Math 504 Final

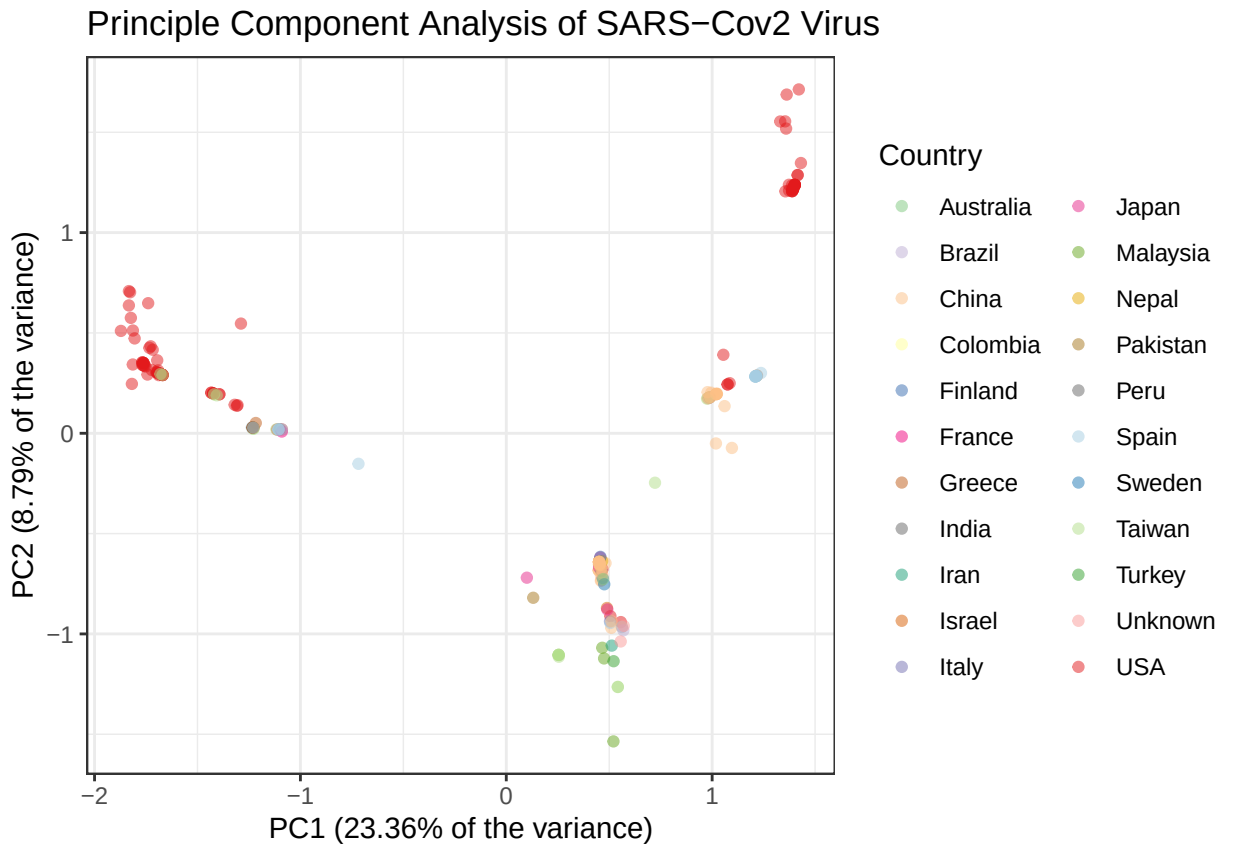
Jeff Gould

5/3/2020

1

```
sequences_load <- data.table::fread("matrix_sequences.csv", header = FALSE,
                                    sep = ",", colClasses = cols(.default = "c"))
countries <- read_csv("matrix_countries.csv", col_names = FALSE)
reference <- data.table::fread("matrix_reference.csv", header = FALSE,
                       sep = ",", colClasses = cols(.default = "c"))

cl <- parallel::makeCluster(10)
parallel::clusterExport(cl = cl, varlist = "reference")
sequences <- t(parallel::parApply(cl = cl, sequences_load, 1,
                                  function(x){as.numeric(x == reference[1,])}))
parallel::stopCluster(cl)

mu <- colMeans(sequences)
sequences_center <- t(apply(sequences, 1, function(x){x-mu}))

n <- ncol(sequences_center)
N <- nrow(sequences_center)

norm <- function(x) sqrt(sum(x^2))
set.seed(123)
V <- matrix(runif(3 * n), nrow = n)
V[,1] = V[,1] / norm(V[,1])
V[,2] = V[,2] / norm(V[,2])


for (i in 1:100) {
  V <- t(sequences_center) %*% (sequences_center %*% V)
  V <- qr.Q(qr(V))
}

PCs <- sequences_center %*% V

plot_data <- data.frame(Country = countries, PC1 = PCs[,1], PC2 = PCs[,2]) %>%
  rename(Country = X1)

variance <- sum(apply(sequences_center, 2, function(x){sum(x^2)}))
lambda_1 <- norm(sequences_center %*% V[,1]) ^2 / (norm(V[,1])^2)
lambda_2 <- norm(sequences_center %*% V[,2]) ^2 / (norm(V[,2])^2)
```

```r
var_captured <- (lambda_1 + lambda_2) / variance * 100

library(RColorBrewer)

ii <- length(unique(countries$X1)) %>% as.numeric()
qual_col_pals = brewer.pal.info[brewer.pal.info$category == 'qual',]
col_vector = unlist(mapply(brewer.pal, qual_col_pals$maxcolors, rownames(qual_col_pals)))

ggplot(data = plot_data, aes(x = PC1, y = PC2, color = Country)) +
  geom_jitter(alpha = 0.5) +
  theme_bw() +
  scale_color_manual(values = col_vector) +
  labs(title = "Principle Component Analysis of SARS-Cov2 Virus",
       x = glue::glue("PC1 ({round(lambda_1 / variance *100,2)}% of the variance)"),
       y = glue::glue("PC2 ({round(lambda_2 / variance *100,2)}% of the variance)"))
```



