Machine Learning & Discovery *ECS 271, Fall 2025*

HW1: Decision trees and KNN

Daniel Scalettar

1 Math Practice (NOT GRADED)

I was familiar with all of the concepts at some point and am reviewing those in which I am rusty.

2 Continuous Variable vs Discrete Variables

All the attributes we observed in the class so far for decision tree are binary like "Is it Morning?". Now lets assume that our attributes are numerical.

1. *Let's say we want to consider the **Time** on which the class begins. One can argue that we can pick a threshold τ and use (Time $< \tau$)? as a criteria to split the data in two and make a binary tree. Explain how you might pick the optimal value of τ.*

   For a continuous attribute like **Time**, we can consider all the unique values of **Time** in our training data. For example, the training data may have times 9:00, 11:00, 12:00, and 16:00. Unlike the yes/no case where we only have a single potential boundary, we now can consider splitting at any time between the times in our training data. A simple way to do this would be to take the midpoint between each pair of consecutive unique values in the training data. In our example, this would mean our candidates for τ would be 10:00 (between 9:00 and 11:00), 11:30 (between 11:00 and 12:00), or 14:00 (between 12:00 and 16:00). We can then evaluate each candidate τ as we did for binary attributes, but this time across all candidate thresholds of a continuous attribute. In other words, we assign a score like misclassification error (what we did in lecture) to each candidate τ and pick the one that optimizes our chosen score.

2. *In the decision tree learning algorithm discussed in class, once a binary attribute is used, the subtrees do not need to consider it anymore. Explain why in using continuous attributes this may not be the case.*

   In the binary attribute case, once the attribute is used to split the data, no more useful information can be gained from splitting on it gain. For example, if we split on the attribute "Morning?", then all data points on one side of the split will have the "Morning?" attribute set to true, and all data points on the other side will have it set to false. Therefore, if split again on "Morning?" down either branch, no new information will be gained since all the data points on that branch already have the same value for that attribute.

   However, with a continuous attribute like **Time**, even after splitting once, there is still information to be gained down either branch. For example, if you split at $\tau = 12:00$, then on one side you have all data points with **Time** $\geq 12:00$. Splitting on **Time** again at a different time can help further partition the data. For example, on the branch where **Time** $\geq 12:00$, the data can be partitioned at a new split on **Time** such as 14:00.

3 To Memorize or Not to Memorize:

Why using the training data as a dictionary or lookup table and referring to it is a bad strategy for learning? What is it called? How do we prevent it?

Using the training data as a dictionary or lookup table and referring to it is a bad strategy for learning because it will fail to generalize to unseen data. The analogy covered in class was of a student memorizing answers to questions she has seen in lectures instead of understanding the underlying concepts. When presented with new questions on an exam which are similar but not identical, the student will struggle to answer them correctly.

This phenomenon is known as **overfitting** where a model learns the training data too specifically including noise and outliers, rather than capturing the general patterns that would allow it to perform well on new data.

To address this, we can split the training data into an additional validation set. For example, instead of 60% training and 40% testing, we can have 60% training, 20% validation, and 20% testing. We perform training on the training set, but then evaluate the model's performance on the validation set. If the error on the validation set starts increasing while the training error continues decreasing, it indicates overfitting and we can stop training.

4 Visualize

What does the decision boundary of 1-nearest neighbor classifier for 2 classes (one positive, one negative) look like when the features are 1-D? How about 2-D and 3-D? Can you give a general form?

In a 1-nearest neighbor classifier for 2 classes, a test point is classified based on the class of its closest training point. Therefore, the decision boundary is the set of points that are equidistant to the nearest training data points from each class.

In 1-D, this means the boundary is a set of points each located halfway between every pair of adjacent training points from different classes. For example, suppose from our training data we have a positive label point at $x = 2$, a negative label point at $x = 6$, and a positive label point at $x = 8$. The decision boundaries would be at $x = 4$ (midpoint between 2 and 6) and $x = 7$ (midpoint between 6 and 8). This means any point less than 4 would be classified as positive, points between 4 and 7 would be classified as negative, and points greater than 7 would be classified as positive.

In 2-D, the decision boundary is a set of line segments that are the perpendicular bisectors of the line between every pair of nearest training points from different classes. For example, if we have a positive point at (1,1) and a negative point at (3,3) which are the closest positive and negative point to each other, the line segment of the decision boundary for these two points would be a line through the midpoint (2,2), perpendicular to the line connecting (1,1) and (3,3).

In 3-D, the decision boundary is a set of planes that are the perpendicular bisectors of the line between every pair of nearest training points from different classes. For example, if we have a positive point at (1,1,1) and a negative point at (3,3,3) which are the closest positive and negative point to each other, the plane of the decision boundary for these two points would be a plane through the midpoint (2,2,2), perpendicular to the line connecting (1,1,1) and (3,3,3).

