SC1_Proj

Alessio

13 January 2020

Description of the dataset and problem

TODO: describe

```
data <- read_csv("../data/data.csv")
```

```
## Warning: Missing column names filled in: 'X33' [33]
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   diagnosis = col_character(),
##   X33 = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Warning: 569 parsing failures.
## row col   expected    actual               file
##   1  -- 33 columns 32 columns '../data/data.csv'
##   2  -- 33 columns 32 columns '../data/data.csv'
##   3  -- 33 columns 32 columns '../data/data.csv'
##   4  -- 33 columns 32 columns '../data/data.csv'
##   5  -- 33 columns 32 columns '../data/data.csv'
## ... ... .......... .......... ..................
## See problems(...) for more details.
```

Dataset Preprocessing Visualisation and Exploration

```
colSums(is.na(data))
```

```
##                      id               diagnosis             radius_mean
##                       0                       0                       0
##            texture_mean          perimeter_mean               area_mean
##                       0                       0                       0
##         smoothness_mean        compactness_mean          concavity_mean
##                       0                       0                       0
##     concave points_mean           symmetry_mean  fractal_dimension_mean
##                       0                       0                       0
##               radius_se              texture_se            perimeter_se
##                       0                       0                       0
##                 area_se           smoothness_se          compactness_se
##                       0                       0                       0
##            concavity_se       concave points_se             symmetry_se
##                       0                       0                       0
##    fractal_dimension_se            radius_worst           texture_worst
##                       0                       0                       0
##         perimeter_worst              area_worst        smoothness_worst
##                         0                       0                       0
##       compactness_worst         concavity_worst    concave points_worst
##                         0                       0                       0
##          symmetry_worst fractal_dimension_worst                     X33
##                         0                       0                     569
```

```
data %<>% mutate_at(vars(diagnosis), factor)
```

```
train <- data %>% sample_frac(0.8)
test <- anti_join(data,train, by='id')
```

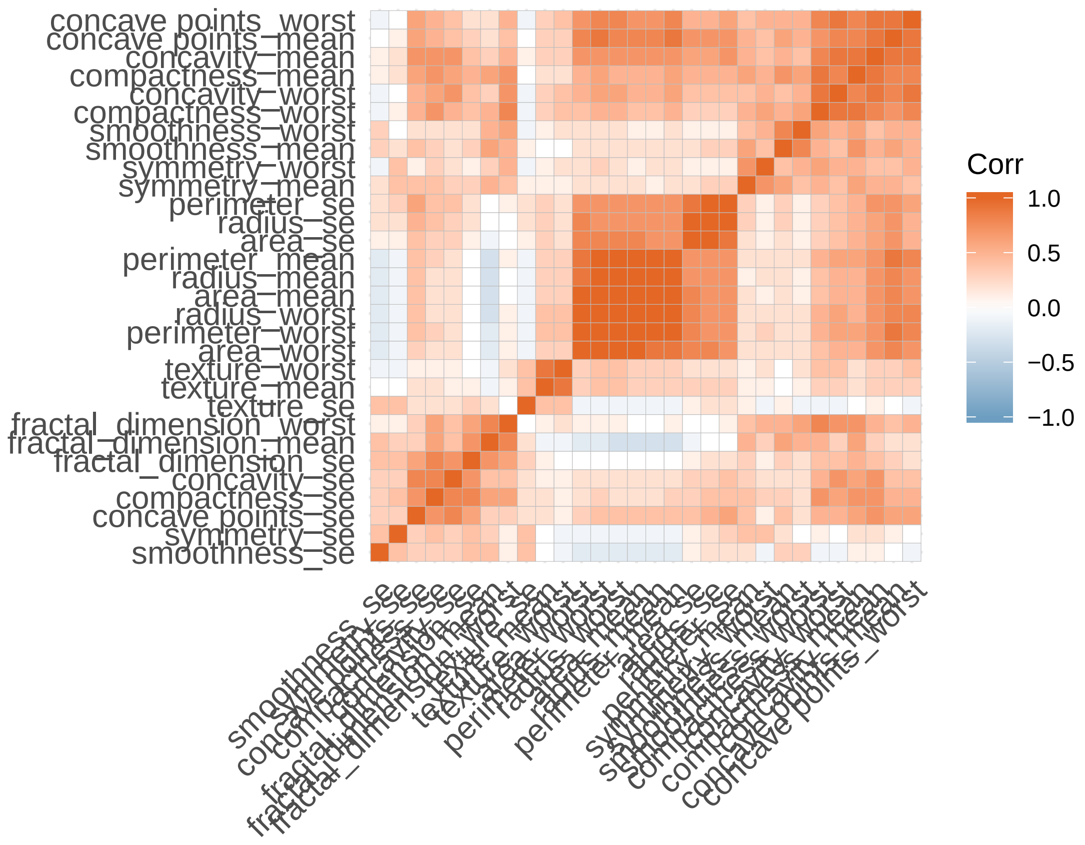
```
data %<>%
  dplyr::select(-c(id, X33))
train %<>%
  dplyr::select(-c(id, X33))
test %<>%
  dplyr::select(-c(id, X33))
```

```
training_data <- train[2:dim(train)[2]]
training_classes <- train[1]

test_data <- test[2:dim(test)[2]]
test_classes <- test[1]
```

```
sum(is.na(data))
```

```
## [1] 0
```

```
corr <- data[,-1] %>%
          cor() %>%
           round(1)
ggcorrplot(corr,
           hc.order = TRUE,
           colors = c("#6D9EC1", "white", "#E46726"),
           ggtheme = ggplot2::theme_minimal)
```



