# MATH 390.4 / 650.2 Spring 2018 Homework #4t

Professor Adam Kapelner

Monday 7th May, 2018

## Problem 1

These are questions about Silver's book, chapters ... For all parts in this question, answer using notation from class (i.e. $t, f, g, h^*, \delta, \epsilon, e, t, z_1, \ldots, z_t, \mathbb{D}, \mathcal{H}, \mathcal{A}, \mathcal{X}, \mathcal{Y}, X, y, n, p, x_{\cdot 1}, \ldots, x_{\cdot p}$, $x_{1\cdot}, \ldots, x_{n\cdot}$, etc. and also we now have $f_{pr}, h^*_{pr}, g_{pr}, p_{th}$, etc from probabilistic classification as well as different types of validation schemes).

(a) [easy] What algorithm that we studied in class is PECOTA most similar to?

K nearest neighbors

(b) [easy] Is baseball performance as a function of age a linear model? Discuss.

No. The data is extremely noisy. However, on average, performance increases with age until a player reaches their peak in their late 20's. After that, they start declining.

(c) [harder] How can baseball scouts do better than a prediction system like PECOTA?

They have access to more $x$'s like the ball's velocity or speed of a runner. This can help their predictions come closer to the truth in some cases, or might make them fit noise in other cases.

(d) [harder] Why hasn't anyone (at the time of the writing of Silver's book) taken advantage of Pitch f/x data to predict future success?

Collecting that data wasn't possible (or was just very expensive), since 3-D recording cameras weren't widely available for use.

(e) [difficult] Chapter 4 is all about predicting weather. Broadly speaking, what is the problem with weather predictions? Make sure you use the framework and notation from class. This is not an easy question and we will discuss in class. Do your best.

$f$ is extremely complicated. The weather behaves like a dynamic system in chaos theory; there are non linear interactions between particles, and there are trillions of particles in the atmosphere.
Our ability to calculate and predict weather outcomes lags behind our theoretical understanding. To get a precise prediction, we'll need to solve equations of motions

for every molecule in the atmosphere. This is impossible. So instead, we need to make simplifying approximations. This is problematic since an $n$ increase in resolution requires an $n^4$ increase in computing power to account for 3 spatial dimensions and time.

(f) [easy] Why does the weatherman lie about the chance of rain? And where should you go if you want honest forecasts?

The weatherman lies about a chance of rain either to seem more confident with his prediction, or to protect himself against unexpected showers. People tend to get more upset over false negatives than false positives when it comes to rain.
"The National Weather Service's forecasts are, it turns out, admirably well calibrated".

(g) [difficult] Chapter 5 is all about predicting earthquakes. Broadly speaking, what is the problem with earthquake predictions? It is *not* the same as the problem of predicting weather. Read page 162 a few times. Make sure you use the framework and notation from class.

there's no real way to measure the meaningful $x$'s since all the stress points occur 15 km below ground. Scientists only have access to $\boldsymbol{y}$'s from the past. The available data contains very little signal, and is mostly noise.

(h) [easy] Silver has quite a whimsical explanation of overfitting on page 163 but it is really educational! What is the nonsense predictor in the model he describes?

Knowing the combinations to 3 specific locks, or knowing specific physical flaws about these locks, and trying to use that to pick other locks.

(i) [easy] John von Neumann was credited with saying that "with four parameters I can fit an elephant and with five I can make him wiggle his trunk". What did he mean by that and what is the message to you, the budding data scientist?

That you can overfit a model quite easily. just a few parameters are enough to fit an obscure shape like an elephant.

(j) [difficult] Chapter 6 is all about predicting unemployment, an index of macroeconomic performance of a country. Broadly speaking, what is the problem with unemployment predictions? It is *not* the same as the problem of predicting weather or earthquakes. Make sure you use the framework and notation from class.

I didn't get to this chapter yet. It's too late now.

(k) [E.C.] Many times in this chapter Silver says something on the order of "you need to have theories about how things function in order to make good predictions." Do you agree? Discuss.

## Problem 2

This question is about validation for the supervised learning problem with one fixed $\mathbb{D}$.

(a) [easy] For one fixed $\mathcal{H}$ and $\mathcal{A}$ (i.e. one model), write below the steps to do a simple validation and include the final step which is shipping the final $g$.

