CS 189 Introduction to Machine Learning
Summer 2018 HW3

1 Getting Started

Read through this page carefully. You may typeset your homework in latex or submit neatly handwritten/scanned solutions. Please start each question on a new page. Deliverables:

1. Submit a PDF of your writeup, **with an appendix for your code**, to assignment on Gradescope, "HW3 Write-Up". If there are graphs, include those graphs in the correct sections. Do not simply reference your appendix.
2. If there is code, submit all code needed to reproduce your results, "HW3 Code".
3. If there is a test set, submit your test set evaluation results, "HW3 Test Set".

After you've submitted your homework, watch out for the self-grade form.

(a) Who else did you work with on this homework? In case of course events, just describe the group. How did you work on this homework? Any comments about the homework?

(b) Please copy the following statement and sign next to it. We just want to make it *extra* clear so that no one inadverdently cheats.

I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up.

This homework is due **Monday, July 9th at 11:59pm.**

2 PCA and Random Projections

In this question, we revisit the task of dimensionality reduction. Dimensionality reduction is useful for several purposes, including but not restricted to, visualization, storage, faster computation etc. While reducing dimension is useful, it is desirable to demand that such reductions preserve some properties of the original data. Often, certain geometric properties like distance and inner products are important to perform certain machine learning tasks. And as a result, we may want to perform dimensionality reduction but ensuring that we approximately maintain the pairwise distances and inner products.

While you have already seen many properties of PCA so far, in this question we investigate if random projections are a good idea for dimensionality reduction. A few advantages of random projections over PCA can be: (1) PCA is expensive when the underlying dimension is high and the number of principal components is also large (however note that there are several very fast algorithms dedicated to doing PCA), (2) PCA requires you to have access to the feature matrix for performing computations. The second requirement of PCA is a bottle neck when you want to take only a low dimensional measurement of a very high dimensional data, e.g., in FMRI and in compressed sensing. In such cases, one needs to design a projection scheme before seeing the data. We now turn to a concrete setting to study a few properties of PCA and random projections.

Suppose you are given n points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^d$. Define the $n \times d$ matrix $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}$ where each row of the matrix represents one of the given points. In this problem, we will consider a few low-dimensional linear embedding $\psi : \mathbb{R}^d \mapsto \mathbb{R}^k$ that maps vectors from $\mathbb{R}^d$ to $\mathbb{R}^k$.

Let $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ denote the singular value decomposition of the matrix $\mathbf{X}$. Assume that $n \geq d$ and let $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d$ denote the singular values of $\mathbf{X}$.

Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$ denote the columns of the matrix $\mathbf{V}$. We now consider the following k-dimensional PCA embedding: $\psi_{\text{PCA}}(\mathbf{x}) = (\mathbf{v}_1^\top \mathbf{x}, \ldots, \mathbf{v}_k^\top \mathbf{x})^\top$. Note that this embedding projects a d-dimensional vector on the linear span of the set $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ and that $\mathbf{v}_i^\top \mathbf{x}$ denotes the i-th coordinate of the projected vector in the new space.

We begin with a few matrix algebra relationships, and use them to investigate certain mathematical properties of PCA and random projections in the first few parts, and then see them in action on a synthetic dataset in the later parts.

Notation: The symbol $[n]$ stands for the set $\{1, \ldots, n\}$.

(a) **What is the ij-th entry of the matrices $\mathbf{X}\mathbf{X}^\top$ and $\mathbf{X}^\top\mathbf{X}$? Express the matrix $\mathbf{X}\mathbf{X}^\top$ in terms of $\mathbf{U}$ and $\mathbf{\Sigma}$, and, express the matrix $\mathbf{X}^\top\mathbf{X}$ in terms of $\mathbf{\Sigma}$ and $\mathbf{V}$.**

(b) **Show that**

$$\psi_{\text{PCA}}(\mathbf{x}_i)^\top \psi_{\text{PCA}}(\mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{V}_k \mathbf{V}_k^\top \mathbf{x}_j \quad \text{where} \quad \mathbf{V}_k = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_k \end{bmatrix}.$$

Also **show that** $\mathbf{V}_k\mathbf{V}_k^\top = \mathbf{V}\mathbf{I}^k\mathbf{V}^\top$, where the matrix $\mathbf{I}^k$ denotes a $d \times d$ diagonal matrix with first k diagonal entries as 1 and all other entries as zero.