- Class Frequencies
- Density
- box plots

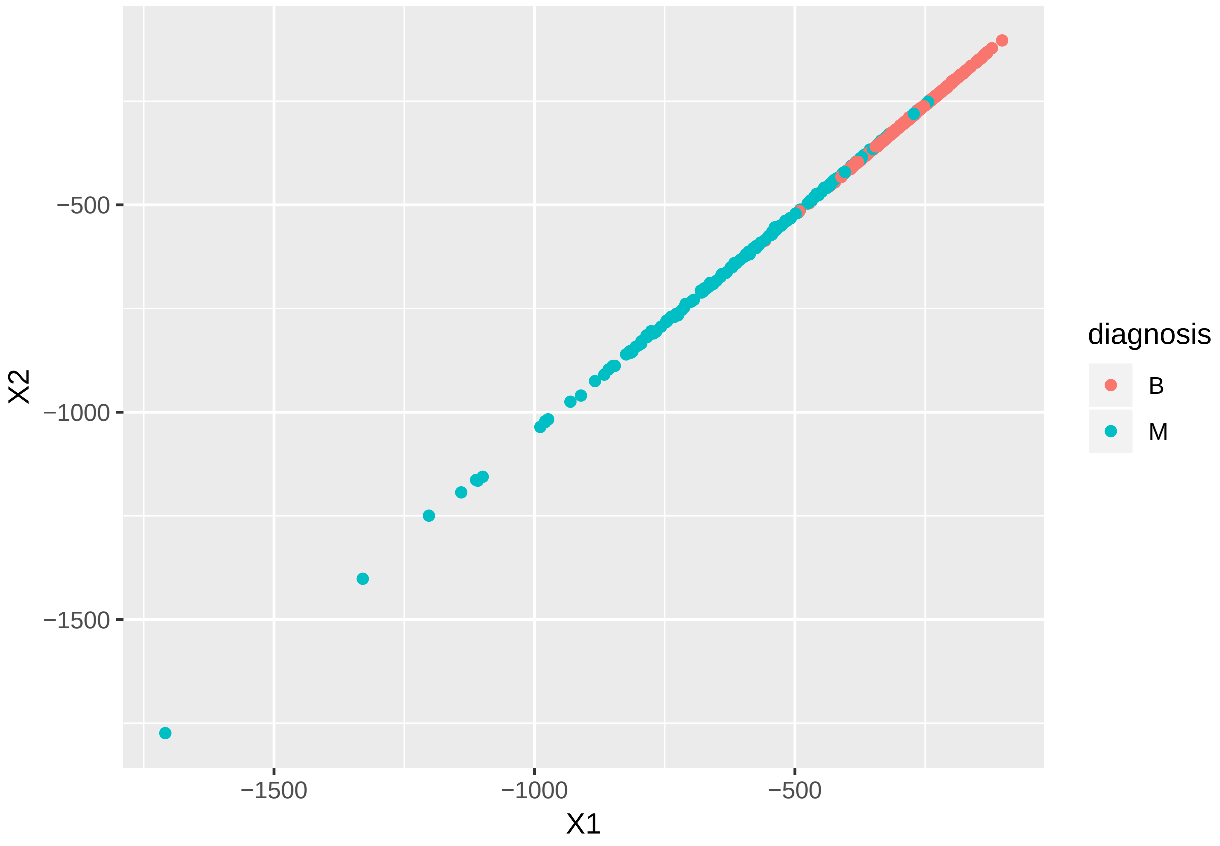
Dimensionality Reduction and Feature Selection

