# Slides 6

## Problem 2.1

The bernouli likelihood function, defined for random variable $t \in -1, 1$, takes the form:

$$p(t = 1) = p$$
$$p(t = -1) = 1 - p$$

If we now consider a sigmoid function $\sigma(u)$ which gives us a bernouli probability, $p \in [0, 1]$, we then get the following:

$$p(t = 1) = \sigma(u)$$
$$p(t = -1) = 1 - \sigma(u)$$

If we consider the situation where $\sigma(u)$ is the logistic function:

$$p(t = 1) = \frac{1}{1 + e^{-u}}$$
$$p(t = -1) = 1 - \frac{1}{1 + e^{-u}}$$

The sigmoid/logistic function has the special property that in addition to have an output bounded between $[0, 1]$, it's first derivative is also symmetric around 0. This is to say that from $[0, \infty]$, the function moves towards 1 at the exact same rate it moves towards 0 over the range $[0, -\infty]$. This implies the following:

$$\sigma(u) = 1 - \sigma(-u)$$

Using this, we can rewrite our previous pair of equations:

$$p(t = 1) = \frac{1}{1 + e^{-u}}$$
$$p(t = -1) = \frac{1}{1 + e^{u}}$$

Or better still:

$$p(t) = \frac{1}{1 + e^{-tu}}$$

If we then plug the linear model $\boldsymbol{w}^T\phi(x)+b$ in for u, we find exactly the large margin classifier with the logistic loss function:

$$p(t) = \frac{1}{1+\exp\Big\{-t(\boldsymbol{w}^T\phi(x)+b)\Big\}}$$

## Problem 2.2

The MAP estimator is given, as per usual:

$$w_{MAP} = \underset{w}{\text{argmax}}\, p(\boldsymbol{t}|\boldsymbol{w}, X)p(\boldsymbol{w})$$

Our $\boldsymbol{t}$ observed variables are bernouli $t \in -1, 1$, and the linear model we are applying on our data to map it to our bernouli predictions is the logistic function. Our likelihood function $p(\boldsymbol{t}|\boldsymbol{w}, X)$, is therefore exactly what we found in 2.1, while our prior on $\boldsymbol{w}$ is Gaussian. We include only the terms that are dependent on $\boldsymbol{w}$ and assume independence between rows of our design matrix:

$$w_{MAP} = \underset{w}{\text{argmax}} \prod_n \left(\frac{1}{1+\exp\Big\{-t_n(\boldsymbol{w}^T\phi(x_n)+b)\Big\}}\right) * \exp\Big\{-\frac{1}{2}\boldsymbol{w}^T\lambda N\,\boldsymbol{w}\Big\}$$

$$w_{MAP} = \underset{w}{\text{argmax}} \sum_n \Big(-\log\Big(1+\exp\Big\{-t_n(\boldsymbol{w}^T\phi(x_n)+b)\Big\}\Big)\Big) - \frac{\lambda N}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$w_{MAP} = \underset{w}{\text{argmin}} \sum_n \log\Big(1+\exp\Big\{-t_n(\boldsymbol{w}^T\phi(x_n)+b)\Big\}\Big) + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

Because $\log(1+e^x)$ approximates $1+e^x$, minimizing over these two is equivalent, and we have our solution.

# Slides 7

## Problem 2

The probability that a new point belongs to any given mixture k is the joint probability of that data and the latent variable that points to that particu-

lar mixture. We decompose this into the conditional probability of the data given that particular mixture multiplied by the prior on the probability of that mixture. To coerce this into a probability, we normalize by the sum of the probabilities that the point belongs to each mixture k:

$$\Pi_k = \frac{p(x_{n+1}, z_k)}{\sum_i^K p(x_{n+1}|z_i)}$$
$$\Pi_k = \frac{p(x_{n+1}|z_k)p(z_k)}{\sum_i^K p(x_{n+1}|z_i)}$$

We will focus on determining the formula for the numerator, for one particular k, as this is simply repeated for all k's to create a vector of probabilities, and the denominator follows directly from the numerator. The latent variable $z_k$ corresponds to the mean $\mu_k$ and precision $Q_k$ for that particular guassian, so we compute the probability that $x_{n+1}$ came from that gaussian, multiplied by whatever our prior probability was on that gaussian (we could assume this is a):

$$p(x_{n+1}|z_k)p(z_k)$$
$$\frac{exp\Big\{ -\frac{1}{2}(x_{n+1} - \mu_k)^T Q(x_{n+1} - \mu_k)\Big\}}{\sqrt{|2\pi Q^{-1}|}} * p(z_k)$$

