Teacher Guide: Breast Cancer Classification

Overview
Welcome to the Breast Cancer Classification project. This project is an exciting way to introduce students to how they can leverage real data to draw meaningful conclusions. Although this project is challenging, we're working to make it as approachable as possible.

This project is a 5-7 class day project (for ~40 minute class periods), where students are encouraged to work in pairs (or groups of 3, if necessary). The goal of the project is to implement the k-Nearest-Neighbors machine learning algorithm (more on this below) to classify tumors as benign or malignant. In this folder, we include slides to introduce the project and algorithm to students and starter code that breaks the algorithm down into manageable functions. Everything else, including test cases, GUI, and data files, are fully functional and do not need to be edited.

Contents
Within the folder, you will find the following resources to support you in teaching this project:

1. This teacher guide
2. A PowerPoint introducing machine learning and breast cancer classification
3. An instructions document for students to follow
4. A zip folder with starter code for the Breast Cancer Classification Java project
5. Solution code for the completed Breast Cancer Classification Java Project
6. kNearestNeighbors Extension with further challenges, if any students feel so inclined

Introduction to Machine Learning
We understand that teaching this project, especially if you've never seen machine learning before, can be difficult. As an intro to the topic, we'll walk you through what you need to know to teach the initial slide deck.

The introductory concepts are about machine learning -- the field of study that gives computers the ability to learn without being explicitly programmed. Unlike projects you might have done in class, such as chatbots where you specify in code how the bot should respond to people, the algorithm learns on its own based on data its seen before how to answer questions. Machine learning is broken into two high-level subproblems: supervised machine learning, and unsupervised machine learning. At a high level, supervised machine learning is when you have data, and corresponding 'labels', or outcomes, for that data. For example, you might have emails that are either labeled as spam or not, and your goal is to predict whether a new email is spam or not. For unsupervised learning, algorithms are just given a lot of data and must figure out the groups they belong to. In our case, the breast cancer classification project is based on labeled data (malignant or benign), so it is a supervised learning project.

An important concept is how data is used in machine learning. As I mentioned above, the machine (just like humans) needs to learn from data and experience. However, we can't test our algorithm on the exact same experiences it learned from -- we want to see if it learns well on new data. That's why we split our data into training data and testing data -- one set used for teaching the algorithm, and the other used for evaluating it.

We'll explain the specific algorithm in more detail in the '**k-Nearest-Neighbors Algorithm'** section, but first, how should you approach teaching this project?

Teaching this Project

Here's what we think is a good way to approach teaching this project.

1. Begin by walking through the introductory slides, explaining what's in the above section and showing the videos. Ask students questions like other applications they can think of for machine learning/supervised learning, and ask them which class (supervised vs. unsupervised) they think breast cancer classification falls into. This will likely take half an hour.
2. Allow students to pair off and work on the code. Make sure you encourage them to run their tester code from time to time. Instructions are in the starter code, and they can talk to each other for help.

When evaluating the project, it's likely that not everyone will get it completely accurately. That's fine, and totally understandable -- it's a hard challenge. This project should be graded relatively leniently.

k-Nearest-Neighbors (kNN) Algorithm

At the highest level, kNN works using the general intuition that when I see a new point, it'll probably be similar to the points that are closest to it. What we guess for the new point will be the average prediction of the K points closest to it! How do we measure distance between two points? Well, in our standard x/y coordinate plane, we take $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. For data points with 10 pieces of information, we can generalize this and add up the squared differences of all the coordinates!

Here's some more detail specific to the project:

- We are given a dataset of tumors, each identified by an ID, a set of parameters (ie tumor size, curvature, etc), and most importantly, a label for whether that tumor is malignant or benign. You can find and share more information about the dataset here: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).
- Given a new tumor that we have never seen before, we can compare it to the previous data, based off of its parameters (tumor size, curvature, etc)
- Now, we can figure out the distance between the new tumor and all its neighbors
- Looking at the k closest neighbors to the new tumor and whether those k neighbors are malignant or benign will give us a pretty good approximation for whether the new tumor is benign or malignant.

BreastCancerClassify.java

This Java file is found in the zip folder with starter code for the Breast Cancer Classification Java project. It is the only file that students will be editing to complete the project; it directly implements the k-Nearest-Neighbors algorithm. Here are the functions that students will be writing.