(c) Suppose that we know the first k singular values are the dominant singular values. In particular, we are given that

$$\frac{\sum_{i=1}^k \sigma_i^4}{\sum_{i=1}^d \sigma_i^4} \geq 1 - \epsilon,$$

for some $\epsilon \in (0, 1)$. Then **show that the PCA projection to the first k-right singular vectors preserves the *inner products* on average:**

$$\frac{1}{\sum_{i=1}^n \sum_{j=1}^n (\mathbf{x}_i^\top \mathbf{x}_j)^2} \sum_{i=1}^n \sum_{j=1}^n |(\mathbf{x}_i^\top \mathbf{x}_j) - (\psi_{\text{PCA}}(\mathbf{x}_i)^\top \psi_{\text{PCA}}(\mathbf{x}_j))|^2 \leq \epsilon. \tag{1}$$

Thus, we find that if there are dominant singular values, PCA projection can preserve the inner products on average.

Hint: Using previous two parts and the definition of Frobenius norm might be useful.

(d) Now consider a different embedding $\psi : \mathbb{R}^d \mapsto \mathbb{R}^k$ which preserves all pairwise distances and norms up-to a multiplicative factor, that is,

$$(1-\epsilon)\|\mathbf{x}_i\|^2 \leq \|\psi(\mathbf{x}_i)\|^2 \leq (1+\epsilon)\|\mathbf{x}_i\|^2 \qquad \text{for all } i \in [n], \quad \text{and} \tag{2}$$

$$(1-\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|^2 \leq \|\psi(\mathbf{x}_i) - \psi(\mathbf{x}_j)\|^2 \leq (1+\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|^2 \qquad \text{for all } i, j \in [n], \tag{3}$$

where $0 < \epsilon \ll 1$ is a small scalar. Further assume that $\|\mathbf{x}_i\| \leq 1$ for all $i \in [n]$. **Show that the embedding ψ satisfying equations** (3) **and** (2) **preserves *each pairwise inner product*:**

$$|\psi(\mathbf{x}_i)^\top \psi(\mathbf{x}_j) - (\mathbf{x}_i^\top \mathbf{x}_j)| \leq C\epsilon, \qquad \text{for all } i, j \in [n], \tag{4}$$

for some constant C. Thus, we find that if an embedding approximately preserves distances and norms upto a small multiplicative factor, and the points have bounded norms, then inner products are also approximately preserved upto an additive factor.

Hint: You may use the Cauchy-Schwarz inequality.

(e) Now we consider the *random projection* using a Gaussian matrix as introduced in the section. In next few parts, we work towards proving that if the dimension of projection is moderately big, then with high probability, the random projection preserves norms and pairwise distances approximately as described in equations (3) and (2).

Consider the random matrix $\mathbf{J} \in \mathbb{R}^{k \times d}$ with each of its entry being i.i.d. $\mathcal{N}(0, 1)$ and consider the map $\psi_{\mathbf{J}} : \mathbb{R}^d \mapsto \mathbb{R}^k$ such that $\psi_{\mathbf{J}}(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{J}\mathbf{x}$. **Show that for any *fixed non-zero vector* $\mathbf{u}$, the random variable** $\dfrac{\|\psi_{\mathbf{J}}(\mathbf{u})\|^2}{\|\mathbf{u}\|^2}$ **can be written as**

$$\frac{1}{k}\sum_{i=1}^k Z_i^2$$

where Z_i's are i.i.d. $\mathcal{N}(0, 1)$ random variables.