By extension, for each feature dimension D, the decision boundary is a set of $(D-1)$-dimensional hyperplanes that split the space between the nearest pair of training points from different classes.

5 kNN and data manipulation

Does the accuracy of a kNN classifier using the Euclidean distance change if you:(a) translate the data (b) scale the data (i.e., multiply all the points by a constant), or (c) rotate the data? Explain. Answer the same

for a kNN classifier using Manhattan distance[1] *and Cosine Distance*[2]*. Make sure you provide mathematical proofs or at least some intuition that why your claim is correct.*

To determine whether the accuracy of a kNN classifier changes under data transformations, we can ask whether the transformation impacts the relative distances between points and thus the nearest neighbors.

In other words, if the transformation preserves the order of distances between points, then the kNN classifier's predictions will remain unchanged, and thus the accuracy will not change.

For the following examples, let p and q be two points, p_t and q_t be their transformed versions, c be a constant scalar, t be a translation vector, and R be a rotation matrix.

1. **Euclidean Distance:** $\mathrm{d(p, q)} = \sqrt{\sum_i (p_i - q_i)^2} = ||p - q||_2$
   - (a) **Translate**
     $\mathrm{d}(p_t, q_t) = ||(p + t) - (q + t)||_2 = ||p - q + t - t||_2 = ||p - q||_2 = \mathrm{d(p, q)}$.
     $\mathrm{d}(p_t, q_t) = \mathrm{d(p, q)}$.
     No change in accuracy.
   - (b) **Scale**
     $\mathrm{d}(p_t, q_t) = ||(cp) - (cq)||_2 = ||c(p - q)||_2 = |c|\,||p - q||_2 = |c|\ \mathrm{d(p, q)}$.
     $\mathrm{d}(p_t, q_t) = |c|\ \mathrm{d(p, q)}$.
     After the scaling transformation, the distances are simply scaled by $|c|$. This means that the relative ordering of distances between points remains unchanged and therefore there is no change in accuracy.
     No change in accuracy.
   - (c) **Rotate**
     $\mathrm{d}(p_t, q_t) = ||Rp - Rq||_2 = ||R(p - q)||_2$.
     Utilizing the fact that rotation matrices preserve L2-norms ($||Rv||_2 = ||v||_2$ for any vector v).
     Therefore, $||R(p - q)||_2 = ||p - q||_2 = \mathrm{d(p, q)}$.
     $\mathrm{d}(p_t, q_t) = \mathrm{d(p, q)}$.
     No change in accuracy.
2. **Manhattan Distance:** $\mathrm{d(p, q)} = \sum_i |p_i - q_i| = ||p - q||_1$
   - (a) **Translate**
     $\mathrm{d}(p_t, q_t) = ||(p + t) - (q + t)||_1 = ||p - q + t - t||_1 = ||p - q||_1 = \mathrm{d(p, q)}$.
     $\mathrm{d}(p_t, q_t) = \mathrm{d(p, q)}$.
     No change in accuracy.
   - (b) **Scale**
     $\mathrm{d}(p_t, q_t) = ||(cp) - (cq)||_1 = ||c(p - q)||_1 = |c|\,||p - q||_1 = |c|\ \mathrm{d(p, q)}$.
     $\mathrm{d}(p_t, q_t) = |c|\ \mathrm{d(p, q)}$.
     After the scaling transformation, the distances are simply scaled by $|c|$. This means that the relative ordering of distances between points remains unchanged and therefore there is no change in accuracy.
     No change in accuracy.
   - (c) **Rotate**
     $\mathrm{d}(p_t, q_t) = ||Rp - Rq||_1 = ||R(p - q)||_1$.
     Unlike Euclidean distance, rotation does not preserve Manhattan distance.

[1] `http://en.wikipedia.org/wiki/Taxicab_geometry`
[2] `https://en.wikipedia.org/wiki/Cosine_similarity`

An intuitive example is to consider the set of all points with an L2 distance of 1 from the origin, which forms a circle in 2D. When rotated, the points remain on the circle with radius 1, thus Euclidean distances between points remain unchanged. However, the set of all points with an L1 distance of 1 from the origin forms a diamond shape. When this diamond is rotated, the distances between points change.

For example, originally the points $p_1 = (1, 0)$ and $p_2 = (0.5, 0.5)$ both have an L1 distance of 1 from the origin. After a 45-degree rotation, the equivalent points become $p_{1t} \approx (0.707, 0.707)$ and $p_{2t} \approx (0, 0.707)$.

The new L1 distances from the origin are:

- d(p_{1t}, origin) $\approx |0.707| + |0.707| \approx 1.414$
- d(p_{2t}, origin) $\approx |0| + |0.707| = 0.707$

After rotation, the original distances (both 1) are not equal to the new distances (approximately 1.414 and 0.707). This demonstrates that rotation can change the relative Manhattan distances between points.