## Problem 5

### Robust Regression

Assuming $\nu$ is fixed and applying bayes theorum:

$$p(\eta_n|t_n, \nu) = \frac{p(t_n|\nu, \eta_n)p(\eta_n|\nu)}{p(t_n, \nu)}$$

In robust regression we model the observed data, $\boldsymbol{t}$, as gaussian, but where the gaussian for every observation has a variance derived from a latent variable $\eta_n$ which itself is a random variable picked from a gamma distribution. Because the gamma is a conjugate prior for the precision of the gaussian distribution, we end with a gamme distribution of the form, which we shall provide later in its full, ugly, PDF form, and therefore spare the reader at this

point. Suffice to say at this juncture that we increase the $\alpha$ parameter by a half, and the $\beta$ paramter by half the squared residuals:

$$p(\eta_n|t_n, \nu) \sim Gamma((\nu+1)/2, (\nu + q(t_n - \phi(\boldsymbol{x}_n)^T\boldsymbol{w})^T(t_n - \phi(\boldsymbol{x}_n)^T\boldsymbol{w})/2 - 1))$$

**Logistic and Probit Models**

We will consider a general case of K discrete possible values for $t$, which will easily hold true for our binomial case, and will be shown to be a more general expression of the truncated distribution in the currently-considered binomial case. We can marginalize over $t$:

$$p(z_n|\boldsymbol{w}, \boldsymbol{x}_n) = \sum_k p(z_n|\boldsymbol{w}, \boldsymbol{x}_n, t_k)p(t_k|\boldsymbol{w}, \boldsymbol{x}_n)$$

Expanding the sum over possible values of $t$, and gathering terms, we come to:

$$p(z_n|\boldsymbol{w}, \boldsymbol{x}_n, t_n) = \frac{p(z_n|\boldsymbol{w}, \boldsymbol{x}_n) - \sum_{k\neq n}^{K} p(z_n|\boldsymbol{w}, \boldsymbol{x}_n, t_k)p(t_k|\boldsymbol{w}, \boldsymbol{x}_n)}{p(t_n|\boldsymbol{w}, \boldsymbol{x}_n)}$$

We gather the marginal distributions into their joint distribution as follows:

$$p(z_n|\boldsymbol{w}, \boldsymbol{x}_n, t_n) = \frac{p(z_n|\boldsymbol{w}, \boldsymbol{x}_n) - \sum_{k\neq n}^{K} p(z_n, t_k|\boldsymbol{w}, \boldsymbol{x}_n)}{p(t_n|\boldsymbol{w}, \boldsymbol{x}_n)} \tag{1}$$

We can begin to see some things in the above equation that match our intuition, and similarly match the definition of a truncated distribution, where we have turned a continous distribution function into a piece-wise function that returns 0 over certain intervals. For any $z_n$ that deterministically gives us $t_k$:

$$p(z_n|\boldsymbol{w}, \boldsymbol{x}_n) = p(z_n, t_k|\boldsymbol{w}, \boldsymbol{x}_n)$$

Together with our (1), this gives us the desired result that for any $z_n$ that deterministically gives us $t_k$, where $t_k \neq t_n$:

$$p(z_n|\boldsymbol{w}, \boldsymbol{x}_n, t_n) = 0$$

npThis gives us a probability function that will return 0 for any $z_n$ outside of the range of the $t_n$ it is conditioned upon, and for every other value of $z_n$ will return:

$$p(z_n|\boldsymbol{w}, \boldsymbol{x}_n, t_n) = \frac{p(z_n|\boldsymbol{w}, \boldsymbol{x}_n)}{\int p(t_n|\boldsymbol{w}, \boldsymbol{x}_n, z_n)p(z_n)dz} \tag{2}$$

It is easy to see that through the defined relationship, $t_n = 1[z_n > 0]$, $p(t_n|\boldsymbol{w}, \boldsymbol{x}_n, z_n)$ is deterministic, and therefore marginilizing over the $p(z_n)$ is equal to the probability given by the CDF of $p(z_n|\boldsymbol{w}, \boldsymbol{x}_n)$ of $z_n$ being within the range of $t_n$, which relates back to the truncated distribution over $z_n$, which attempts to scale up the probabilities within the range by the amount of total distributed mass outside the range. We therfore have another way to express this function, which is useful:

$$p(z_n|\boldsymbol{w}, \boldsymbol{x}_n, t_n = 0) \sim \mathit{TruncatedNormal}(\phi(\boldsymbol{x}_n)^T\boldsymbol{w}, \sigma^2, (-\infty, 0])$$