(f) **(BONUS)** For i.i.d. $Z_i \sim \mathcal{N}(0,1)$, we have the following probability bound

$$\mathbb{P}\left[\left|\frac{1}{k}\sum_{i=1}^{k} Z_i^2\right| \notin (1-t, 1+t)\right] \leq 2e^{-kt^2/8}, \quad \text{for all } t \in (0,1).$$

Note that this bound suggests that $\sum_{i=1}^{k} Z_i^2 \approx \sum_{i=1} E[Z_i^2] = k$ with high probability. In other words, sum of square of Gaussian random variables concentrates around its mean with high probability. Using this bound and the previous part, now **show that if** $k \geq \frac{16}{\epsilon^2}\log\left(\frac{N}{\delta}\right)$**, then**

$$\mathbb{P}\left[\text{for all } i,j \in [n], i \neq j, \frac{\left\|\psi_{\mathbf{J}}(\mathbf{x}_i) - \psi_{\mathbf{J}}(\mathbf{x}_j)\right\|^2}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2} \in (1-\epsilon, 1+\epsilon)\right] \geq 1-\delta.$$

That is show that for k large enough, with high probability the random projection $\psi_{\mathbf{J}}$ approximately preserves the pairwise distances. Using this result, we can conclude that random projection serves as a good tool for dimensionality reduction if we project to enough number of dimensions. This result is popularly known as the *Johnson-Lindenstrauss Lemma*.

Hint 1: The following (powerful technique cum) bound might be useful: For a set of events A_{ij}, we have

$$\mathbb{P}\left[\cap_{i,j} A_{ij}\right] = 1 - \mathbb{P}\left[(\cap_{i,j} A_{ij})^c\right] = 1 - \mathbb{P}\left[\cup_{i,j} A_{ij}^c\right] \geq 1 - \sum_{ij} \mathbb{P}\left[A_{ij}^c\right].$$

You may define the event $A_{ij} = \left\{\frac{\left\|\psi_{\mathbf{J}}(\mathbf{x}_i) - \psi_{\mathbf{J}}(\mathbf{x}_j)\right\|^2}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2} \in (1-\epsilon, 1+\epsilon)\right\}$ and use the union bound above.

(g) Suppose there are two clusters of points $S_1 = \{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ and $S_2 = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ which are far apart, i.e., we have

$$d^2(S_1, S_2) = \min_{u \in S_1, v \in S_2} \|u - v\|^2 \geq \gamma.$$

Then using the previous part, **show that the random projection $\psi_{\mathbf{J}}$ also approximately maintains the distance between the two clusters if k is large enough, that is, with high probability**

$$d^2(\psi_{\mathbf{J}}(S_1), \psi_{\mathbf{J}}(S_2)) = \min_{u \in S_1, v \in S_2} \left\|\psi_{\mathbf{J}}(\mathbf{u}) - \psi_{\mathbf{J}}(\mathbf{v})\right\|^2 \geq (1-\epsilon)\gamma \quad \text{if } k \geq \frac{C}{\epsilon^2}\log(m+n)$$

for some constant C. Note that such a property can help in several machine learning tasks. For example, if the clusters of features for different labels were far in the original dimension, then this problem shows that even after randomly projecting the clusters they will remain far enough and a machine learning model may perform well even with the projected data. We now turn to visualizing some of these conclusions on a synthetic dataset.

(h) In the next few parts, we visualize the effect of PCA projections and random projections on a classification task. You are given 3 datasets in the data folder and a starter code.