1. Randomly split data into $\mathbb{D}_{test}$ and $\mathbb{D}_{train}$ where $\frac{1}{k}$th of the data is reserved for testing.
2. Produce $g_{train}(x)$ from $\mathcal{A}(\mathcal{H}, \mathbb{D}_{train})$
3. Produce $\hat{\boldsymbol{y}}_{test} = g_{train}(x_{test})$
4. Compute OOSE from $\hat{\boldsymbol{y}}_{test}$ and $\boldsymbol{y}_{test}$
5. Now, train $g_{final}(x) = \mathcal{A}(\mathcal{H}, \mathbb{D})$ (using the whole data set)
6. Ship $g_{final}$ with the OOSE as an estimate for performance in the real world.

(b) [easy] For one fixed $\mathcal{H}$ and $\mathcal{A}$ (i.e. one model), write below the steps to do a $K$-fold cross validation and include the final step which is shipping the final $g$.

1. Randomly shuffle $\mathbb{D}$
2. Split $\mathbb{D}$. The 1st $\frac{1}{k}$th of the data is $\mathbb{D}_{test_i}$ and the rest is $\mathbb{D}_{train_i}$.
3. Produce $g_{train_i}(x)$ from $\mathcal{A}(\mathcal{H}, \mathbb{D}_{train_i})$
4. Produce $\hat{\boldsymbol{y}}_{test_i} = g_{train_i}(x_{test_i})$
5. Compute $OOSE_i$ from $\hat{\boldsymbol{y}}_{test_i}$ and $\boldsymbol{y}_{test_i}$
6. Now, split $\mathbb{D}$ again, using the next $\frac{1}{k}$th of the data for $\mathbb{D}_{test_i}$.
7. Repeat steps 3- 6 for each split of the data into test and train. Store $OOSE_i$ for each.
8. Compute an average OOSE for average performance estimate.
9. Finally, train $g_{final}(x) = \mathcal{A}(\mathcal{H}, \mathbb{D})$ (using the whole data set)
10. Ship $g_{final}$ with the average OOSE as an estimate for performance in the real world.

(c) [harder] For one fixed $\mathcal{H}$ and $\mathcal{A}$ (i.e. one model), write below the steps to do a bootstrap validation and include the final step which is shipping the final $g$.

1. Randomly sample with replacement $n$ observations from $\mathbb{D}$. The selected observations (a.k.a. in bag) constitute $\mathbb{D}_{train}$, the rest (a.k.a. out of bag) constitute $\mathbb{D}_{test}$.
2. Produce $g_{train}(x)$ from $\mathcal{A}(\mathcal{H}, \mathbb{D}_{train})$
3. Produce $\hat{\boldsymbol{y}}_{test} = g_{train}(x_{test})$
4. Compute $OOSE$ from $\hat{\boldsymbol{y}}_{test} - \boldsymbol{y}_{test}$
5. Repeat 1-4 a few times. Take an average of the OOSE's.
6. Finally, train $g_{final}(x) = \mathcal{A}(\mathcal{H}, \mathbb{D})$ (using the whole data set)

7. Ship $g_{final}$ with the average OOSE as an estimate for performance in the real world.

(d) [harder] For one fixed $\mathcal{H}_1, \ldots \mathcal{H}_M$ and $\mathcal{A}$ (i.e. $M$ different models), write below the steps to do a simple validation and include the final step which is shipping the final $g$.

1. Divide $\mathbb{D}$ into $\mathbb{D}_{train}$, $\mathbb{D}_{select}$, $\mathbb{D}_{test}$
2. For each model $1, \ldots, M$
   $g_j = \mathcal{A}(\mathcal{H}_j, \mathbb{D}_{train})$
3. Compute error $OSSE_j = error(\boldsymbol{y}_{select}, g_j(X_{select}))$
4. Repeat 2-3 for all $j \in \{1, \ldots, M\}$
5. $j^* = argmin_j\{OOSE_1, \ldots, OOSE_M\}$
6. Retrain $g_{j^*}(X) = \mathcal{A}(\mathcal{H}_{j^*}, \mathbb{D}_{train+select})$

   $OOSE_{j^*} = error(\boldsymbol{y}_{test}, g_{j^*}(X_{test}))$
7. Do steps 1 to 4 on $\mathbb{D}$ to produce $g_{final}$

(This is copied straight from my notes. I have an issue with the last step. I would have thought that after we found $j^*$, we would retrain $g_{j^*}(X) = \mathcal{A}(\mathcal{H}_{j^*}, \mathbb{D}_{train+select})$, validate it against $\mathbb{D}_{test}$, retrain on all data, and ship. Why do we need to compare against all possible models again? We already found the optimal one. And if the optimal one isn't the one that produces a minimum OOSE in the beginning, why bother doing the whole process?)

(e) [difficult] For one fixed $\mathcal{H}_1, \ldots \mathcal{H}_M$ and $\mathcal{A}$ (i.e. $M$ different models), write below the steps to do a $K$-fold cross validation and include the final step which is shipping the final $g$. This is not an easy problem! There are a lot of steps and a lot to keep track of...