Now that we have seen the whole pictures, we can easily define the probability in our binary casein regards to a PDF g(x), based on scaling the probability of a given $z_n$ by the CDF of $z$ over the defined distribution for $t_n$:

$$p(z_n|\boldsymbol{w}, \boldsymbol{x}_n, t_n = 0) = \frac{g(z_n, \boldsymbol{x}_n, \boldsymbol{w})}{\int_{-\infty}^{0} g(z_n, \boldsymbol{x}_n, \boldsymbol{w})dz}$$
$$p(z_n|\boldsymbol{w}, \boldsymbol{x}_n, t_n = 1) = \frac{g(z_n, \boldsymbol{x}_n, \boldsymbol{w})}{\int_{0}^{\infty} g(z_n, \boldsymbol{x}_n, \boldsymbol{w})dz}$$

We then defin g(x) for each distribution, for the probit, it is a gaussian PDF:

$$g(z_n, \boldsymbol{x}_n, \boldsymbol{w}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ - \frac{(z_n - \phi(\boldsymbol{x}_n)\boldsymbol{w})^2}{2\sigma^2} \right\}$$

And for the logistic, we are given the PDF as follows:

$$g(z_n, \boldsymbol{x}_n, \boldsymbol{w}) = \frac{e^{\phi(\boldsymbol{x}_n)^T\boldsymbol{w}}}{(1 + e^{\phi(\boldsymbol{x}_n)^T\boldsymbol{w}})^2}$$

**Gaussian Mixture Models**

The probability of

$$z_n|\boldsymbol{x}_n, \boldsymbol{\mu}_{1:K}, \boldsymbol{Q}_{1:K}$$

Is the latent variable for each observation, that contains the "responsibility" that each gaussian mixture, K, has over the variable. This is a vector that consists of the following for each K:

$$\Pi_k = \frac{p(x_{n+1}, z_k)}{\sum_i^K p(x_{n+1}|z_i)}$$

### Factor Model

Applying bayes theorum and assuming independence of parameters:

$$p(\boldsymbol{z}_n|\boldsymbol{x}_n, \boldsymbol{w}, \boldsymbol{\Sigma}) = \frac{p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{w}, \boldsymbol{\Sigma})p(\boldsymbol{z}_n, \boldsymbol{w}, \boldsymbol{\Sigma})}{p(\boldsymbol{x}_n, \boldsymbol{w}, \boldsymbol{\Sigma})}$$
$$p(\boldsymbol{z}_n|\boldsymbol{x}_n, \boldsymbol{w}, \boldsymbol{\Sigma}) = \frac{p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{w}, \boldsymbol{\Sigma})p(\boldsymbol{z}_n)}{p(\boldsymbol{x}_n|\boldsymbol{w}, \boldsymbol{\Sigma})}$$

Focusing on the numerator (we can recover the normalizing constant later, as per usual), we see that we have the multiplication of two provided normal distributions, seeing as:

$$\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{w}, \boldsymbol{\Sigma} \sim \mathcal{N}(\boldsymbol{\mu} + \boldsymbol{W}\boldsymbol{z}_n\boldsymbol{\Sigma}) \quad \boldsymbol{z}_n \sim \mathcal{N}(\boldsymbol{0}_R, \boldsymbol{I}_R)$$

We know that the conditional distribution of two gaussians is also gaussian. Bishop provides us with a handy rule for finding the conditional distribution of two gaussians, which we will employ here:

$$\begin{aligned} P &= \boldsymbol{I}_R + \boldsymbol{W}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{W} \\ G &= \boldsymbol{W}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}) + \boldsymbol{I}_R^{-1}\boldsymbol{0}_R \\ &= \boldsymbol{W}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}) \\ p(\boldsymbol{z}_n|\boldsymbol{x}_n, \boldsymbol{w}, \boldsymbol{\Sigma}) &\sim \mathcal{N}(P^{-1}G, P^{-1}) \end{aligned}$$