The first two Principle Components account for 32.14% of the total variance in the data

We see that there are 4 clusters of the SARs Cov-2 when using the first two principle components. The top right cluster is made up entirely of cases from U.S. patients. The far left cluster is made up primarily of patients from the U.S., Taiwan, and Spain, suggesting perhaps a transmission pattern between the three countries (although geographically this doesn't make much sense). The bottom cluster has a lot of cases from China (although tough to see because the cases are so tightly clustered together), and then cases from other Southeast Asia countries and Middle East countries. This suggests that these patients/countries might have been primary transmissions from the initial spread in China. And the last cluster in the middle-right has a mix of U.S., China, Spain, and other European countries

2

a

The dimension of w in this model is m

In general, the log-likelihood model is $\sum_{i=1}^{N} y_i \log(P(y_i = 1) + (1 - y_i) \log(1 - P(y_i = 1))$ So in our example this gives:

$$\log L(w) = \sum_{i=1}^{N} y_i \log\left(\frac{1}{1+\exp(-w^T\phi(x^{(i)}))}\right) + (1-y_i)\log\left(1 - \frac{1}{1+\exp(-w^T\phi(x^{(i)}))}\right) =$$

$$\sum_{i=1}^{N} -y_i \log[1+\exp(-w^T\phi(x^{(i)}))] + (1-y_i)\log\left[\frac{\exp(-w^T\phi(x^{(i)}))}{1+\exp(-w^T\phi(x^{(i)}))}\right] =$$

$$\sum_{i=1}^{N} -y_i \log[1+\exp(-w^T\phi(x^{(i)}))] + (1-y_i)(-w^T\phi(x^{(i)}) - (1-y_i)(\log[1+\exp(-w^T\phi(x^{(i)}))] =$$

$$\sum_{i=1}^{N} (1-y_i)(-w^T\phi(x^{(i)})) - \log[1+\exp(-w^T\phi(x^{(i)}))]$$

b

$B^T = \left(\phi(x^{(1)})\phi(x^{(2)} \dots \phi(x^{(N)})\right)$

i

Let $w = B^T a + z$ for some $a \in \mathbb{R}^N$ and z a constant vector. Then $w = a_1\phi(x^{(1)}) + a_2\phi(x^{(2)}) + \cdots + a_N\phi(x^{(N)}) + z$. So clearly $B^T a \in \text{span}(\phi(x))$, so z is the part of w not in the span(ϕ), so z is orthognal to ϕ Then $\max \log L(w) = \max \sum_{i=1}^{N}(1-y_i)((B^T a + z)^T\phi(x^{(i)})) - \log(1+\exp(-(B^T a + z)^T\phi(x^{(i)})) = \max \sum_{i=1}^{N}(1-y_i)\left[a^T B\phi(x^{(i)}) + z\cdot\phi(x^{(i)})\right] - \log[1+\exp(-(a^T B\phi(x^{(i)}))\exp(-z\cdot\phi(x^{(i)}))]$

Since z is orthgonal to $\phi(x^{(i)})$, $z \cdot \phi(x^{(i)}) = 0$, and the above equation becomes:

$$\max \sum_{i=1}^{N}(1-y_i)\left[a^T B\phi(x^{(i)})\right] - \log[1+\exp(-(a^T B\phi(x^{(i)}))]$$

This is the same result we would achieve if we let $z = 0$. So using a^* that maximes the above equation, $w^* = B^T a^*$ maximizes $\log L(w)$

ii

$$B\phi(x^{(i)}) = \begin{pmatrix} \phi(x^{(1)}) \\ \phi(x^{(2)}) \\ \dots \\ \phi(x^{(N)}) \end{pmatrix} \phi(x^{(i)}) = \begin{pmatrix} \phi(x^{(1)})\cdot\phi(x^{(i)}) \\ \phi(x^{(2)})\cdot\phi(x^{(i)}) \\ \dots \\ \phi(x^{(N)})\cdot\phi(x^{(i)}) \end{pmatrix} = k^{(i)}$$

where $k^{(i)}$ is the ith column of an $N \times N$ matrix K, where each $K_{jl} = \phi(x^{(j)}) \cdot \phi(x^{(l)})$

