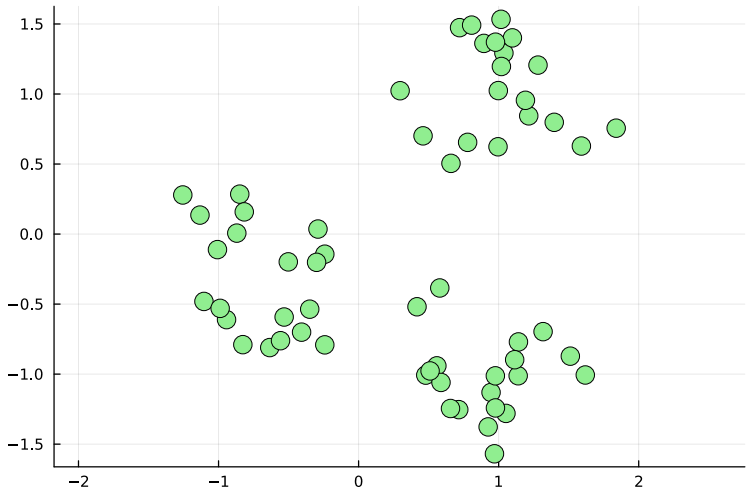
The method of k means is described in Chapter 4 of the text and is an algorithm for discovering clusters in a collection of vectors, such as the following:



A human can look at this figure and determine that there are three categories of vectors, and classify each vector into its category. But the human approach only works in dimensions 2 and 3 where we can visualize: what about dimension 784? The method of k-means can be used, for example, to classify handwriting digits, as in page 80 of your text. The vectors in that application are images arranged as vectors with dimension $28 \times 28 = 784$!

In this lab, you will learn how the algorithm works, finish a Julia implementation that has been started for you, and apply it to a categorize a collection of vectors.

The lab has a companion Jupyter notebook. Follow the instructions below and update the corresponding sections of the notebook as needed. For some problems, you will be asked to attach a hand computation to your final output. To do this, you will make a PDF of your Jupyter notebook and then attach to the end of it scanned PDF pages of your handwritten work.

Exercise 1:
As a warmup exercise, suppose you have the numbers 1, 4, and 9. What number x is closest to all these numbers? There isn't a single best answer to this question; it depends on what is meant precisely by closest. We'll use the following objective function:

$$J(x) = (x-1)^2 + (x-4)^2 + (x-9)^2.$$

Notice that each term of J is a kind of measurement of how far x is from 1, from 4 and from 9 respectively. Our notion of closest will be the number x that minimizes J. (There are lots of other alternatives of "closest", but this one has some very nice analytical properties.)

Anyway, you took a calculus class. By hand, compute the value of x that minimizes J and is therefore (in this sense) closest to the numbers 1, 4 and 9.

Exercise 2:
Compute the average of the numbers 1, 4 and 9. See anything suspicious?

Exercise 3:
We're going to use the same principle for vectors. Suppose $p_1 = (1, -1)$, $p_2 = (2, 2)$ and $p_3 = (-3, 2)$. We'll say w is "closest" to these vectors if

$$J(w) = ||w - p_1||^2 + ||w - p_2||^2 + ||w - p_3||^2$$

is as small as possible. A reasonable guess, given the above, is that the minimum occurs when w is the average of p_1, p_2 and p_3. You can show this is true by an easy hand computation using calculus III skills. Let $w = (x, y)$, plug it in to J, and take the derivative with respect to x and the derivative with respect to y. Set the derivatives equal to zero and solve for x and y. Please do this!

Exercise 4:
I'd like visualize the previous problem using Julia as follows:

a) Create a function `J(x,y)` in Julia that returns the value of the function $J(w)$ above with $w = (x, y)$.

b) Create a contour plot of $J(w)$ with $-4 \le x \le 4$ and $-2 \le y \le 3$. The `JuliaBasics` tutorial on the course website shows how to make contour plots.

c) Add the three point p_1, p_2 and p_3 to the figure. You can do this using `Plots.scatter!` (the exclamation mark is part of the name and indicates you are updating the current plot rather than making a new plot).

d) Again using `Plots.scatter!`, add a single point, the average of the three points, to the figure. It should land at the minimum point!

Use light green circles with a marker size of 8 for the three points and a blue square with the same marker size for the average (which should lie at the minimum of J). The legend for the plot should be informative: "data" for the given points and "min point" for the location of the average.

Exercise 5:
The square distance between vectors is a key part of the k-means algorithm, and other applications in linear algebra.

Create a function `dist_sq(a,b)` that consumes two vectors `a` and `b` of the same length and returns the square distance between them.

Test that your function generates the correct answer when $a = (-1, 2, 2)$ and $b = (3, 5, 1)$ (in which case the square distance is 26).

Exercise 6:
Suppose we have the vectors $p_1 = (1, 3)$, $p_2 = (4, 7)$ and $p_3 = (2, -2)$ and we are trying to group them into two categories.

We'll keep track of category assignments with a vector c as follows: if $c = (1, 1, 2)$ this means that p_1 is in category 1, p_2 is in category 1, but p_3 is in category 2.

Each category gets a representative vector that is supposed to be the best example vector for the category. We'll assume for now that the representatives we have so far are $r_1 = (0, 0)$ and $r_2 = (1, 4)$.

The k-means algorithm proceeds in two steps:

1. Update the category assignments based on the current representatives.
2. Update the representatives based on the current category assignments.

We the repeat the above over and over again until, well, we'll talk about that.

Anyway, for the scenario above, let's see how the category assignment update goes. For each vector p_j we determine which vector r_k is closest to it. Whichever r_k is closest, p_j is put in category k.

