Contents

- Logistic Regression
- K- Nearest neighbour (KNN)

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Logistic regression

Introduction

- Logistic Regression is commonly used to estimate the probability that an instance belongs to a particular class (e.g., what is the probability that this email is spam?).
- If the estimated probability is greater than 50%, then the model predicts that the instance belongs to that class (called the positive class, labeled “1”), or else it predicts that it does not (i.e., it belongs to the negative class, labeled “0”). This makes it a binary classifier.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Estimating Probabilities :

- Logistic Regression model computes a weighted sum of the input features (plus a bias term), but instead of outputting the result directly like the Linear Regression model does, it outputs the logistic of this result.

$$\hat{p} = h_\theta(\mathbf{x}) = \sigma\left(\theta^T \cdot \mathbf{x}\right)$$

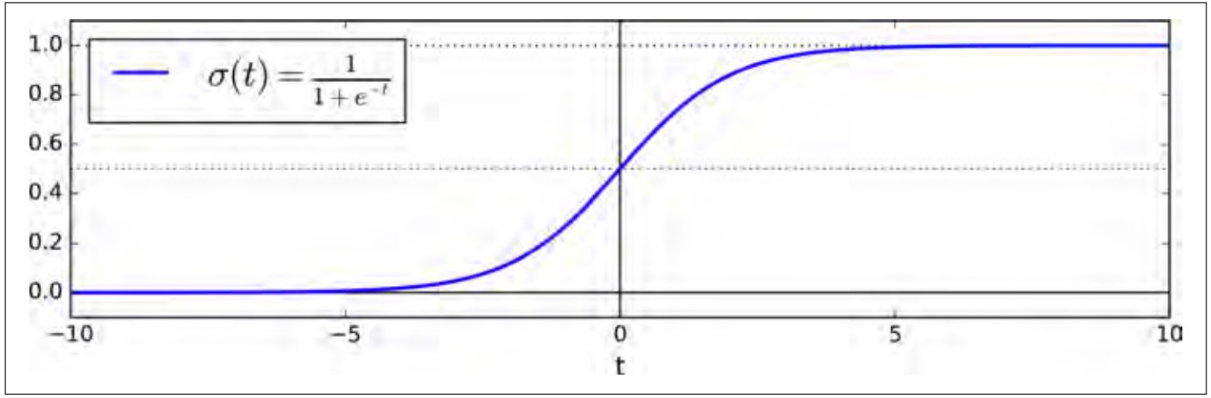
The logistic—also called the logit, noted $\sigma(\cdot)$—is a sigmoid function (i.e., S-shaped) that outputs a number between 0 and 1.

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Logistic Function

- Once the Logistic Regression model has estimated the probability p = hθ (x) that an instance x belongs to the positive class, it can make its prediction ŷ easily.
- Logistic Regression model prediction ŷ = 0 if p < 0 . 5 , 1 if p ≥ 0 . 5



Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Training and Cost function

- The objective of training is to set the parameter vector θ so that the model estimates high probabilities for positive instances (y = 1) and low probabilities for negative instances (y = 0). This idea is captured by the cost function for a single training instance x.

- Cost function of a single training instance

$$c(\theta) = \begin{cases} -\log\left(\hat{p}\right) & \text{if } y = 1, \\ -\log\left(1 - \hat{p}\right) & \text{if } y = 0. \end{cases}$$

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Training and Cost function

- This cost function makes sense because – log(t) grows very large when t approaches 0, so the cost will be large if the model estimates a probability close to 0 for a positive instance, and it will also be very large if the model estimates a probability close to 1 for a negative instance.
- On the other hand, – log(t) is close to 0 when t is close to 1, so the cost will be close to 0 if the estimated probability is close to 0 for a negative instance or close to 1 for a positive instance, which is precisely what we want. The cost function over the whole training set is simply the average cost over all train- ing instances.
- It can be written in a single expression (as you can verify easily), called the log loss.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}log\left(\hat{p}^{(i)}\right) + \left(1 - y^{(i)}\right)log\left(1 - \hat{p}^{(i)}\right)\right]$$

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Training and Cost function

- The bad news is that there is no known closed-form equation to compute the value of θ that minimizes this cost function (there is no equivalent of the Normal Equation).
- But the good news is that this cost function is convex, so Gradient Descent (or any other optimization algorithm) is guaranteed to find the global minimum (if the learn- ing rate is not too large and you wait long enough).
- The partial derivatives of the cost function with regards to the jth model parameter θj is given by

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(\sigma\left(\theta^T \cdot \mathbf{x}^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