Thus:

$$\max \sum_{i=1}^{N}(1-y_i)\left[a^T B\phi(x^{(i)})\right] - \log[1+\exp(-(a^T B\phi(x^{(i)}))] =$$

$$\max \sum_{i=1}^{N} (1 - y_i) \left[a^T k^{(i)} \right] - \log[1 + \exp(-a^T k^{(i)})]$$

and solving the above equation will give us a^*

iii

$w^* = B^T a^*$, then

$$w^{*T}\phi(x) = (B^T a^*)^T \phi(x) = (a^*)^T (B\phi(x)) = (a^*)^T \begin{pmatrix} \phi(x^{(1)}) \\ \phi(x^{(2)}) \\ \dots \\ \phi(x^{(N)}) \end{pmatrix} \phi(x) = (a^*)^T \begin{pmatrix} \phi(x^{(1)})^T \phi(x) \\ \phi(x^{(2)})^T \phi(x) \\ \dots \\ \phi(x^{(N)})^T \phi(x) \end{pmatrix} = (a^*)^T \tilde{k}$$

where $\tilde{k} = \begin{pmatrix} \phi(x^{(1)})^T \phi(x) \\ \phi(x^{(2)})^T \phi(x) \\ \dots \\ \phi(x^{(N)})^T \phi(x) \end{pmatrix}$

And so,

$$P(y = 1|w^*, \phi(x)) = \frac{1}{1 + \exp[-(w^*)^T \phi(x)]} \Rightarrow P(y = 1|a^*, \phi(x)) = \frac{1}{1 + \exp[-(a^*)^T \tilde{k}]}$$

c

Replacing equations (1) and (2) with equations (4) and (3) can be advantage when m, the dimension of $\phi(x)$ is very large and N is a smaller dimension, that is when $m >> N$

d

i

$$\log L(w) = \sum_{i=1}^{N} (1 - y_i)(-w^T \phi(x^{(i)})) - \log(1 + \exp(-w^T \phi(x^{(i)})) - \lambda ||w||^2 =$$

$$\sum_{i=1}^{N} (1 - y_i)(-w \cdot \phi(x^{(i)})) - \log(1 + \exp(-w \cdot \phi(x^{(i)})) - \lambda ||w||^2$$

To show the above is concave, we simply need to show $(-w \cdot \phi(x^{(i)})) - \log(1 + \exp(-w \cdot \phi(x^{(i)}))$ is concave and that $-\lambda ||w||^2$ is concave, as the sum of concave functions is concave.

Since $\lambda ||w||^2$ is a polynomial of degree 2, it is clearly convex with respect to w, and thus $-\lambda ||x||^2$ is concave.

$$\nabla_w \left[(-w \cdot \phi(x^{(i)})) - \log(1 + \exp(-w \cdot \phi(x^{(i)})) \right] =$$

$$\nabla_w = -\phi(x^{(i)}) + \phi(x^{(i)}) \left[\frac{\exp(-w \cdot \phi(x^{(i)}))}{1 + \exp(-w \cdot \phi(x^{(i)}))} \right]$$

$$H_w = -\phi(x^{(i)}) \phi(x^{(i)})^T \left[\frac{\exp(-w \cdot \phi(x^{(i)}))}{(1 + \exp(\phi(x^{(i)})))^2} \right]$$

Since $\phi(x^{(i)})\phi(x^{(i)})^T = \phi(x^{(i)}) \cdot \phi(x^{(i)}) \geq 0 \forall x^{(i)}$, $\exp(-w \cdot \phi(x^{(i)})) > 0$ and $(1 + \exp(\phi(x^{(i)})))^2 > 0$, then $H_w \leq 0$, therefore H_w is negative semi-definite and $(-w \cdot \phi(x^{(i)})) - \log(1 + \exp(-w \cdot \phi(x^{(i)}))$ is concave.

Therefore, $\sum_{i=1}^{N} (1 - y_i)(-w \cdot \phi(x^{(i)})) - \log(1 + \exp(-w \cdot \phi(x^{(i)})) - \lambda ||w||^2$ is a concave function.

ii

$$\lambda||w||^2 = \lambda||(B^T a)||^2 = \lambda[(B^T a)^T B^T a] = \lambda[a^T B B^T a] = \lambda a^T K a$$

Using results from equation (3), this gives $\log L(w) = \sum_{i=1}^{N}(1-y_i)(-w^T\phi(x^{(i)})) - \log(1+\exp(-w^T\phi(x^{(i)})) - \lambda\|w\|^2 =$

$$\sum_{i=1}^{N}(1-y_i)(-a^T k^{(i)}) - \log(1+\exp(-a^T k^{(i)})) - \lambda a^T K a$$

3

a

$x^{(i)} \in \mathbb{R}^2$, and for $x \in \mathbb{R}^2$, define $\phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)$ $x, x' \in \mathbb{R}^2$

$$\phi(x)\cdot\phi(x') = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, \sqrt{2}x_1x_2, x_2^2)\cdot(1, \sqrt{2}x_1', \sqrt{2}x_2', x_1'^2, \sqrt{2}x_1'x_2', x_2'^2) =$$
$$1 + 2x_1x_1' + 2x_2x_2' + x_1^2x_1'^2 + 2x_1x_1'x_2x_2' + x_2^2x_2'^2$$

$$(1 + x\cdot x')^2 = (1 + (x_1, x_2)\cdot(x_1', x_2'))^2 = (1 + x_1x_1' + x_2x_2')^2 =$$
$$1 + x_1x_1' + x_2x_2' + x_1x_1' + x_1^2x_1'^2 + x_1x_1'x_2x_2' + x_2x_2' + x_2x_2'x_1x_1' + x_2^2x_2'^2 =$$
$$1 + 2x_1x_1' + 2x_2x_2' + x_1^2x_1'^2 + 2x_1x_1'x_2x_2' + x_2^2x_2'^2 = \phi(x)\cdot\phi(x')$$