1. Begin by shuffling $\mathbb{D}$.
2. Draw $\frac{1}{k}$ of the observations and assign them to $\mathbb{D}_{test_k}$, draw another $\frac{1}{k}$ observations for $\mathbb{D}_{select_i}$, the rest of the data is assigned to $\mathbb{D}_{train_i}$.
3. For each model $1, \ldots, M$
   $g_j = \mathcal{A}(\mathcal{H}_j, \mathbb{D}_{train_i})$
4. Compute error $OSSE_{ji} = error(\boldsymbol{y}_{select_i}, g_j(X_{select_i}))$
5. Then, Keep $\mathbb{D}_{test_k}$ as it is, and select the next $\frac{1}{k}$ indices for $\mathbb{D}_{select_i}$
6. $OSSE_{j_{ave}}$ is the average $OOSE_{ji}$ of model $g_j$.
7. Repeat 3 - 6 for all $j \in \{1, \ldots, M\}$
8. $j^* = argmin_j\{OOSE_{1_{ave}}, \ldots, OOSE_{M_{ave}}\}$

9. Retrain $g_{j^*}(X) = \mathcal{A}(\mathcal{H}_{j^*}, \mathbb{D}_{train+select})$

   $OOSE_{j^*} = error(\boldsymbol{y}_{test}, g_{j^*}(X_{test}))$

10. Repeat steps 2- 8 for all k slicing into $\mathbb{D}_{test}$ and $\mathbb{D}_{train+select}$
11. Compute an average OOSE for the selected model from the cross validation.
12. Train $g_{j^*}$ using all the data, and ship with the average OOSE for estimated performance in the real world.

## Problem 3

This question is about ridge regression — an alternative to OLS.

(a) [harder] Imagine we are in the "Luis situation" where we have $\boldsymbol{X}$ with dimension $n \times (p+1)$ but $p+1 > n$ and we still want to do OLS. Why would the OLS solution we found previously break down in this case?

The matrix $X$ has rank $n < p+1$.
Therefore, $X^\top X$ is rank deficient and non invertible.

(b) [harder] We will embark now to provide a solution for this case. The solution will also give nice results for other situations besides the Luis situation as well. First, assume $\lambda$ is a positive constant and demonstrate that the expression $\lambda \left|\left|\boldsymbol{w}\right|\right|^2 = \boldsymbol{w}^\top(\lambda \boldsymbol{I})\boldsymbol{w}$ i.e. it can be expressed as a quadratic form where $\lambda \boldsymbol{I}$ is the determining matrix. We will call this term $\lambda \left|\left|\boldsymbol{w}\right|\right|^2$ the "ridge penalty".

$$\boldsymbol{w}^\top(\lambda \boldsymbol{I})\boldsymbol{w} \;= \lambda(\boldsymbol{w}^\top \boldsymbol{I}\boldsymbol{w}) \;=\; \lambda \boldsymbol{w}^\top \boldsymbol{w} \;=\; \lambda \left|\left|\boldsymbol{w}\right|\right|^2$$

(c) [easy] Write the $\mathcal{H}$ for OLS below where there parameter is the $\boldsymbol{w}$ vector. $\boldsymbol{w} \in$ ?

$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_p x_p \quad \boldsymbol{w} \in \mathbb{R}^{p+1}$

(d) [easy] Write the error objective function that OLS minimizes using vectors, then expand the terms similar to the previous homework assignment.

$$SSE \;=\; \sum (y_i - \hat{y}_i)^2$$

$$= (\boldsymbol{y} - \hat{\boldsymbol{y}})^\top (\boldsymbol{y} - \hat{\boldsymbol{y}})$$

$$= (\boldsymbol{y}^\top - \hat{\boldsymbol{y}}^\top)(\boldsymbol{y} - \hat{\boldsymbol{y}})$$

$$= \boldsymbol{y}^\top \boldsymbol{y} - \boldsymbol{y}^\top \hat{\boldsymbol{y}} - \hat{\boldsymbol{y}}^\top \boldsymbol{y} + \hat{\boldsymbol{y}}^\top \hat{\boldsymbol{y}}$$

$$= \boldsymbol{y}^\top \boldsymbol{y} - 2\hat{\boldsymbol{y}}^\top \boldsymbol{y} + \hat{\boldsymbol{y}}^\top \hat{\boldsymbol{y}}$$

$$= \boldsymbol{y}^\top \boldsymbol{y} - 2\big(X\boldsymbol{w}\big)^\top \boldsymbol{y} + \big(X\boldsymbol{w}\big)^\top \big(X\boldsymbol{w}\big)$$

$$= \boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{w}^\top X^\top \boldsymbol{y} + \boldsymbol{w}^\top X^\top X\boldsymbol{w}$$