- PCA Code

```
normalise_z <-  function(X){
  mean_cols <- colMeans(X)
  sd_cols <- apply(X, 2, sd)
  mean_normalised_X <- t(apply(X, 1, function(x){x - mean_cols}))
  normalised_X <- t(apply(mean_normalised_X, 1, function(x){x / sd_cols}))
  return(normalised_X)
}


pca <- function(X, number_components_keep) {
  normalised_X <- normalise_z(X)

  corr_mat <- t(normalised_X) %*% normalised_X

  eigenvectors <- eigen(corr_mat, symmetric=TRUE)$vectors

  reduced_data <-  X %*% eigenvectors[,1:number_components_keep]
  relevant_eigs <-  eigenvectors[,1:number_components_keep]
  returnds <- list(reduced_data, relevant_eigs)
  names(returnds) <- c("reduced_data", "reduction_matrix")
  return(returnds)
}
```

Apply to dataset

```
pca_result <- pca(as.matrix(training_data), 2)
pca_reduced_training_data <- data.frame(cbind(pca_result$reduced_data, training_classes))

ggplot(data=pca_reduced_training_data, aes(x=X1, y=X2)) + geom_point(aes(colour=diagnosis))
```



tSNE

TODO: try different perplexity parameters

{r} #reduced_training_data <- tsne::tsne(training_data) # #reduced_train <- data.frame(cbind(reduced_training_data, training_classes)) #ggplot(data=reduced_training_data, aes(x=X1, y=X2)) + geom_point(aes(co

- Correlation Feature Selection
- LDA

Classification

To solve the problem of finding a SVM like classifier for non-separable data we must permit a certain number of points to violate the boundaries set however this number and the amount they violate the constraints by must be as small as possible. To formulate this we introduce a variable ϵ_i for each data point into the objective functions and the constraints leading to the optimisation problem:

$$\min_{w,\epsilon_i} \frac{1}{2} w^T w + C \sum_{i=0}^{n} \epsilon_i$$
$$\text{such that } w \cdot x_i + b + \epsilon_i > 1 \text{ if } y_i = 1$$
$$\text{and } w \cdot x_i + b + \epsilon_i < -1 \text{ if } y_i = -1$$

Note that we have swapped the sign of the b term in the equation for the hyperplane because I implemented it this way before realising they were different and am lazy.