b

If $K(x,x') = (1+x\cdot x')^3 = (1+x_1x_1'+x_2x_2')^3 = 1+3x_1x_1'+3x_2x_2'+6x_1x_1'x_2x_2'+3x_1^2x_1'^2+3x_2^2x_2'^2+3x_1x_1'x_2^2x_2'^2+3x_1^2x_1'^2x_2x_2'+x_1^3x_1'^3+x_2^3x_2'^3$, then the feature vectors are: $\phi(x) = (1, \sqrt{3}x_1, \sqrt{3}x_2, x_1^2, \sqrt{6}x_1x_2, \sqrt{3}x_2^2, \sqrt{3}x_1^2x_2, \sqrt{3}x_1x_2^2, x_1^3, x_2^3)$ and $\phi(x') = (1, \sqrt{3}x_1', \sqrt{3}x_2', x_1'^2, \sqrt{6}x_1'x_2', \sqrt{3}x_2'^2, \sqrt{3}x_1'^2x_2', \sqrt{3}x_1'x_2'^2, x_1'^3, x_2'^3)$

c

$\nabla_a = \sum_{i=1}^{N}\nabla_a(1-y_i)(-a^T k^{(i)}) - \nabla_a[\log(1+\exp(-a^T k^{(i)}))] - 2\lambda K a$

```
nn_data <- read_delim("nn.txt", delim = " ")
set.seed(123)
j <- 600
nn_sample <- nn_data %>% sample_n(j)
X <- nn_sample[,1:2] %>% data.matrix()

y <- nn_sample[,3] %>% data.matrix()

a <- matrix(rnorm(j), nrow = j)

K <- apply(X, 1, function(x){apply(X,1,function(y){(1 + sum (x * y))^2})})

lambda <- 10

HessL <- function(a, K, lambda){
  N <- length(a)
```

```
  n <- nrow(K)

  hess <- matrix(0, nrow = N, ncol = N)
  e_term <- exp(-K %*% a)

  for (i in 1:n) {
    X_i <- K[i,]
    hess <- hess - X_i %*% t(X_i) * e_term[i] / ((1 + e_term[i])^2)
  }

  hess <- hess - lambda *2 * K

  return(hess)
}

gradL <- function(a, K, y, lambda){
  N <- length(a)
  n <- nrow(K)

  grad <- rep(0, N)
  e_term <- exp(-K %*% a)

  # for (i in 1:n) {
  #   X_i <- K[i,]
  #   grad <- grad + X_i * e_term[i] / (1 + e_term[i]) - (1-y[i]) * X_i
  # }
  grad <- rowSums(sapply(c(1:n), function(i){K[i,] *
      e_term[i] / (1 + e_term[i]) - (1-y[i]) * K[i,]}))
  grad <- grad + -2 * lambda * K %*% a
  return(grad)
}

logL <- function(a,K,y, lambda){
  return(sum(-(1-y) * (K %*% a) - log(1 + exp(-K %*% a))) - lambda * t(a) %*% K %*% a)
}

norm <- function(x){sqrt(sum(x^2))}

Newton <- function(a,K,y, lambda){

  g <- gradL(a = a,K = K,y = y, lambda = lambda)
  i <- 1
  a_star <- matrix(0, nrow = nrow(a))
  while(norm(a-a_star) > 5E-4) {
    a <- a_star
    h <- HessL(a = a, K = K, lambda = lambda)
    I <- diag(nrow = nrow(h))
    hm <- 0

    # make sure we can invert h before we try,
    # modify the Hessian if needed
    while (kappa(h + hm*I) > 1E15) {
      hm <- 2*(hm + .1)
```

```
    }
    h <- h + hm*I

    g <- gradL(a = a,K = K,y = y, lambda = lambda)
    d <- -solve(h,g)


    cL <- logL(a = a, K = K, y = y, lambda = lambda)

    cat("iteration", i, "like-", cL, "\n")

    s <- 1
    while(cL > logL(a + s*d,K = K, y = y, lambda = lambda)){
      s <- s/2
      if(s < .000001){break}
    }
    a_star <- a + s * d
    i <- i + 1

  }
  return(a_star)
}
lamb <- c(1,10,50,100)
a_star <- sapply(lamb, Newton, a = a, K = K, y = y)
```