(e) [easy] Now add the ridge penalty $\lambda\,||\boldsymbol{w}||^2$ to the expanded form you just found and write it below. We will term this two-part error function the "ridge objective".

$$cost \;=\; \sum (y_i - \hat{y}_i)^2 + \lambda\,||\boldsymbol{w}||^2$$

$$= \boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{w}^\top X^\top \boldsymbol{y} + \boldsymbol{w}^\top X^\top X\boldsymbol{w} + \lambda\,||\boldsymbol{w}||^2$$

(f) [easy] Note that the ridge objective looks a bit like the hinge loss we spoke about when we were learning about support vector machines. There are two pieces of this error function in counterbalance. When this is minimized, describe conceptually what is going on.

We want to minimize both the errors and the effective wights. The ridge term pushes estimates towards 0. This minimizes variance in the residuals.

(g) [harder] Now, the ridge penalty term as a quadratic form can be combined with the last term in the least squares error from OLS. Do this, then use the rules of vector derivatives we learned to take $d/d\boldsymbol{w}$ and write the answer below.

$$cost = \boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{w}^\top X^\top \boldsymbol{y} + \boldsymbol{w}^\top X^\top X\boldsymbol{w} + \boldsymbol{w}^\top (\lambda \boldsymbol{I})\boldsymbol{w}$$

$$\tfrac{\partial}{\partial \boldsymbol{w}}[cost] \;=\; -2X^\top \boldsymbol{y} + 2X^\top X\boldsymbol{w} + 2(\lambda \boldsymbol{I})\boldsymbol{w}$$

(h) [easy] Now set that derivative equal to zero. What matrix needs to be invertible to solve?

$$-2X^\top \boldsymbol{y} + 2X^\top X\boldsymbol{w} + 2(\lambda \boldsymbol{I})\boldsymbol{w} = 0$$

$$2X^\top X\boldsymbol{w} + 2\lambda \boldsymbol{I}\boldsymbol{w} = 2X^\top \boldsymbol{y}$$

$$\Big(X^\top X + \lambda \boldsymbol{I}\Big)\boldsymbol{w} = X^\top \boldsymbol{y}$$

The matrix $X^\top X + \lambda \boldsymbol{I}$ needs to be invertible.

(i) [difficult] There's a theorem that says *positive definite* matrices are invertible. A matrix is said to be positive definite if every quadratic form is positive for all vectors i.e. if $\forall \boldsymbol{z} \neq \boldsymbol{0} \;\; \boldsymbol{z}^\top A\boldsymbol{z} > 0$ then $A$ is positive definite. Prove this matrix from the previous question is positive definite.

$$z^\top\Big(X^\top X + \lambda \boldsymbol{I}\Big)z \;=\; z^\top(X^\top X)z \;+\; z^\top(\lambda \boldsymbol{I})z \;=\; ||Xz||^2 + \lambda\,||z||^2$$

Since $\lambda > 0$ the term is always positive.

(j) [easy] Now that it's positive definite (and thus invertible), solve for the $\boldsymbol{w}$ that is the argmin of the ridge objective, call it $\boldsymbol{b}_{ridge}$. Note that this is called the "ridge estimator" and computing it is called "ridge regression" and it was invented by Hoerl and Kennard in 1970.

$$\Big(X^\top X + \lambda \boldsymbol{I}\Big)^{-1}\Big(X^\top X + \lambda \boldsymbol{I}\Big)\boldsymbol{w} = \Big(X^\top X + \lambda \boldsymbol{I}\Big)^{-1} X^\top \boldsymbol{y}$$

$$\boldsymbol{w} = \Big(X^\top X + \lambda \boldsymbol{I}\Big)^{-1} X^\top \boldsymbol{y}$$

(k) [easy] Did we just figure out a way out of Luis's situation? Explain.

It looks like we found a way to minimize errors and solve analytically for $\boldsymbol{w}$

(l) [harder] It turns out in the Luis situation, many of the values of the entries of $\boldsymbol{b}_{ridge}$ are close to 0. Why should that be? Can you explain now conceptually how ridge regression works?

The $||w||$ term in the cost function causes us to choose a $b$ with smaller weights than the OLS solution.
In the case where $\lambda \to 0$ $\;b_{ridge} \to b_{OLS}$
When $\lambda \to \infty$ $\;b_{ridge} \to 0$

(m) [easy] Find $\hat{\boldsymbol{y}}$ as a function of $\boldsymbol{y}$ using $\boldsymbol{b}_{ridge}$. Is $\hat{\boldsymbol{y}}$ an orthogonal projection of $\boldsymbol{y}$ onto the column space of $\boldsymbol{X}$?