# Slides 8

## Problem 1

We begin with bayes theorum, and rearrange taking advantage of indepence of parameters and expressing the marginal distribution denominator as the joint distribution of our observed variable with our latent variable, integrated over our latent variable:

$$p(\eta_n|\boldsymbol{t}_n, \boldsymbol{X}_n, \boldsymbol{w}, q) = \frac{p(\boldsymbol{t}_n|\eta_n, X_n, \boldsymbol{w}, q)p(\eta_n, \boldsymbol{X}_n, \boldsymbol{w}, q)}{p(\boldsymbol{t}_n, \boldsymbol{X}_n, \boldsymbol{w}, q)}$$
$$p(\eta_n|\boldsymbol{t}_n, \boldsymbol{X}_n, \boldsymbol{w}, q) = \frac{p(\boldsymbol{t}_n|\eta_n, X_n, \boldsymbol{w}, q)p(\eta_n)}{\int p(\boldsymbol{t}_n, |\eta_n, \boldsymbol{X}_n, \boldsymbol{w}, q)p(\eta_n)d\eta}$$

We will focus first on simplifying our numerator, $p(\boldsymbol{t}_n|\eta_n, X_n, \boldsymbol{w}, q)p(\eta_n)$. We plug in the given distributions for $\boldsymbol{t}_n$ and $/eta_n$ and combine terms:

$$\frac{(\frac{\nu}{2}-1)^{\frac{\nu}{2}}\eta^{\frac{\nu}{2}-1}}{\Gamma(\frac{\nu}{2})}\exp\left\{-\eta(\frac{\nu}{2}-1)\right\}\frac{1}{|2\pi(\eta q)^{-1}\boldsymbol{I}|^{\frac{1}{2}}}\exp\left\{-\frac{\eta q}{2}(\phi(\boldsymbol{X}_n)^T\boldsymbol{w})^T(\phi(\boldsymbol{X}_n)^T\boldsymbol{w})\right\}$$

$$\frac{(\frac{\nu}{2}-1)^{\frac{\nu}{2}}\eta^{\frac{\nu}{2}-1}}{\Gamma(\frac{\nu}{2})(2\pi)^{\frac{1}{2}}\eta^{-\frac{1}{2}}q^{-\frac{1}{2}}}\exp\left\{-\eta(\frac{\nu}{2}-1+\frac{q}{2}(\phi(\boldsymbol{X}_n)^T\boldsymbol{w})^T(\phi(\boldsymbol{X}_n)^T\boldsymbol{w}))\right\}$$

$$\frac{q^{\frac{1}{2}}(\frac{\nu}{2}-1)^{\frac{\nu}{2}}\eta^{\frac{\nu+1}{2}-1}}{\Gamma(\frac{\nu}{2})(2\pi)^{\frac{1}{2}}}\exp\left\{-\eta(\nu+q(\phi(\boldsymbol{X}_n)^T\boldsymbol{w})^T(\phi(\boldsymbol{X}_n)^T\boldsymbol{w})/2-1)\right\}$$

At this point we see that we have found our $\beta$ in the exponent term, and in our $\eta$ term, which is great. We also see that we have a $sqrt\pi$, which we know is equal to $\Gamma(\frac{1}{2})$. Safe to say, that attempting to perform the integration at this point would be a waste of my youth. I will go ahead and skip to the following:

$$C * \eta^{\frac{\nu+1}{2}-1} * \exp\left\{-\eta(\nu+q(\phi(\boldsymbol{X}_n)^T\boldsymbol{w})^T(\phi(\boldsymbol{X}_n)^T\boldsymbol{w})/2-1)\right\}$$

$$\sim Gamma((\nu+1)/2, (\nu+q(t_n-\phi(\boldsymbol{x}_n)^T\boldsymbol{w})^T(t_n-\phi(\boldsymbol{x}_n)^T\boldsymbol{w})/2-1))$$

## Problem 2

We start from the following definition:

$$\int \log p(\boldsymbol{t}, \boldsymbol{\eta}|\boldsymbol{\theta})p(\boldsymbol{\eta}|\boldsymbol{t}, \theta')d\boldsymbol{\eta} + C$$

With C referring to everything that does not depend on $\theta$. Quite luckily, we have already derived the joint distribution of $t_n$ and $\eta$, and we can substitute everything there that does not depend on $\theta$, which is the exponent term, which of course cancels the log:

$$\int -\eta(\nu+q(\phi(\boldsymbol{X}_n)^T\boldsymbol{w})^T(\phi(\boldsymbol{X}_n)^T\boldsymbol{w})/2-1)p(\boldsymbol{\eta}|\boldsymbol{t}, \theta')d\boldsymbol{\eta} + C$$