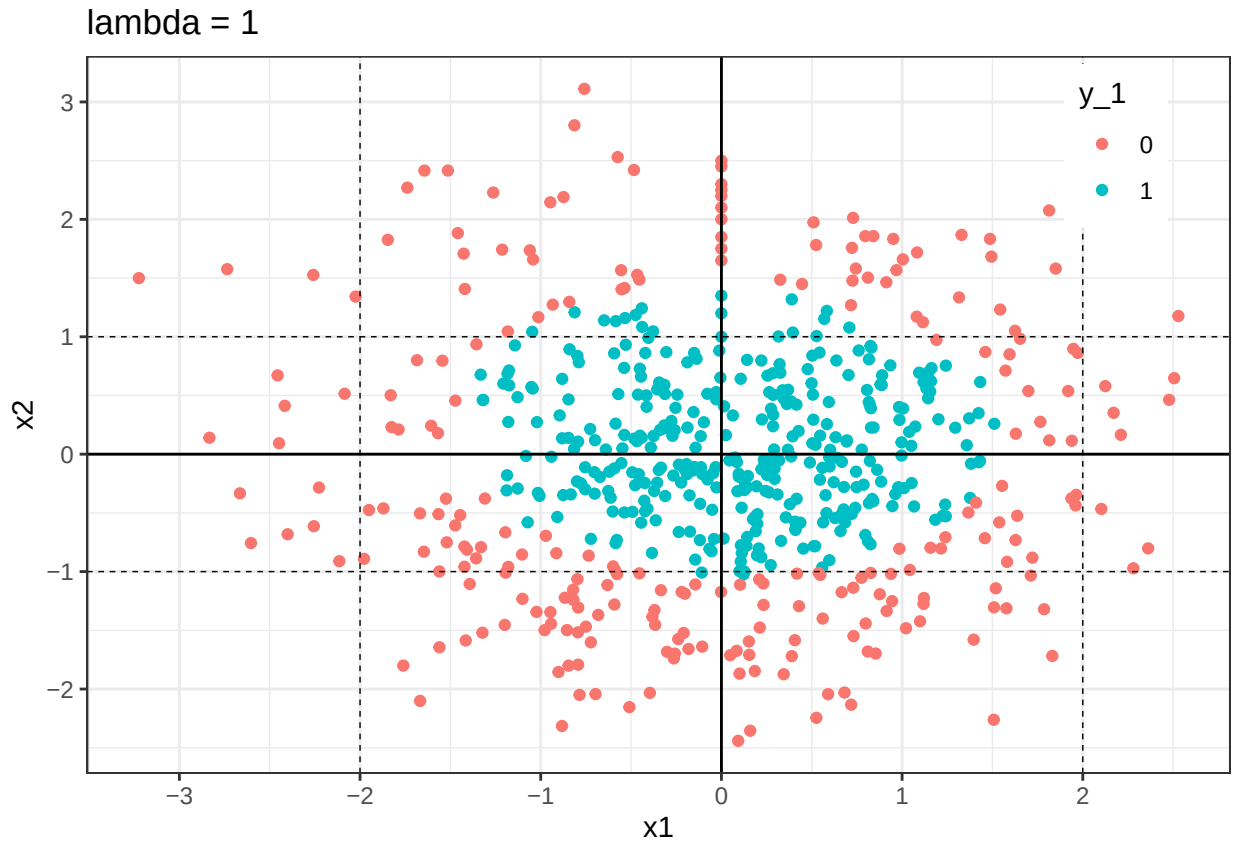
```
## iteration 1 like- -415.8883
## iteration 2 like- -247.9792
## iteration 3 like- -214.1258
## iteration 4 like- -206.7989
## iteration 5 like- -206.3354
## iteration 6 like- -206.3332
## iteration 1 like- -415.8883
## iteration 2 like- -282.5738
## iteration 3 like- -270.4368
## iteration 4 like- -269.8183
## iteration 1 like- -415.8883
## iteration 2 like- -338.3191
## iteration 3 like- -336.851
## iteration 1 like- -415.8883
## iteration 2 like- -360.821
## iteration 3 like- -360.466
```

d

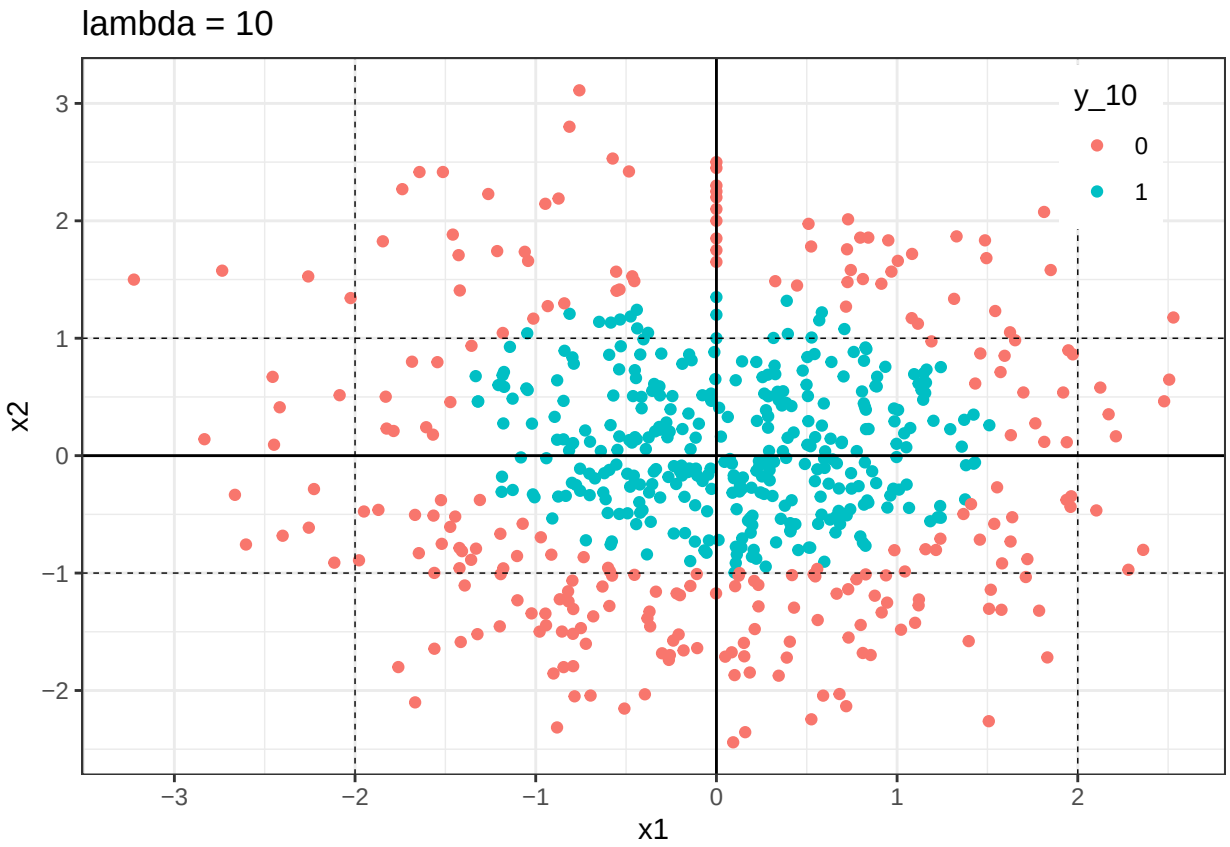
Use a cutoff of $p = 0.5$, and view results for different lambdas on the in sample data:

```
p_hat <- apply(a_star, 2, function(x){1 / (1 + exp(-(t(x) %*% K)))})
y_hat <- matrix(as.numeric(p_hat >=0.5), nrow = j)

test_results <- data.frame(y_1 = y_hat[,1],
                           y_10 = y_hat[,2],
                           y_50 = y_hat[,3],
```

