rate for subset 4 as k in Uk varies
Error rate
k
No cols of U_k removed
First three cols of U_k removed


X1
X2
Class 1
Class 2
e1
e2
1st principle components
2nd principle components


X1
X2
Class 1
Class 2
e1
e2
1st principle components
2nd principle components


X1
X2
Class 1
Class 2
e1
e2
1st principle components
2nd principle components