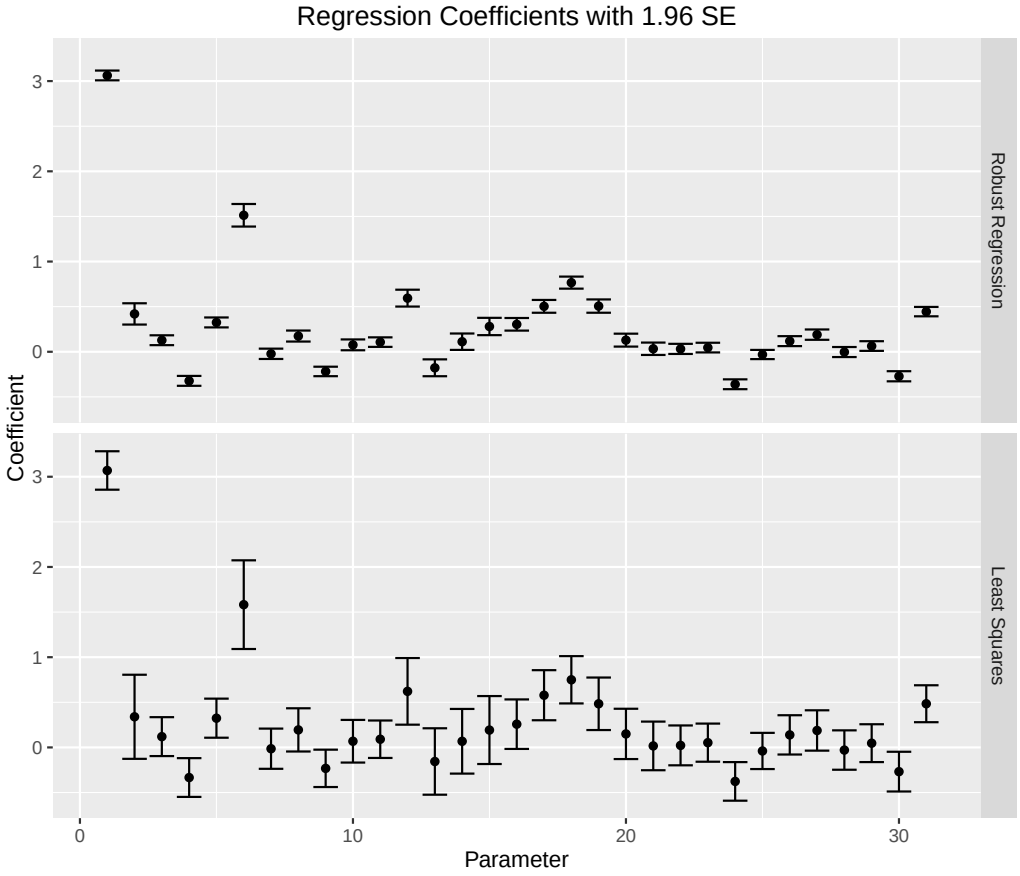
$$\int -(\nu + q(\phi(\boldsymbol{X}_n)^T \boldsymbol{w})^T (\phi(\boldsymbol{X}_n)^T \boldsymbol{w})/2 - 1)\boldsymbol{\eta} p(\boldsymbol{\eta}|\boldsymbol{t}, \theta') d\boldsymbol{\eta} + C$$

$$\int -(\nu + q(\phi(\boldsymbol{X}_n)^T \boldsymbol{w})^T (\phi(\boldsymbol{X}_n)^T \boldsymbol{w})/2 - 1)\mathbb{E}[\boldsymbol{\eta}|\boldsymbol{t}, \boldsymbol{X}, \theta'] + C$$