1. calculateDistance - calculates the distance between a given tumor and one other tumor (each tumor is plotted based off of a set of parameters)
2. getAllDistances - calculates the distance between a given tumor and *all* other tumors
3. findKClosestEntries - based off of all the distances between a given tumor and all other tumors, finds the k closest tumors to that given tumor (this function is more challenging to implement correctly, but with the support of some of our test cases it is definitely possible)
4. classify - based off of the k closest tumors to a given tumor, determines if that tumor is benign or malignant; if more than half of a given tumor's k nearest neighbors are malignant, than that tumor is also malignant, and vice versa
5. kNearestNeighbors - combines the first four helper functions to run the entire kNN algorithm

As you can see, we have roughly broken down the k-Nearest-Neighbors Algorithm (see above) into four helper functions that are combined in the last function. Our hope is that by completing each step, students will get a better understanding of how the algorithm works.

Questions

If you have any questions about the curriculum or any of the materials presented, please feel free to email us at teachcsforsocialgood@gmail.com.

Student Guide: Breast Cancer Classification Project

Overview

Welcome to the Breast Cancer Classification project. This project is an exciting way to explore real data and draw meaningful conclusions. Although this project is challenging, we're working to make it as approachable as possible.

The goal of this project is to decide whether a tumor is benign or malignant based on having some information about it: namely, its radius, texture, perimeter, area, smoothness, etc. You're going to be doing this based on a machine learning algorithm, or an algorithm that learns how to make decisions based on having seen previous correct examples. Specifically, you'll be implementing the k-Nearest-Neighbors algorithm. This project isn't easy, but we've broken it down along the way to help make it as manageable as possible! Don't worry about grading -- it'll be lenient, considering the project is hard.

We've also provided you some test cases to see if each of your functions is doing the right thing. Make sure to use these! Every time you write a function, run the tester code to see if it's doing the right thing. It'll help a lot with making sure your code works, and is how development in the real world works.

Introduction to Machine Learning

Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed. Unlike projects you might have done in class, such as chatbots where you specify in code how the bot should respond to people, the algorithm learns on its own based on data its seen before how to answer questions. Machine learning is broken into two high-level subcategories: supervised machine learning, and unsupervised machine learning. At a high level, supervised machine learning is when you have data, and corresponding 'labels', or outcomes, for that data. For example, you might have emails that are either labeled as spam or not, and your goal is to predict whether a new email is spam or not. For unsupervised learning, algorithms are just given a lot of data and must figure out the groups they belong to. In our case, the breast cancer classification project is based on labeled data (malignant or benign), so it is a supervised learning project.

An important concept is how data is used in machine learning. As we discussed above, the machine (just like humans) needs to learn from data and experience. However, we can't test our algorithm on the exact same experiences it learned from -- we want to see if it learns well on new data. That's why we split our data into training data and testing data -- one set used for teaching the algorithm (training it), and the other used for evaluating it (testing it).

k-Nearest-Neighbors (kNN) Algorithm

At the highest level, nKK works using the general intuition that when I see a new point, it'll probably be similar to the points that are closest to it. What we guess for the new point will be the average prediction of the K points closest to it! How do we measure distance between two points? Well, in our standard x/y coordinate plane, we take $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. For data points with 10 pieces of information, we can generalize this and add up the squared differences of all the coordinates!

Here's some more detail specific to the project:

- We are given a dataset of tumors, each identified by an ID, a set of parameters (ie tumor size, curvature, etc), and most importantly, a label for whether that tumor is malignant or benign. You can find more information about the dataset here: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).
- Given a new tumor that we have never seen before, we can compare it to the previous data, based off of its parameters (tumor size, curvature, etc)
- Now, we can figure out the distance between the new tumor and all its neighbors
- Looking at the k closest neighbors to the new tumor and whether those k neighbors are malignant or benign will give us a pretty good approximation for whether the new tumor is benign or malignant.

BreastCancerClassify.java

This Java file is found in the zip folder with starter code for the Breast Cancer Classification Java project. It is the only file that you should be editing to complete the project; it directly implements the k-Nearest-Neighbors algorithm. Here are the functions that you will be writing.

1. calculateDistance - calculates the distance between a given tumor and one other tumor (each tumor is plotted based off of a set of parameters)
2. getAllDistances - calculates the distance between a given tumor and *all* other tumors
3. findKClosestEntries - based off of all the distances between a given tumor and all other tumors, finds the k closest tumors to that given tumor (this function is more challenging to implement correctly, but with the support of some of our test cases it is definitely possible)
4. classify - based off of the k closest tumors to a given tumor, determines if that tumor is benign or malignant; if more than half of a given tumor's k nearest neighbors are malignant, than that tumor is also malignant, and vice versa
5. kNearestNeighbors - combines the first four helper functions to run the entire kNN algorithm

As you can see, we have roughly broken down the k-Nearest-Neighbors Algorithm (see above) into four helper functions that are combined in the last function.