As the above problem is convex (as it is quadratic) and Slater's condition holds then strong duality holds and we can take the Lagrangian of the optimisation problem and consider the result of the KKT conditions. By doing so we can reformulate the optimisation problem as the dual problem:

$$\min_{\lambda} \frac{\overline{\lambda} X X^T \overline{\lambda^T}}{4} + \lambda^T \mathbf{1}$$
$$\text{such that } 0 \leq \lambda_i \leq C$$
$$\text{and } \sum_{i}^{n} \lambda_i y_i = 0$$

where

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \text{ and } \overline{\lambda} = [\lambda_1 \cdot y_i, ..., \lambda_n \cdot y_n] \text{ and } \mathbf{1} = [1, ..., 1] \in \mathbb{R}^{\ltimes}$$

As before we have to massage this optimisation problem into one that can be solved using `solve.QP`. In this formulation

$$d = \mathbf{1}$$

and

$$D = \begin{pmatrix} y_1 & 0 & ... & 0 \\ 0 & y_2 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & y_n \end{pmatrix} X X^T \begin{pmatrix} y_1 & 0 & ... & 0 \\ 0 & y_2 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & y_n \end{pmatrix}$$

A and b_0 require slightly more manipulation this time around with

$$A = \begin{pmatrix} y_1 & y_2 & ... & y_n \\ & & I & \\ & & -I & \end{pmatrix}$$

and

$$b_0 = \begin{pmatrix} 0 \\ \mathbf{0} \\ -\boldsymbol{C} \end{pmatrix}$$

where

$$\mathbf{0} = [0, ..., 0]^T \in \mathbb{R}^n$$

and

$$\boldsymbol{C} = [C, \dots, C]^T \in \mathbb{R}^n$$

The code for this applied to the non-separable data can be found below.

```r
C <- 1

X <- as.matrix(combined_class)[,1:2]
y <- as.matrix(combined_class)[,3]
Dmat2 <- diag(y) * X %*% t(X) %*% diag(y)
diag(Dmat2) <- diag(Dmat2) + 1e-11
dv2 <- rep(1, 30)

A2 <- rbind( y,diag(30))
A2 <- rbind(A2, -1*diag(30))

bv2 <- c(c(0), rep(0, 30), rep(-C, 30) )
model <- solve.QP(Dmat2, dv2, t(A2), bv2, meq = 1)
```

In order to recover w and b from λ we use the relationship

$$w = \sum_{i=0}^{n-1} \lambda_i x_i^T y_i$$

and

$$b = \text{mean}(\sum_{i=0}^{k} y_i - w \cdot x_i) \, . \, \forall \, i . 0 < \lambda_i < C$$

Which can be made as functions in R as so:

```r
calculate_b <- function(w, X, y, a, C) {
  ks <- sapply(a, function(x){return(x > 0 && x < C)})
  indices <- which(ks)
  sum_bs <- 0
  for(i in indices) {
    sum_bs <- sum_bs + (y[i] - w %*% X[i,])
  }
  return(sum_bs / length(indices))
}


recover_w <- function(a, y, X){
  colSums(diag(a) %*% diag(y) %*% X)
}
```

We can see the results of using the dual regression below

```r
soft_margin_svm_plotter <- function(w, b) {
  plotter <- function(x) {
    return(1/w[2]   * -(b + (w[1]*as.numeric(x))))
  }
  return(plotter)
}
```

```r
factor_to_label <- function(x)  {
  if(as.character(x) == "M")  {
    return(1)
  }
  else {
    return(-1)
  }
}
```

```
label_to_factor <- function(x) {
  if(x == 1) {
    return(as.factor("M"))
  }
  else{
    return(as.factor("B"))
  }
}
numeric_test_labels <- apply(test_classes, 1, factor_to_label)
numeric_training_labels <- apply(training_classes, 1, factor_to_label)
```