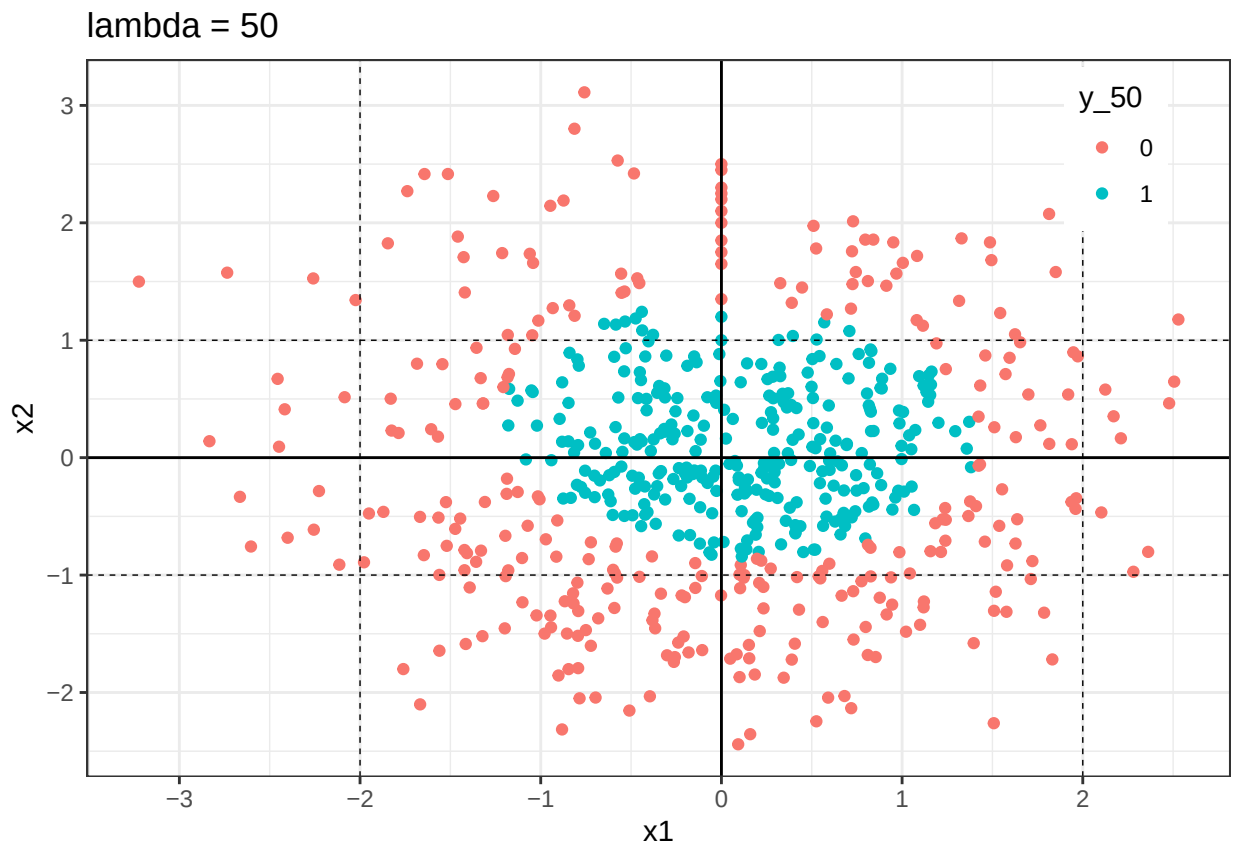
```
                            y_100 = y_hat[,4],
                            x1 = nn_sample$x1,
                            x2 = nn_sample$x2) %>%
  mutate_at(vars(starts_with("y_")), as.factor)

ggplot(data = test_results, aes(x = x1, y = x2, color = y_1)) +
  geom_point() +
  theme_bw() +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = c(-1,1), color = "black", linetype = 2, size = 0.25) +
  geom_vline(xintercept = c(-2,2), color = "black", linetype = 2, size = 0.25) +
  theme(legend.position = c(0.9,0.875)) +
  labs(title = "lambda = 1")
```