## Problem 3

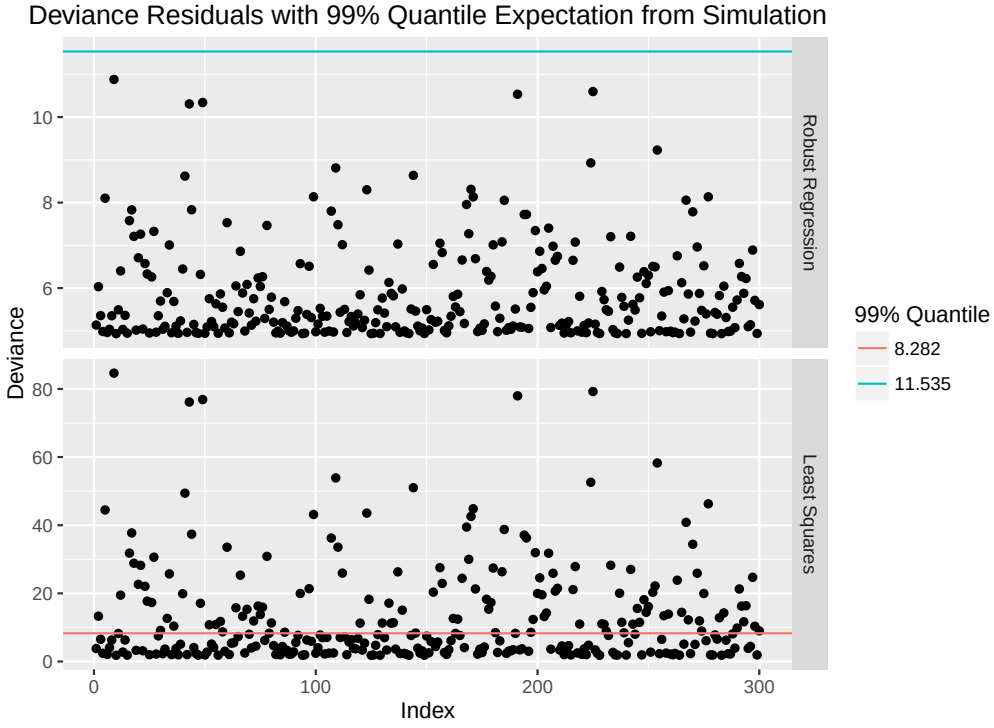
### 1

Code is included in the appendix, here are the juicy plots that show how much more certain we are about our weights in the robust regresssion:



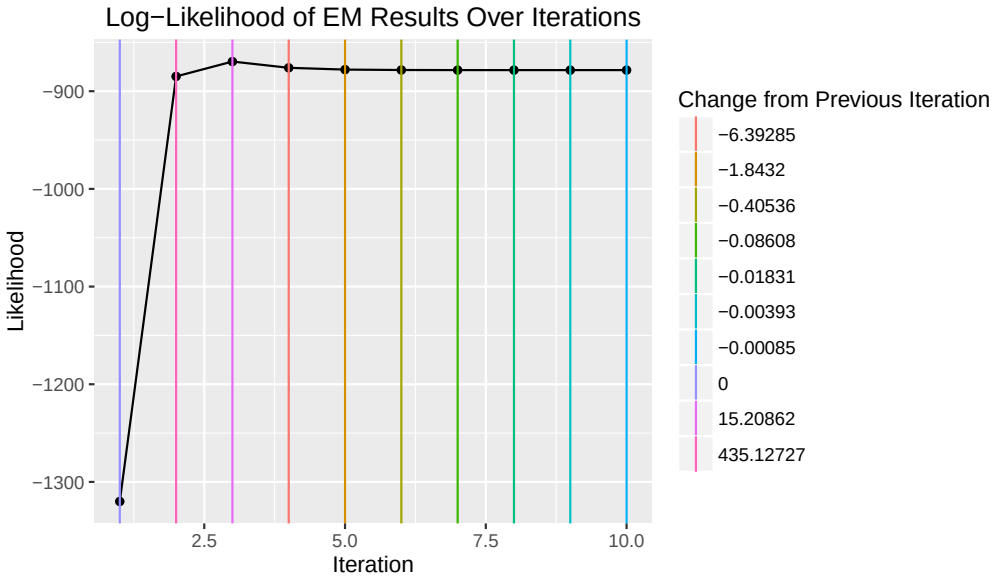
## 2

Here we can see that the data does not at all fit with the OLS assumption of gaussian errors distributed by the sample variance!