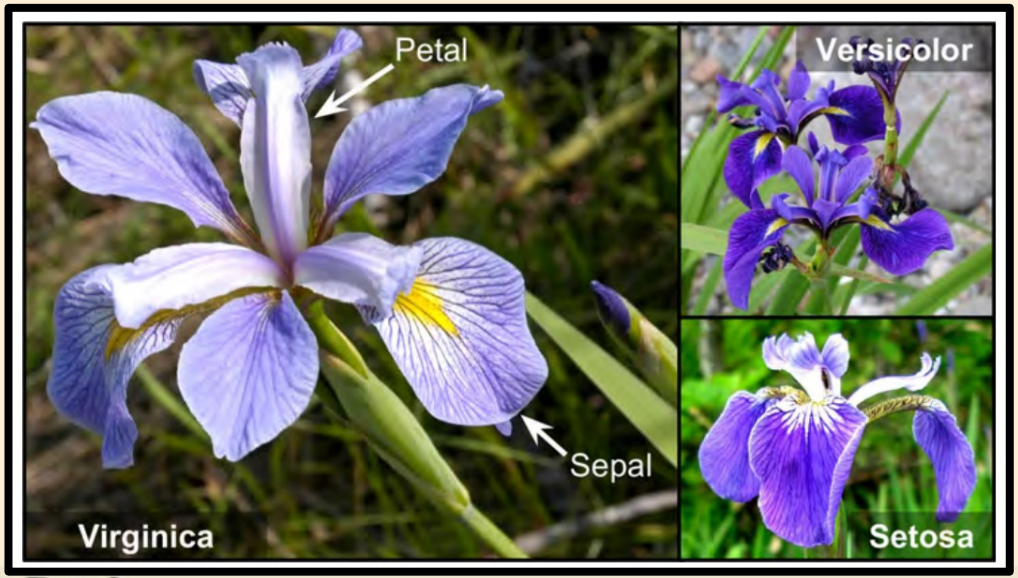
Training and Cost function

- for each instance it computes the prediction error and multiplies it by the jth feature value, and then it computes the average over all training instances.
- Once you have the gradient vector containing all the partial derivatives you can use it in the Batch Gradient Descent algorithm. That's it: you now know how to train a Logistic Regression model.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Decision Boundaries :

- Let's use the iris dataset to illustrate Logistic Regression. This is a famous dataset that contains the sepal and petal length and width of 150 iris flowers of three different species: Iris-Setosa, Iris-Versicolor, and Iris-Virginica.



Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Decision Boundaries :

- Let's try to build a classifier to detect the Iris-Virginica type based only on the petal width feature. First let's load the data:

```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> list(iris.keys())
['data', 'target_names', 'feature_names', 'target', 'DESCR']
>>> X = iris["data"][:, 3:]  # petal width
>>> y = (iris["target"] == 2).astype(np.int)  # 1 if Iris-Virginica, else 0
```

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Decision Boundaries :

- Now let's train a Logistic Regression model:

```
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression()
log_reg.fit(X, y)
```

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Decision Boundaries :

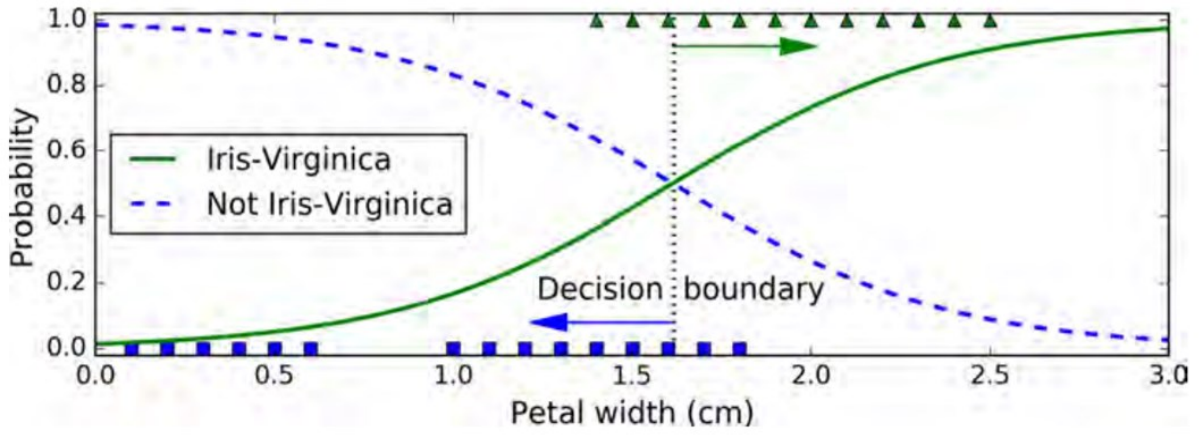
- Let's look at the model's estimated probabilities for flowers with petal widths varying from 0 to 3 cm.

```
X_new = np.linspace(0, 3, 1000).reshape(-1, 1)
y_proba = log_reg.predict_proba(X_new)
plt.plot(X_new, y_proba[:, 1], "g-", label="Iris-Virginica")
plt.plot(X_new, y_proba[:, 0], "b--", label="Not Iris-Virginica")
# + more Matplotlib code to make the image look pretty
```

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Decision Boundaries :

- The petal width of Iris-Virginica flowers (represented by triangles) ranges from 1.4 cm to 2.5 cm, while the other iris flowers (represented by squares) generally have a smaller petal width, ranging from 0.1 cm to 1.8 cm.