Therefore, the relative ordering of nearest neighbors may be impacted, affecting the accuracy of kNN.

Change in accuracy is possible.

3. **Cosine Distance:** d(p, q) = 1 - $\frac{p \cdot q}{||p||_2 ||q||_2}$

- (a) **Translate**

  d(p_t, q_t) = 1 - $\frac{(p+t)\cdot(q+t)}{||p+t||_2 ||q+t||_2}$.

  The translation affects both the numerator and denominator in a non-trivial way.

  An intuitive way to think about this is that cosine distance measures the angle between two vectors from the origin. When we translate points p and q by vector t, we are potentially changing their directions relative to the origin and thus changing the angle between them.

  Let's consider a simple example in 2D:

  - Let p = (1, 0) and q = (0, 1)
    The cosine distance d(p, q) = 1 - $\frac{(1)(0)+(0)(1)}{\sqrt{1^2+0^2}\sqrt{0^2+1^2}}$ = 1 - 0 = 1.
  - Now, translate both points by t = (1, 1) such that p_t = (2, 1) and q_t = (1, 2)
    The cosine distance d(p_t, q_t) = 1 - $\frac{(2)(1)+(1)(2)}{\sqrt{2^2+1^2}\sqrt{1^2+2^2}}$ = 1 - $\frac{4}{\sqrt{5}\sqrt{5}}$ = 1 - $\frac{4}{5}$ = 0.2.

  The original cosine distance was 1, but after translation it became 0.2. Therefore, since translation changes the cosine distance between points, the relative ordering of nearest neighbors may be affected, impacting the accuracy of kNN.

  Change in accuracy possible.

- (b) **Scale**

  d(p_t, q_t) = 1 - $\frac{(cp)\cdot(cq)}{||cp||_2 ||cq||_2}$ = 1 - $\frac{c^2(p\cdot q)}{|c|||p||_2 |c|||q||_2}$ = 1 - $\frac{p\cdot q}{||p||_2 ||q||_2}$ = d(p, q).

  d(p_t, q_t) = d(p, q).

  Intuitively, scaling (increasing the magnitude of the vectors) does not change the angle between them.

  No change in accuracy.

- (c) **Rotate**

  Utilizing the fact that rotation matrices preserve:

  - **dot products:** $(Rv) \cdot (Ru) = v \cdot u)$
  - **L2-norms:** $(||Rv||_2 = ||v||_2$

  d(p_t, q_t) = 1 - $\frac{(Rp)\cdot(Rq)}{||Rp||_2 ||Rq||_2}$ = 1 - $\frac{p\cdot q}{||p||_2 ||q||_2}$ = d(p, q).

  d(p_t, q_t) = d(p, q).

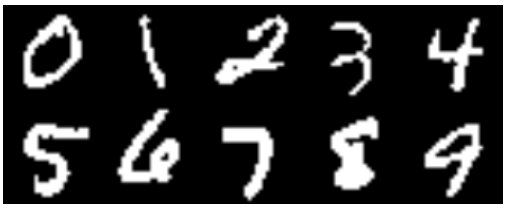
  Intuitively, rotation changes the direction of vectors by the same amount so the angle between them remains unchanged.

  No change in accuracy.

6 Coding

Implement kNN in Python for handwritten digit classification and submit all codes and plots:

Download MNIST digit dataset (60,000 training and 10,000 testing data points) and the starter code from the course page. Each row in the matrix represents a handwritten digit image. The starter code shows how to visualize an example data point. The task is to predict the class (0 to 9) for a given test image, so it is a 10-way classification problem.



1. Write a Python function that implements kNN for this task and reports the accuracy for each class (10 numbers) as well as the average accuracy (one number).

   [acc acc_av] = kNN(images_train, labels_train, images_test, labels_test, k)

   where *acc* is a vector of length 10 and *acc_av* is a scalar. Look at a few correct and wrong predictions to see if it makes sense. To speed it up, in all experiments, you may use only the first 1000 testing images.

2. For $k = 1$, change the number of training data points (30 to 10,000) to see the change in performance. Plot the average accuracy for 10 different dataset sizes. In the plot, x-axis is for the number of training data and y-axis is for the accuracy.

3. Show the effect of k on the accuracy. Make a plot similar to the above one with multiple colored curves on the top of each other (each for a particular k in [1 3 5 10].)

4. Choose the best k. First choose 2,000 training data randomly (to speed up the experiment). Then, split the training data randomly to two halves (the first for training and the second for cross-validation to choose the best k). Please plot the average accuracy wrt k on the validation set. You may search for k in this list: [1 3 5 10]. Finally, report the accuracy for the best k on the testing data.