$\hat{\boldsymbol{y}} = X \cdot \boldsymbol{w}$
$\hat{\boldsymbol{y}} = X\Big(X^\top X + \lambda \boldsymbol{I}\Big)^{-1} X^\top \boldsymbol{y}$
No, $\hat{\boldsymbol{y}}$ is not an orthogonal projection of $\boldsymbol{y}$ onto Colspace[$X$].

(n) [E.C.] Show that this $\hat{\boldsymbol{y}}$ is an orthogonal projection of $\boldsymbol{y}$ onto the column space of some matrix $\boldsymbol{X}_{ridge}$ (which is not $\boldsymbol{X}$!) and explain how to construct $\boldsymbol{X}_{ridge}$ on a separate page.

(o) [easy] Is the $\mathcal{H}$ for OLS the same as the $\mathcal{H}$ for ridge regression? Yes/no.
Is the $\mathcal{A}$ for OLS the same as the $\mathcal{A}$ for ridge regression? Yes/no.

Yes, $\mathcal{H}$ for OLS is the same as the $\mathcal{H}$ for ridge regression.
No, $\mathcal{A}$ for OLS is different than the $\mathcal{A}$ for ridge regression.

(p) [harder] What is a good way to pick the value of $\lambda$, the hyperparameter of the $\mathcal{A}$ = ridge?

Each value of $\lambda$ will result in a wildly different model. Therefore, for the set of all candidate $\lambda$'s run the model selection process.

(q) [easy] In classification via $\mathcal{A}$ = support vector machines with hinge loss, how should we pick the value of $\lambda$? Hint: same as previous question!

Each value of $\lambda$ will result in a wildly different model. Therefore, for the set of all candidate $\lambda$'s run the model selection process.

(r) [E.C.] Besides the Luis situation, in what other situations will ridge regression save the day?

(s) [difficult] The ridge penalty is beautiful because you were able to take the derivative and get an analytical solution. Consider the following algorithm:

$$\boldsymbol{b}_{lasso} = \underset{\boldsymbol{w} \in \mathbb{R}^{p+1}}{\arg\min} \left\{ (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + \lambda ||\boldsymbol{w}||^1 \right\}$$

This penalty is called the "lasso penalty" and it is different from the ridge penalty in that it is not the norm of $\boldsymbol{w}$ squared but just the norm of $\boldsymbol{w}$. It turns out this algorithm (even though it has no closed form analytic solution and must be solved numerically a la the SVM) is very useful! In "lasso regression" the values of $\boldsymbol{b}_{lasso}$ are not shrunk *towards* 0 they are harshly punished *directly to* 0! How do you think lasso regression would be useful in data science? Feel free to look at the Internet and write a few sentences below.

Lasso regression forces some of the $b$ coefficients to be identically 0. Thus, excluding useless features that only cause overfitting.

(t) [easy] Is the $\mathcal{H}$ for OLS the same as the $\mathcal{H}$ for lasso regression? Yes/no.
Is the $\mathcal{A}$ for OLS the same as the $\mathcal{A}$ for lasso regression? Yes/no.

Yes, $\mathcal{H}$ for OLS is the same as the $\mathcal{H}$ for lasso regression.
No, $\mathcal{A}$ for OLS is different than the $\mathcal{A}$ for lasso regression.

## Problem 4

These are questions about non-parametric regression.

(a) [easy] In problem 1, we talked about schemes to validate algorithms which tried $M$ different prespecified models. Where did these models come from?

A pre-specified model space $\mathcal{H}$

(b) [harder] What is the weakness in using $M$ pre-specified models?

The best fit model will be only as complex and expressive as the pre- defined $\mathcal{H}$. Which is limited by the creativity of the person selecting the $\mathcal{H}$'s.

(c) [difficult] Explain the steps clearly in forward stepwise linear regression.

1. Start by specifying an overly complicated $\mathcal{H}$ and a blank model.
2. Split $\mathbb{D}$ into $\mathbb{D}_{train}$ and $\mathbb{D}_{test}$
3. Try adding every feature in the complicated $\mathcal{H}$ to the model. Produce a $g_j$ for every such feature.
4. Compute $OOSE_j = error(\hat{\boldsymbol{y}}_{test}, g_j(X_{test}))$
5. Find $j^* = argmin(OOSE_1, \ldots, OOSE_M)$ which is the most expressive feature.
6. Repeat steps 3- 5 for all features that weren't yet added.
7. Stop when computer crashes, or when OOSE starts to steadily increase with each feature addition.
8. Retrain the model on the whole thing.