Determine the value of the vector c after this update and enter it into the notebook. To do this, you'll need to compute six squared distances (between each p_j and r_k). Computing these by hand would be a pain, so use Julia to help! The notebook has some code set up to help you get started on this.

Exercise 7:
Once the category update has happened, we update the representatives as follows. Suppose vectors p_1 through p_L are in category 1. We let r_1 be the vector that is closest to the p_ℓ's in the sense that r_1 is the vector that minimizes

$$J(w) = ||w - p_1||^2 + \cdots + ||w - p_L||^2$$

As we discussed above, this minimum will be the average of the p_ℓ's. No new calculus required! We just compute averages.

Using this principle, continue the computation from the previous exercise and compute the new representatives r_1 and r_2. This is an easy computation, so you should do it by hand. But also update `r1` and `r2` in the notebook by letting Julia do the same computation.

Exercise 8:
From here on in, it's rinse and repeat! We can keep track of how good a job we are doing by introducing the objective functional J_{clust} defined as follows.

We have vectors $p_1, \ldots, p_L$ that we are categorizing into n categories, we have a category vector c of length L, and we have representatives $r_1, \ldots, r_n$. Then

$$J_{\text{clust}} = \frac{1}{L} \sum_{\ell=1}^{L} ||p_\ell - r_{c_\ell}||^2.$$

The $1/L$ part is just a normalization factor and isn't essential. Each of the terms has the form $||p_\ell - r_{c_\ell}||^2$, which is the square distance from p_ℓ to the current representative of the category that p_ℓ is currently assigned to.

Whew! With all this in mind, compute the value of J_{clust} for the example we've been working. Do this (using Julia) for the the starting values of c and r_j and for the values of c and r_j after the update. The lab notebook has a place for you to record the results of your computation.

You should see that J_{clust} went down!

Exercise 9:
Why does J_{clust} have to go down (or stay the same) when doing a category update?

Exercise 10:
Why does J_{clust} have to go down (or stay the same) when doing a representative update?

Exercise 11:
We need to stop the algorithm at some point! Sometimes we get lucky: for two iterations in a row, the categories don't change, in which case we say that the algorithm converged. Alternatively, we can monitor the value of J_{clust}, which is always going down. When it stops improving much, we can stop the computation. In our work below, we're just going to specify a number of iterations in advance and see what happens.

If for two iterations in a row, the categories don't change, then the categories and the representatives are never going to change every again. Why is that? [This justifies stopping when we see the same category assignments in two iterations]

Exercise 12:
Recall how the category update step goes. Given vector p and a list r_1, through r_k of representatives, we find the the representative r_j with closest square distance to p and assign category j to p.

Your job for this exercise is to fill in a small amount of the code needed for this part of the algorithm. Specifically, the notebook contains most of a function `nearest_rep(p0, reps)` which is supposed to work like this: given the single vector `p0` and the list of representative vectors `reps`, return the index of the representative that is closest to `p0`. See the notebook for the full instructions.

You should be aware of how `reps` is storing all of the representatives all at once. It is a matrix, a two-indexed object. The convention is that `reps[i,j]` is the i^{th} index of representative r_j. I.e., it is $(r_j)_i$ in the notation introduced in class. You can access all of representative j using the notation `reps[:,j]` where the colon means "all the indices".

Follow the instructions in the notebook and fill in the bit of Julia code needed to finish `nearest_rep`

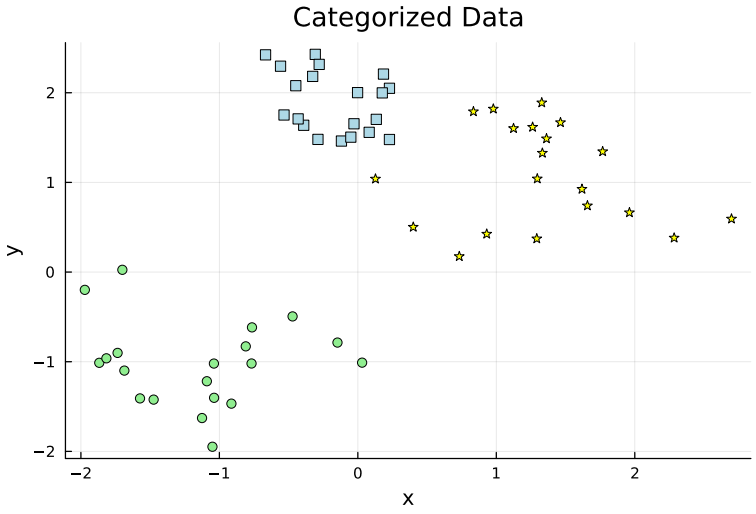
Exercise 13:
The notebook contains the full k-means algorithm (so long as you have filled in `dist_sq` and `nearest_rep`).

If you have data vectors stored in a matrix `p` and want to run the algorithm with 3 clusters and 20 iterations you write:

```
reps, clusters, J = k_means( p, 3, 20)
```

The return values are the matrix `reps` of cluster representatives, the cluster assignments `clusters`, and a vector `J` of the values of the objective function J_{clust} for every iteration.

At the start of exercise 13 in the notebook we construct the following data points:



The notebook also shows you how to run the k-means algorithm on this data and how to generate a plot of the categories when zero iterations of the k-means algorithm have been applied. You'll see the the initial randomly assigned categories.

Now generate an analogous plot, but after three iterations of the algorithm. You should see that the data has started to be grouped, but that the categories don't match the true categories.

Exercise 14:
Generate a plot of the value of J_{clust} over the first 10 iterations of the algorithm. Given this plot, at what iteration do you think we could we have stopped and had our data categorized as best as this algorithm is going to?

Exercise 15:
Generate a plot of the data in their categories after 10 iterations. Add onto this plot the three representatives in red squares.

Comment on the quality of the category assignments.