Use the starter code to load the three datasets one by one. Note that there are two unique values in $\mathbf{y}$. **Visualize the features of $\mathbf{X}$ these datasets using (1) Top-2 PCA components, and (2) 2-dimensional random projections. Use the code to project the features to 2 dimensions and then scatter plot the 2 features with a different color for each class.** Note that you will obtain 2 plots for each dataset (total 6 plots for this part). **Do you observe a difference in PCA vs random projections? Do you see a trend in the three datasets?**

(i) For each dataset, we will now fit a linear model on different projections of features to perform classification. The code for fitting a linear model with projected features and predicting a label for a given feature, is given to you. Use these functions and write a code that does prediction in the following way: (1) Use top k-PCA features to obtain one set of results, and (2) Use k-dimensional random projection to obtain the second set of results (take average accuracy over 10 random projections for smooth curves). Use the projection functions given in the starter code to select these features. You should vary k from 1 to d where d is the dimension of each feature $\mathbf{x}_i$. **Plot the accuracy for PCA and Random projection as a function of k. Comment on the observations on these accuracies as a function of k and across different datasets.**

(j) Now plot the singular values for the feature matrices of the three datasets. **Do you observe a pattern across the three datasets? Does it help to explain the performance of PCA observed in the previous parts?**

3 Estimation and approximation in linear regression

In typical applications, we are dealing with data generated by an *unknown* function (with some noise), and our goal is to estimate this function. So far we used linear and polynomial regressions. In this problem we will explore the quality of polynomial regressions when the true function is not polynomial.

Suppose we are given a full column rank feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and an observation vector $\mathbf{y} \in \mathbb{R}^n$. Let the vector $\mathbf{y}$ represent the noisy measurement of a true signal $\mathbf{y}^*$:

$$\mathbf{y} = \mathbf{y}^* + \mathbf{z}, \tag{5}$$

with $\mathbf{z} \in \mathbb{R}^n$ representing the random noise in the observation $\mathbf{y}$, where $z_j \sim \mathcal{N}(0, \sigma^2)$ are i.i.d. We define the vectors $\mathbf{w}^*$ and $\hat{\mathbf{w}}$ as follows:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\mathbf{y}^* - \mathbf{X}\mathbf{w}\|_2^2 \qquad \text{and} \qquad \hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2.$$

Observe that for a given true signal $\mathbf{y}^*$ the vector $\mathbf{w}^*$ is fixed, but the vector $\hat{\mathbf{w}}$ is a random variable since it is a function of the random noise $\mathbf{z}$. Note that the vector $\mathbf{X}\mathbf{w}^*$ is the best linear fit of the true signal $\mathbf{y}^*$ in the column space of $\mathbf{X}$. Similarly, the vector $\mathbf{X}\hat{\mathbf{w}}$ is the best linear fit of the observed noisy signal $\mathbf{y}$ in the column space of $\mathbf{X}$.

After obtaining $\hat{\mathbf{w}}$, we would like to bound the error $\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}^*\|_2^2$, which is the *prediction error* incurred based on the specific n training samples. In this problem we will see how to get a good estimate of this prediction error. When using polynomial features, we will also learn how to decide the degree of the polynomial when trying to fit a noisy set of observations from a smooth function.

Remark: You can use the closed form solution for OLS and results from Discussion 3 for all parts of this problem. For parts (a)-(c), assume that the feature matrix $\mathbf{X}$ and the true signal vector $\mathbf{y}^*$ are fixed (and not random). Furthermore, in all parts the expectation is taken over the randomness in the noise vector $\mathbf{z}$.

(a) **Show that** $\mathbb{E}[\hat{\mathbf{w}}] = \mathbf{w}^*$ and use this fact to **show that**

$$\mathbb{E}\left[\|\mathbf{y}^* - \mathbf{X}\hat{\mathbf{w}}\|_2^2\right] = \|\mathbf{y}^* - \mathbb{E}[\mathbf{X}\hat{\mathbf{w}}]\|_2^2 + \mathbb{E}\left[\|\mathbf{X}\hat{\mathbf{w}} - \mathbb{E}[\mathbf{X}\hat{\mathbf{w}}\|_2^2\right].$$

Note that the above decomposition of the squared error corresponds to the sum of bias-squared and the variance of our estimator $\mathbf{X}\hat{\mathbf{w}}$.