(d) [difficult] Explain the steps clearly in *backwards* stepwise linear regression.

Same Idea as previously. However, here we start with a very complicated model and drop a useless feature in every step. At the end we arrive at a optimally fit model.

(e) [harder] What is the weakness(es) in this stepwise procedure?

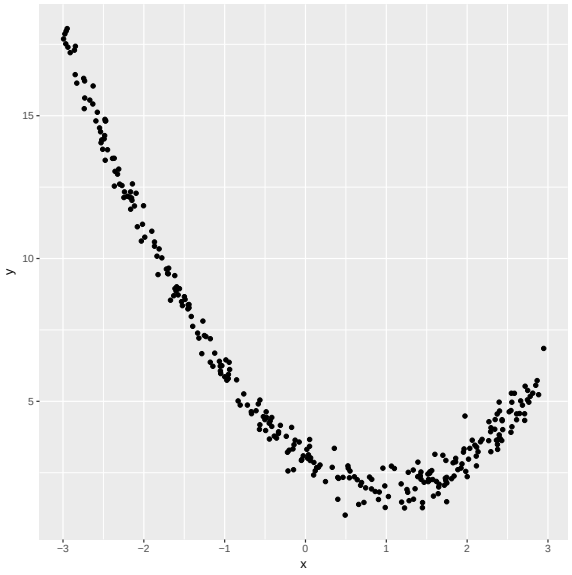
1. You still need to specify predictor set. 2. The models are still linear. 3. Computation is expensive.

(f) [easy] Define "non-parametric regression". What problem(s) does it solve? What are its goals? Discuss.

Regression that doesn't seek to fit certain parameters in a pre- specified $\mathcal{H}$.
It allows you to fit a arbitrarily complicated $f$ without having to account for it in the $\mathcal{H}$.

(g) [harder] Provide the steps for the regression tree (the one algorithm we discussed in class) below.

1. Begin with all data.
2. For every possible split of data into $< X_L, \boldsymbol{y}_L >, < X_R, \boldsymbol{y}_R >$ :
   calculate $SSE_L = \sum (y_{L_i} - \bar{y}_L)^2$ and $SSE_R = \sum (y_{R_i} - \bar{y}_R)^2$
3. Find split than minimizes $SSE_{tot} = SSE_L + SSE_R$
4. Make the split.
5. $< X_L, \boldsymbol{y}_L >$ becomes data, repeat 1- 4 on this.
   $< X_R, \boldsymbol{y}_R >$ becomes data, repeat 1- 4 on this.
6. Recurse until you reach a stop (# observations in a node $\leq N_0$ ).

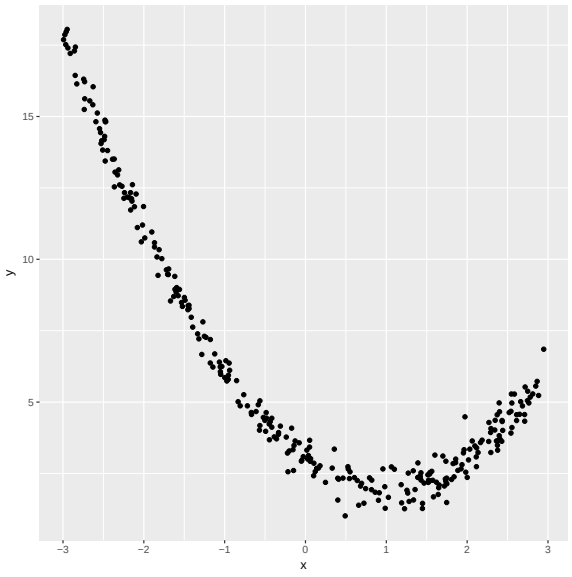



7. Assign $\hat{\boldsymbol{y}} = \bar{y}_0$, where, $\bar{y}_0$ is the sample average in the node.

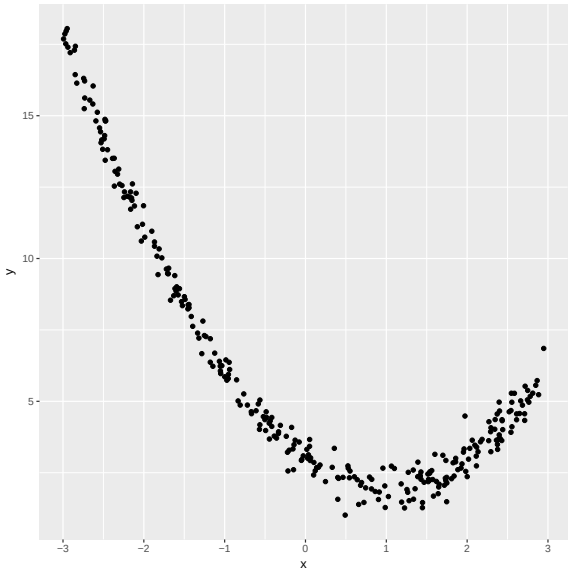
(h) [easy] Consider the following data

Create a tree with maximum depth 1 (i.e one split at the root node) and plot $g$ above.