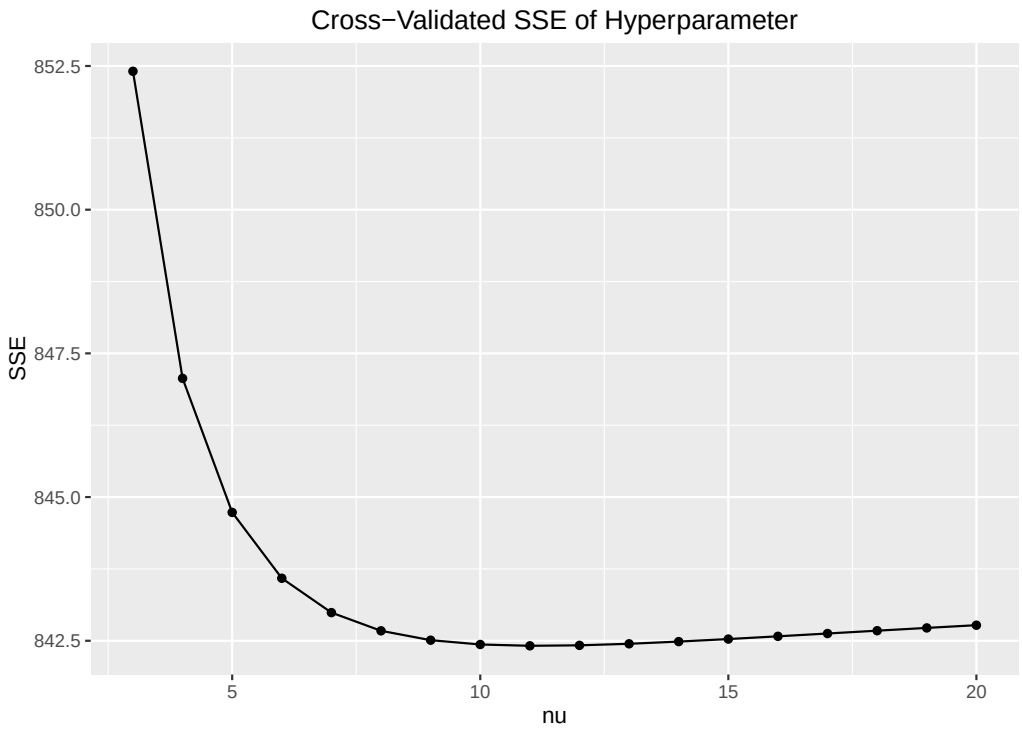

## 3

One simple idea is to track the loglikelihood of the parameters at every given iteration, and stop when the change between the current iteration and the last drops below a certain tolerance level. You can see this implemented in the code in the appendix, but here is a graph showing difference possible tolerance level cutoffs one could pick, in regards to the loglikelihood of the model:




## 4

Estimating the hyperparameter, $\nu$, can be done in a number of ways. One way could be to put a prior on the hyperparameter itself, a hyperprior, and try and learn the parameter from the data. This would not lead to the sample analytical solution for the above implementation, but one could trivially implement this using MCMC. Another option is to use cross validation to try a number of different hyperparameters, and pick the one that gives the best results. This becomes a basic model-selection process. Plotted we have different $\nu$ values and their cross-validated SSE using k-fold cross validation with K = 5. This shows that a choice of 10 is actually very sensible!