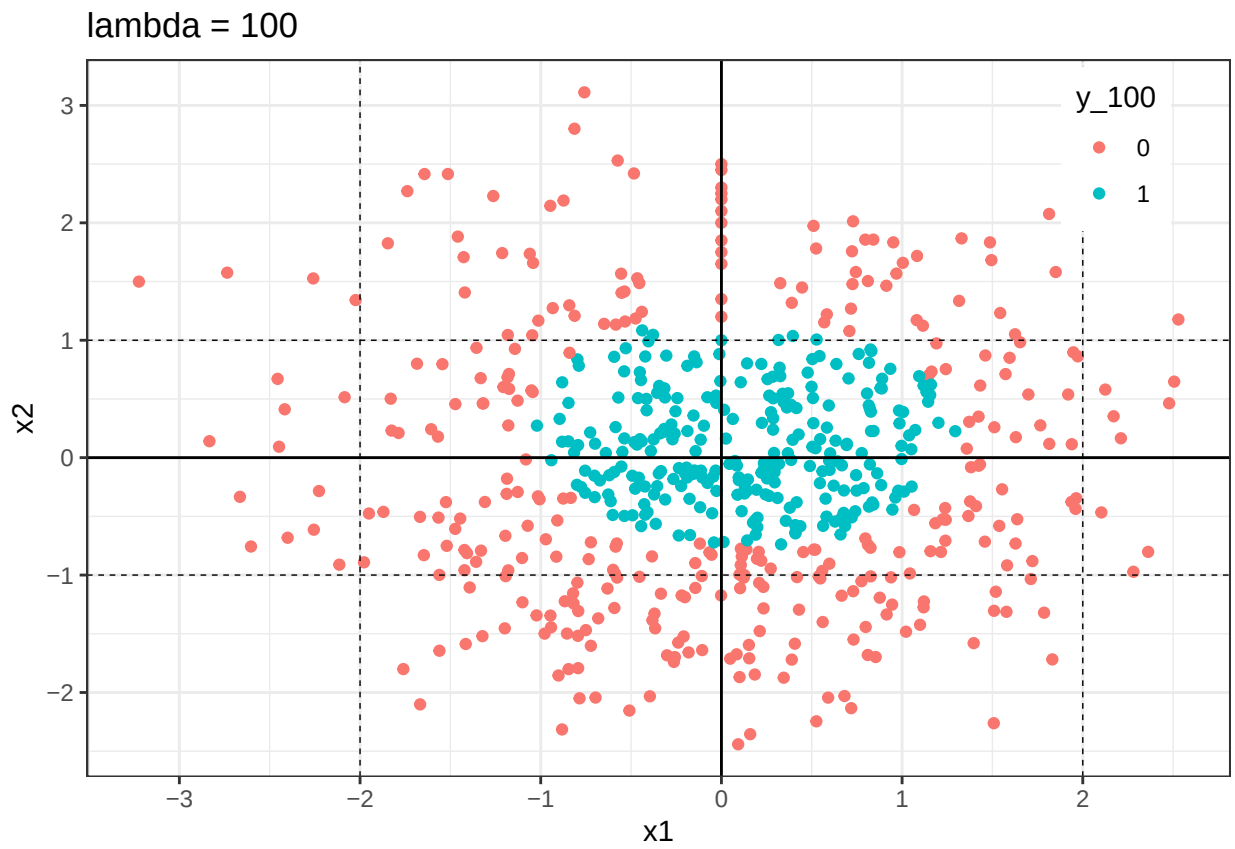

```
ggplot(data = test_results, aes(x = x1, y = x2, color = y_10)) +
  geom_point() +
  theme_bw() +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = c(-1,1), color = "black", linetype = 2, size = 0.25) +
  geom_vline(xintercept = c(-2,2), color = "black", linetype = 2, size = 0.25) +
  theme(legend.position = c(0.9,0.875)) +
  labs(title = "lambda = 10")
```




```
ggplot(data = test_results, aes(x = x1, y = x2, color = y_50)) +
  geom_point() +
  theme_bw() +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = c(-1,1), color = "black", linetype = 2, size = 0.25) +
  geom_vline(xintercept = c(-2,2), color = "black", linetype = 2, size = 0.25) +
  theme(legend.position = c(0.9,0.875)) +
  labs(title = "lambda = 50")
```




```
ggplot(data = test_results, aes(x = x1, y = x2, color = y_100)) +
  geom_point() +
  theme_bw() +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = c(-1,1), color = "black", linetype = 2, size = 0.25) +
  geom_vline(xintercept = c(-2,2), color = "black", linetype = 2, size = 0.25) +
  theme(legend.position = c(0.9,0.875)) +
  labs(title = "lambda = 100")
```