(b) Recall that if $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, where $\Sigma \in \mathbb{R}^{(d\times d)}$ is the covariance matrix, then for any matrix $A \in \mathbb{R}^{k\times d}$, we have $\mathbf{A}\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{A}\Sigma\mathbf{A}^T)$. Use this fact to **show that** the distribution of the vector $\hat{\mathbf{w}}$ is given by

$$\hat{\mathbf{w}} \sim \mathcal{N}(\mathbf{w}^*, \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}).$$

(c) Use part (b) to **show that**

$$\frac{1}{n}\mathbb{E}\left[\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{X}\mathbf{w}^*\|_2^2\right] = \sigma^2\frac{d}{n}.$$

Hint: The trace trick: $\mathrm{trace}(AB) = \mathrm{trace}(BA)$, might be useful.

(d) Assume the underlying model is a noisy linear model with scalar samples $\{\alpha_i, y_i\}_{i=1}^n$, i.e. $y_i = w_1\alpha_i + w_0 + z_i$. We construct matrix $\mathbf{X}$ by using $D+1$ polynomial features $\mathbf{P}_D(\alpha_i) = [1, \alpha_i, \ldots, \alpha_i^D]^\top$ of the *distinct* sampling points $\{\alpha_i\}_{i=1}^n$. For any $D \geq 1$, compare with model (5) and **compute $\mathbf{w}^*$ for this case. Also compute the bias ($\|\mathbf{y}^* - \mathbf{X}\mathbf{w}^*\|_2$) for this case.** Using the previous parts of this problem, **compute the number of samples n required to ensure that the average expected prediction squared error is bounded by ϵ?** Your answer should be expressed as a function of D, σ^2, and ϵ.

Conclude that as we increase model complexity, we require a proportionally larger number of samples for accurate prediction.

(e) Simulate the problem from part (d) for yourself. Set $w_1 = 1$, $w_0 = 1$, and sample n points $\{\alpha_i\}_{i=1}^n$ uniformly from the interval $[-1, 1]$. Generate $y_i = w_1\alpha_i + w_0 + z_i$ with z_i representing standard Gaussian noise.

Fit a D degree polynomial to this data and show how the average error $\frac{1}{n}\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}^*\|_2^2$ scales as a function of both D and n.

You may show separate plots for the two scalings. It may also be helpful to average over multiple realizations of the noise (or to plot point clouds) so that you obtain smooth curves.

(For this part, the libraries numpy.random and numpy.polyfit might be useful. You are free to use any and all python libraries.)

(f) Assume that the underlying model is the noisy exponential function with scalar samples $\{\alpha_i, y_i\}_{i=1}^n$ where $y_i = e^{\alpha_i} + z_i$ with *distinct* sampling points $\{\alpha_i\}_{i=1}^n$, in the interval $[-4, 3]$ and i.i.d. Gaussian noise $z_i \sim \mathcal{N}(0, 1)$. We again construct matrix $\mathbf{X}$ by using $D + 1$ polynomial features $[1, \alpha_i, \ldots, \alpha_i^D]^\top$ and use linear regression to fit the observations. Recall, the definitions of the bias and variance of the OLS estimator from part (a) and **show that for a fixed n, as the degree D of the polynomial increases: (1) the bias decreases and (2) the variance increases**. Use the values you derived for bias and variance and **show that for the prediction error, the optimal choice of D is given by $\mathcal{O}(\log n / \log \log n)$.**

Hint: You can directly use previous parts of this problem, Discussion 3 and Problem 4 of HW 2. You may assume that n and D are large for approximation purposes.