Use PCA then do SVM

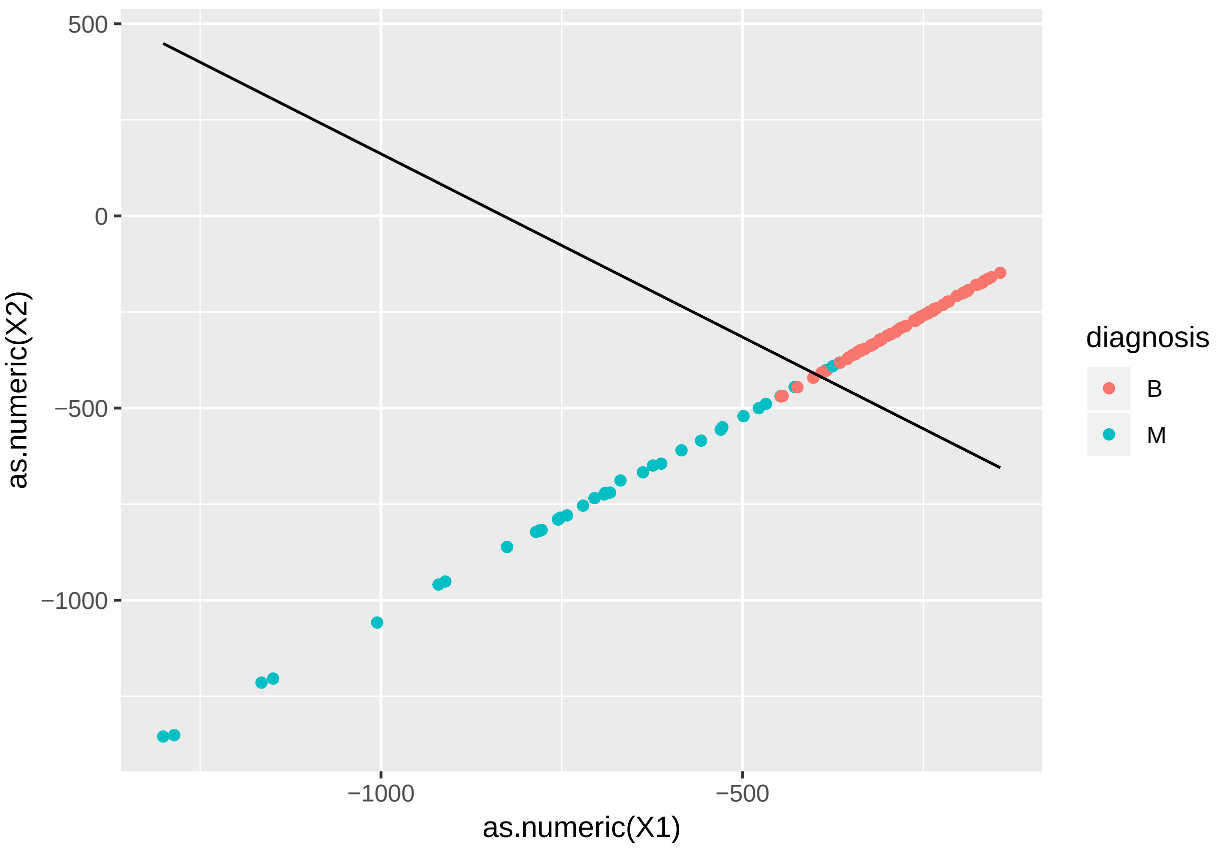
```
model <- svm(X=pca_result$reduced_data,
             classes=numeric_training_labels,
             C=100000, margin_type='soft',
             kernel_function = linear_kernel,
             feature_map = linear_basis_function)


reduced_prediction_fn <- model$prediction_function

pca_reduced_prediction_fn <- function(x) {
  p <- x %*% pca_result$reduction_matrix
  reduced_prediction_fn(t(p))
}
```

```
predictions <- apply(as.matrix(test_data),1, pca_reduced_prediction_fn)
accuracy_calc(numeric_test_labels, predictions)
```

```
## [1] 92.10526
```

```
svm_plotter <- soft_margin_svm_plotter(model$params$w, model$params$b)
embedded_test_data <- data.frame(cbind(as.matrix(test_data) %*% pca_result$reduction_matrix), test_clas

ggplot(embedded_test_data, aes(x=as.numeric(X1), y=as.numeric(X2))) +
  geom_point(aes(colour=diagnosis)) +
  stat_function(fun=svm_plotter)
```

```
## Warning in b + (w[1] * as.numeric(x)): Recycling array of length 1 in array-vector arithmetic is dep
##   Use c() or as.vector() instead.
```



TODO: analyse results

- Naive Bayes
- Logistic Regression
- Lasso

Conclusion

- Evaluation of results
- Discuss outliers