Estimated probabilities and decision boundary

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Decision Boundaries :

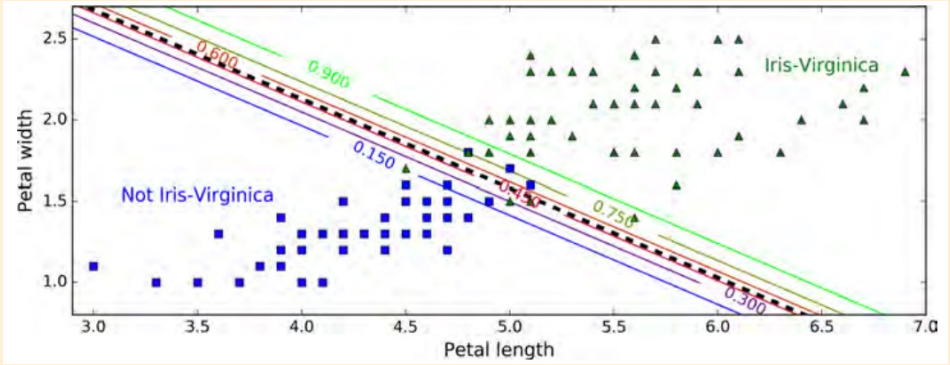
- There is a decision boundary at around 1.6 cm where both probabilities are equal to 50%: if the petal width is higher than 1.6 cm, the classifier will predict that the flower is an IrisVirginica, or else it will predict that it is not (even if it is not very confident):

```
>>> log_reg.predict([[1.7], [1.5]])
array([1, 0])
```

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Decision Boundaries :

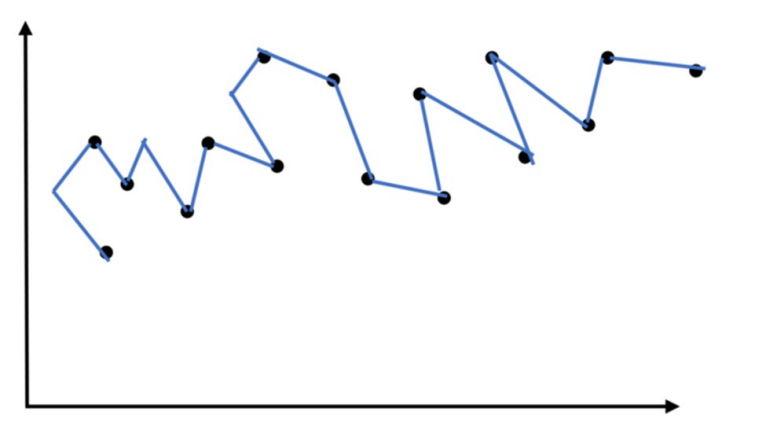
- Fig displays two features: petal width and length.
- Once trained, the Logistic Regression classifier can estimate the probability that a new flower is an Iris-Virginica based on these two features.
- The dashed line represents the points where the model estimates a 50% prrobability: this is the model's decision boundary. Note that it is a linear boundary.17 Each parallel line represents the points where the model outputs a specific probability, from 15% (bottom left) to 90% (top right). All the flowers beyond the top-right line have an over 90% chance of being Iris-Virginica according to the model.



Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Overfitting :

- Overfitting is a modeling error that occurs when a function or model is too closely fit the training set and resulting in a drastic difference of fitting in the test set.



Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Examples :

- Overfitting is a modeling error that occurs when a function or model is too closely fit the training set and resulting in a drastic difference of fitting in the test set.
- we need to predict if a student will land a job interview based on his resume. Now assume we train a model from a dataset of 20,000 resumes and their outcomes.
- Then we try a model out on the original dataset and it predicts outcomes with 98% Accuracy... Wow! It's Amazing, but not in Reality.
- But now comes the bad news. When we run a model out on the new dataset of resumes, we only get 50% of Accuracy.
- Our model doesn't get generalized well from our training data to see unseen data. This is known an Overfitting and it is a common problem in Data Science.
- In fact, Overfitting occurs in the real world all the time. We need to handle it to generalize the model.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Find overfitting :