(g) Simulate the problem in part (f) yourself. Sample n points $\{\alpha_i\}_{i=1}^n$ uniformly from the interval $[-4, 3]$. Generate $y_i = e^{\alpha_i} + z_i$ with z_i representing standard Gaussian noise. **Fit a D degree polynomial to this data and show how the average error $\frac{1}{n}\|\mathbf{X}\hat{\mathbf{w}} - \mathbf{y}^*\|_2^2$ scales as a function of both D and n.** You may show separate plots for the two scalings. For scaling with D, choose $n = 120$. It may also be helpful to average over multiple realizations of the noise (or to plot point clouds) so that you obtain smooth curves. (For this part, the libraries numpy.random and numpy.polyfit might be useful and you are free to use any and all python libraries.)

(h) Comment on the differences in plots obtained in part (e) and part (g).

4 Nonlinear Classification Boundaries

Make sure to submit the code you write in this problem to "HW3 Code" on Gradescope.

In this problem we will learn how to use polynomial features to learn nonlinear classification boundaries.

In Problem 5 on HW1, we found that linear regression can be quite effective for classification. We applied it in the setting where the training data points were *approximately linearly separable*. This means there exists a hyperplane such that most of the training data points in the first class are on one side of the hyperplane and most training data points in the second class are on the other side of the hyperplane.

However, often times in practice classification datasets are not linearly separable. In this case we can create features that are linearly separable by augmenting the original data with polynomial features as seen in Problem 5. This embeds the data points into a higher dimensional space where they are more likely to be linearly separable.

In this problem we consider a simple dataset of points $(x_i, y_i) \in \mathbb{R}^2$, each associated with a label b_i which is -1 or $+1$. The dataset was generated by sampling data points with label -1 from a disk of radius 1.0 and data points with label $+1$ from a ring with inner radius 0.8 and outer radius 2.0.

(a) **Run the starter code to load and visualize the dataset and submit a scatterplot of the points with your homework. Why can't these points be classified with a linear classification boundary?**

(b) Classify the points with the technique from Problem 5 on HW1 ("A Simple classification approach"). Use the feature matrix $\mathbf{X}$ whose first column consists of the x-coordinates of the training points and whose second column consists of the y-coordinates of the training points. The target vector $\mathbf{b}$ consists of the class label -1 or $+1$. Perform the linear regression $\mathbf{w}_1 = \arg\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{b}\|_2^2$. **Report the classification accuracy on the test set.**

(c) Now augment the data matrix $\mathbf{X}$ with polynomial features $1, x^2, xy, y^2$ and classify the points again, i.e. create a new feature matrix

$$\Phi = \begin{pmatrix} x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 & 1 \\ x_2 & x_2 & x_2^2 & x_2 y_2 & y_2^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_n & y_n & x_n^2 & x_n y_n & y_n^2 & 1 \end{pmatrix}$$

and perform the linear regression $\mathbf{w}_2 = \arg\min_{\mathbf{w}} \|\Phi\mathbf{w} - \mathbf{b}\|_2^2$. **Report the classification accuracy on the test set.**

(d) **Report the weight vector that was found in the feature space with the polynomial features. Show that up to small error the classification rule has the form $\alpha x^2 + \alpha y^2 \leq \beta$. What is the interpretation of β/α here? Why did the classification in the augmented space work?**

5 Your Own Question

Write your own question, and provide a thorough solution.

Writing your own problems is a very important way to really learn the material. The famous "Bloom's Taxonomy" that lists the levels of learning is: Remember, Understand, Apply, Analyze, Evaluate, and Create. Using what you know to create is the top-level. We rarely ask you any HW questions about the lowest level of straight-up remembering, expecting you to be able to do that yourself. (e.g. make yourself flashcards) But we don't want the same to be true about the highest level.

As a practical matter, having some practice at trying to create problems helps you study for exams much better than simply counting on solving existing practice problems. This is because thinking about how to create an interesting problem forces you to really look at the material from the perspective of those who are going to create the exams.

Besides, this is fun. If you want to make a boring problem, go ahead. That is your prerogative. But it is more fun to really engage with the material, discover something interesting, and then come up with a problem that walks others down a journey that lets them share your discovery. You don't have to achieve this every week. But unless you try every week, it probably won't happen ever.