# A Source Code

```
################### LOAD ###################################
load("synth_reg.rda")
library(dplyr)
library(ggplot2)
library(MASS)

em <- function (T, X, iter = 10, nu = 10, tol=10**-3) {

    # Initialize our theta variables
    q <- 1
    w <- rep(0, ncol(X))
    ll <- c()
    i <- 1
    flatlined <- FALSE

    while (!flatlined && i <= iter) {
        Z <- estep(T, X, q, w, nu)
        Theta <- mstep(T, X, Z)
        q <- head(Theta, 1)
        w <- tail(Theta, -1)
```

```
        ll <- c(ll, likelihood(T, X, Z, q, w, nu))

        flatlined <- abs(tail(ll, 2) %*% c(-1, 1)) < tol
        i <- i + 1
    }

    # Calculate our standard errors!
    se <- std.err(X, q, Z)

    # Return everything.
    list(Z = Z, q = q, w = w, Z = Z, se = se, ll = ll)
}

std.err <- function (X, q, Z) {
    sqrt(diag(solve(t(X) %*% diag(Z) %*% X)))
}


likelihood <- function (T, X, Z, q, w, nu = 10, sum = TRUE, latent = TRUE) {
    N <- nrow(X)
    s <- c()

    for (i in 1:N) {
        sd <- sqrt(1/(q*Z[i]))
        likelihood <- dnorm(T[i], t(X[i,]) %*% w, sd, log=TRUE)
        lat <- 0
        if (latent) {
            lat <- dgamma(Z[i], nu/2, nu/2 - 1, log=TRUE)
        }
        s <- c(s, likelihood + lat)
    }
    if (sum) {
        s <- sum(s)
    }
    s
}

estep <- function (T, X, q, w, nu = 10) {
    N <- nrow(X)
    sapply(1:N, function (i) {
        (nu + 1)/(nu + q * (T[i] - t(X[i,]) %*% w)**2 - 2)
    })
}

mstep <- function (T, X, Z) {
    N <- nrow(X)
    w = solve((t(X) %*% diag(Z) %*% X ), t(X) %*% diag(Z) %*% T)
    q = 1/(1/N * t(T - X %*% w) %*% diag(Z) %*% (T - X %*% w))
    c(q, w)
}

simulate.dev <- function (Mu, X, Z, q, w, N = 50, quan = .99, latent=TRUE) {
    sim <- rmvnorm(N, Mu, diag(1/(q * Z)))
    dev <- sapply(1:N, function (i) {
        -2 * likelihood(sim[i,], X, Z, q, w, 10, FALSE, latent)
    })
    quantile(dev, quan)
}
```

```r
# Run our algorithm!
X <- cbind(rep(1, nrow(synth_reg$Phy)), synth_reg$Phy)
res <- em(synth_reg$t, X, iter=100, tol = -Inf)

# MLE MODEL (LEAST SQUARES)
lm.model<- lm(t ~ Phy, synth_reg)
lm.coef <- coefficients(lm.model)
lm.Mu <- X %*% coef.lm
lm.q <- 1
lm.Z <- 1/rep(sd(X), nrow(X))

# Weights with errors
rbind(
    data.frame(w = res$w, se = res$se, model = "Robust Regression"),
    data.frame(w = lm.coef, se = coef(summary(lm.model))[, 2], model = "Least Squares")
) %>%
    group_by(model) %>%
    mutate(index = row_number()) %>%
    ggplot(aes(x = index, y = w)) +
    facet_grid(model ~ .) +
    geom_point() +
    geom_errorbar(aes(ymin = w - se, ymax = w + se )) +
    labs(x = "Parameter", y = "Coefficient", title = "Regression Coefficients with 1.96 SE")


# SIMULATIONS FOR 99%
lm.sim.q <- simulate.dev(lm.Mu, X, lm.Z, lm.q, lm.coef, latent = FALSE)
robust.sim.q <- simulate.dev(X %*% res$w, X, res$Z, res$q, res$w)

# DEVIANCES
dev.robust <- -2 * likelihood(T, X, res$Z, res$q, res$w, 10, FALSE)
dev.lm <- -2 * likelihood(T, X, 1/rep(sd(X), nrow(X)), 1, lm.coef, 10, FALSE, FALSE)


# Plotting Deviances
rbind(
    data.frame(residuals = dev.robust, model = "Robust Regression", quant = robust.sim.q),
    data.frame(residuals = dev.lm, model = "Least Squares", quant = lm.sim.q)
    ) %>%
    group_by(model) %>%
    mutate(index = row_number()) %>%
    ggplot(aes(x = index, y = residuals)) +
    facet_grid(model ~ ., scales="free") +
    geom_point() +
    geom_hline(aes(yintercept = quant, color = factor(round(quant, 3)))) +
    scale_colour_discrete(name = "99% Quantile") +
    labs(x = "Index", y = "Deviance",
         title = "Deviance Residuals with 99% Quantile Expectation from Simulation")

res.2 <- em(synth_reg$t, X, iter=20, nu=10)

data.frame(Likelihood = res.2$ll) %>%
    mutate(Iteration = row_number(), Change = c(0, diff(res.2$ll))) %>%
    ggplot(aes(x = Iteration, y = Likelihood)) +
    geom_point() +
    geom_line() +
    geom_vline(aes(xintercept = Iteration, color = factor(round(Change, 5)))) +
```

```
    scale_colour_discrete(name = "Change from Previous Iteration") +
    labs(title = "Log-Likelihood of EM Results Over Iterations")

########################################
# CROSS VALIDATE!!
########################################

my.split <- function (y, n) {

    # support for matrix or vector
    l <- ifelse(class(y) == 'matrix', nrow(y), length(y))
    size <- floor(l/n)
    lapply(1:n, function (i) {
        tail(head(y, i*size), size)
    })
}


cross.validation <- function (y, X, em, nus) {

    # hardcode number of folds
    n <- 5
    sets.y <- my.split(y, n)
    sets.X <- my.split(X, n)
    RSS <- c()

    for (nu in nus) {
        temp <- 0
        for (i in 1:n) {
            test.y <- sets.y[[i]]
            test.X <- sets.X[[i]]

            # ugliness for removing list
            train.y <- unlist(sets.y[setdiff(1:n, i)])
            train.X <- do.call(rbind, sets.X[setdiff(1:n, i)])

            beta <- em(train.y, train.X, iter=20, nu = nu)$w

            # We want the average of the RSS for all, so sum their fraction
            temp <- temp + sum((test.y - test.X %*% beta)**2)/n
        }
        RSS <- c(RSS, temp)
    }

    RSS
}

cv <- cross.validation(synth_reg$t, X, em, 3:20)

data.frame(SSE = cv, nu = 3:20) %>%
    ggplot(aes(x = nu, y = SSE)) +
    geom_point() +
    geom_line() +
    labs(title = "Cross-Validated SSE of Hyperparameter")
```