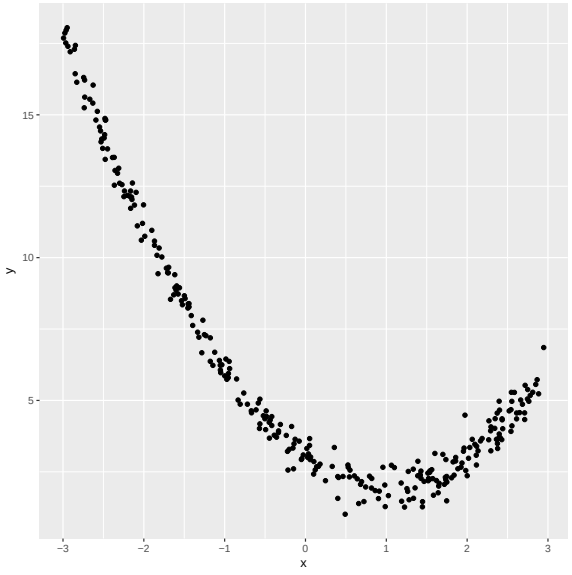
(i) [easy] Now add a second split to the tree and plot $g$ below.




(j) [easy] Now add a third split to the tree and plot $g$ below.

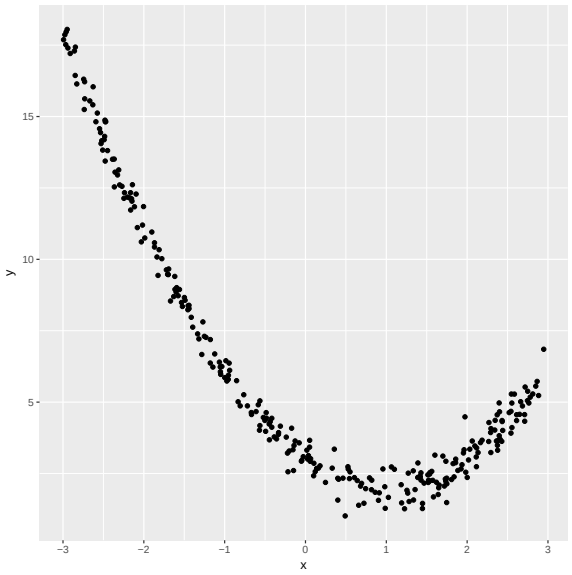



(k) [easy] Now add a fourth split to the tree and plot $g$ below.




(l) [easy] Draw a tree diagram of $g$ below indicating which nodes are the root, inner nodes and leaves. Indicate split rules and leaf values clearly.

(m) [easy] Plot $g$ below for the mature tree with the default $N_0 =$ `nodesize` hyperparameter.




(n) [easy] If $N_0 = 1$, what would likely go wrong?

The model will be overfit, and each point will be mapped to itself.

(o) [easy] How should you pick the $N_0 =$ `nodesize` hyperparameter in practice?

Each $N_0$ creates a different model. run the model selection process on all $N_0$'s in your candidate set.

## Problem 5

These are questions about classification trees.

(a) [easy] How are classification trees different than regression trees?

The $\hat{\boldsymbol{y}}$ in regression trees is continuous, while the $\hat{\boldsymbol{y}}$ in classification trees is a categorical variable.

(b) [harder] What are the steps in the classification tree algorithm?

1. Start with all data.
2. For every possible split into left and right calculate:

   $Gini_L = \sum_{l=1}^{k} \hat{P}_l(1 - \hat{P}_l)$ and $Gini_R = \sum_{l=1}^{k} \hat{P}_l(1 - \hat{P}_l)$

   Where $\hat{P}_l = \frac{\#y_i \text{bin label } l}{\#\text{ observations in node}}$
3. Find split that minimizes $Gini_{weighted} = \frac{n_L Gini_L + n_R Gini_R}{n_l + n_R}$
4. Create split and ad apportion data into right and left daughter nodes.
5. For left and right daughter data, repeat 2- 4. Recurse until reach "stop" (# observations $\leq N_0$).
6. For all leaf nodes, assign $\hat{\boldsymbol{y}}$ = mode $[y_{01}, y_{02}, \ldots, y_{0n}]$

## Problem 6

These are questions about measuring performance of a classifier.