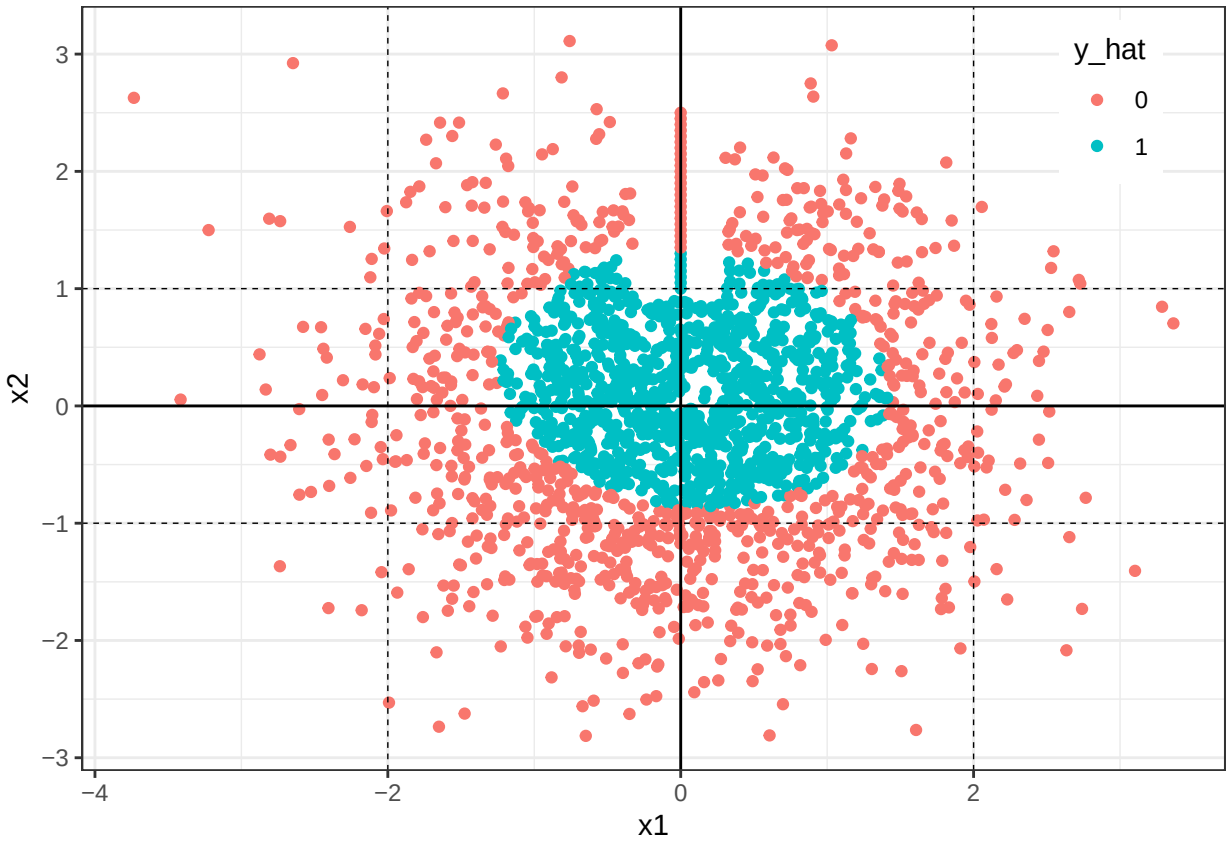
Looking at the results above, we see that the results generated by the kernel machine are more circular than the original image and the results we achieved using a neural net. This is probably due in part to only running our results on about 25% of the data set, as well as limiting our model to two dimensions. We also fail to capture any of the antenna, but this should be expected with only creating two dimensions, as we can only create ellipsoid regression lines.

Looking at the graphs with different lambda parameters, smaller λ's create too large of an ellipse, and $\lambda = 100$ is too restrictive. But $\lambda = 50$ appears to be a good penalty parameter of not restricting too much, but also not being too lax.

Test results on the full data set, again using a cutoff of $p = 0.5$ (includes in and out of sample data)

```
a_star <- a_star[,3]
calc_prob <- function(x){
  x <- matrix(x, nrow = 1)
  p_hat <- 1 / (1 + exp(-t(a_star) %*% apply(nn_sample[,1:2],
                                             1,function(y) (1 + sum(y * x))^2 )))
  return(p_hat)
}

p_hat <- apply(nn_data[,1:2], 1, calc_prob)
y_hat <- as.numeric(p_hat >=0.5)

test_results <- data.frame(y_hat = y_hat,
                           p_hat = p_hat,
                           x1 = nn_data$x1,
                           x2 = nn_data$x2) %>%
```

```
  mutate(y_hat = as.factor(y_hat))

ggplot(data = test_results, aes(x = x1, y = x2, color = y_hat)) +
  geom_point() +
  theme_bw() +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = c(-1,1), color = "black", linetype = 2, size = 0.25) +
  geom_vline(xintercept = c(-2,2), color = "black", linetype = 2, size = 0.25) +
  theme(legend.position = c(0.9,0.875))
```



Using $\lambda = 50$, we see that while we do get a decent ellipse, the center is a bit high, and it is narrow as above. We do capture some of the antenna, but this is just because the antenna is inside of the ellipse region and not because the shape accounts for the antenna

e

If we extended the kernel to $\#{=}K(x, x') = (1 + x \cdot x')^3$, then we might expect to capture more of the antenna shape instead of just an ellipse. Because we only allow a quadratic with our kernal function, we are limited to capturing an elliptical or parabolic shape with our analysis.