(a) [easy] What is a confusion table?

A table that lays out predictions vs true y.

Consider the following in-sample confusion table where "> 50K" is the positive class:

```
          y_hats_train
y_train  <=50K  >50K
  <=50K   3475   262
  >50K     471   792
```

(b) [easy] Calculate the following: $n$ (sample size) = 3475 + 262 + 471 + 792 = 5000

$FP$ (false positives) = 262

$TP$ (true positives) = 792

$FN$ (false negatives) = 471

$TN$ (true negatives) = 3475

$\#P$ (number positive) = 471 + 792 = 1263

$\#N$ (number negative) = 3475 + 262 = 3737

$\#PP$ (number predicted positive) = 262 + 792 = 1054

$\#PN$ (number predicted negative) = 3475 + 471 = 3946

$\#P/n$ (prevalence / marginal rate / base rate) = 1263 / 5000 = 0.253

$(FP + FN)/n$ (misclassification error) = (262 + 471)/ 5000 = 0.1466

$(TP + TN)/n$ (accuracy) = (3475 + 792)/5000 = 0.8534

$TP/\#PP$ (precision) = 792/ 1054 = 0.7514

$TP/\#P$ (recall, sensitivity, true positive rate, TPR) = 792/ 1263 = 0.627

$2/(\text{recall}^{-1} + \text{precision}^{-1})$ (F1 score) = $2/((0.627)^{-1} + (0.7514)^{-1})$ = 0.7094

$FP/\#PP$ (false discovery rate, FDR) = 262/ 1054 = 0.249

$FP/\#N$ (false positive rate, FPR) = 262 / 3737 = 0.070

$FN/\#PN$ (false omission rate, FOR) = 471/ 3946 = 0.119

$FN/\#P$ (false negative rate, FNR) = 471/ 1263 = 0.373

(c) [easy] Why is FPR also called the "false alarm rate"?

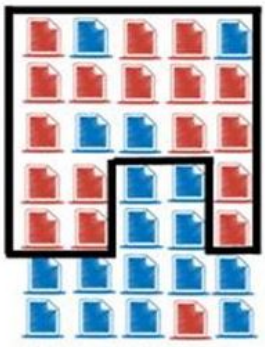
It is the percentage of negatives that were wrongly predicted to be positives. In that sense, they set off "false alarms".

(d) [easy] Why is FNR also called the "miss rate"?

It is the percentage of positives that were wrongly classified as negatives. In that sense, they were missed.

(e) [easy] Below let the red icons be the positive class and the blue icons be the negative class.

The icons included inside the black border are those that have $\hat{y} = 1$. Compute both precision and recall.



$\#P = 18$; $\#PP = 21$; $TP = 17$;
precision $= 17/21 = 0.810$
recall $= 17/18 = 0.944$

(f) [harder] There is always a tradeoff of FP vs FN. However, in some situations, you will look at FPR vs. FNR. Describe such a classification scenario. It does not have to be this income amount classification problem, it can be any problem you can think of.

In a kidney transplant surgery, predicting whether or not the body will reject the organ. Positive means rejection, negative means the transplant was successful. False positive means that the organ is removed (and is disposed of) for no reason, and the patient must wait for another match. False negative might mean failure to detect rejection which might lead to death.
In this case, We will want to minimize both FPR and FNR because both are extremely costly

(g) [harder] There is always a tradeoff of FP vs FN. However, in some situations, you will look at FDR vs. FOR. Describe such a classification scenario. It does not have to be this income amount classification problem, it can be any problem you can think of.

When predicting whether somebody has cancer or not, we care about how many were wrongly classified as ill, and put in treatment for no reason (FDR), and how many were wrongly classified as healthy, and therefore did not receive life saving treatment (FOR).

(h) [harder] There is always a tradeoff of FP vs FN. However, in some situations, you will look at precision vs. recall. Describe such a classification scenario. It does not have to be this income amount classification problem, it can be any problem you can think of.

When predicting fires, we want high precision, which means that most predicted fires were actual true fires. In addition, we want high recall, which means that most actual fires were predicted.

(i) [harder] There is always a tradeoff of FP vs FN. However, in some situations, you will look only at an overall metric such as accuracy (or $F1$). Describe such a classification

scenario. It does not have to be this income amount classification problem, it can be any problem you can think of.

Classifying bugs. We want an overall measure of how successful we were in capturing all of the bugs and none of the dust.