- The primary challenge in machine learning and in data science is that we can't evaluate the model performance until we test it. So the first step to finding the Overfitting is to split the data into the Training and Testing set.
- The performance can be measured using the percentage of accuracy observed in both data sets to conclude on the presence of overfitting. If the model performs better on the training set than on the test set, it means that the model is likely overfitting. For example, it would be a big Alert if our model saw 99% accuracy on the training set but only 50% accuracy on the test set.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Prevent overfitting :

- Training with more data
- Data augmentation
- Cross validation
- Feature selection
- Regularization

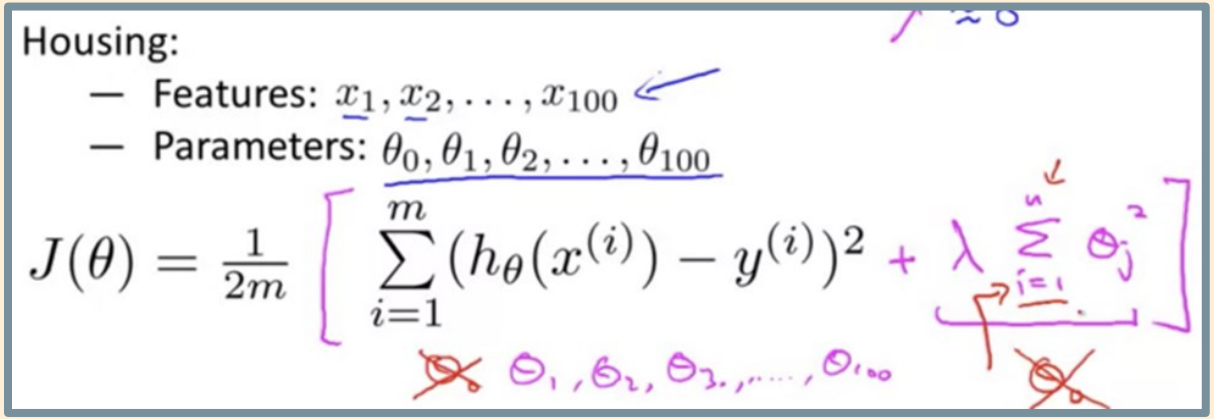
Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Regularization :

- Keep all the features but reduce the magnitude /value of parameters (theta J) to make the value smaller.
- Works well when we have a lot of features, each of which contributes a bit to predicting y.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Regularization :



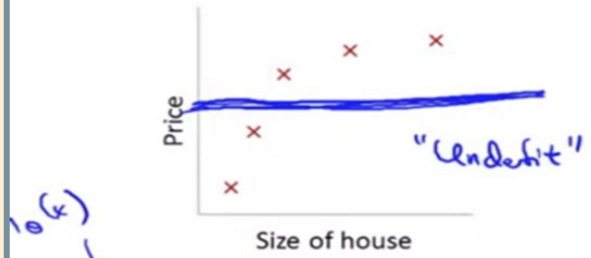
- Modify the cost function by adding an extra regularization term in the end to shrink every single parameter (e.g. close to 0)
- lambda (regularization parameter) controls the tradeoff between two goals:
- former formula — 1st goal: fit the training data well
- extra lambda (purple) — 2nd goal: keep the parameters small to avoid overfitting

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

In regularized linear regression, we choose θ to minimize

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?



$\theta_1, \theta_2, \theta_3, \theta_4$

$\theta_1 \approx 0, \theta_2 \approx 0$

$\theta_3 \approx 0, \theta_4 \approx 0$

$h_\theta(x) = \theta_0$

$h_\theta(x)$

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

- If all parameters (theta) are close to 0, the result will be close to 0. -> it will generate a flat straight line that fails to fit the features well → underfit
- To sum up, if lambda is chosen to be too large, it may smooth out the function too much and cause underfitting.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

The k-nearest neighbors classifier (kNN) is a non-parametric supervised machine learning algorithm. It's distance-based: it classifies objects based on their proximate neighbors' classes.

Non-parametric means that there is no fine-tuning of parameters in the training step of the model. Although k can be considered an algorithm parameter in some sense, it's actually a hyperparameter. It's selected manually and remains fixed at both training and inference time.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

The k-nearest neighbors algorithm is also non-linear. In contrast to simpler models like linear regression, it will work well with data in which the relationship between the independent variable (x) and the dependent variable (y) is not a straight line.

What is k in k-nearest neighbors?

The parameter k in kNN refers to the number of labeled points (neighbors) considered for classification. The value of k indicates the number of these points used to determine the result. Our task is to calculate the distance and identify which categories are closest to our unknown entity.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

How does it works?

Given a point whose class we do not know, we can try to understand which points in our feature space are closest to it.

These points are the k-nearest neighbors. Since similar things occupy similar places in feature space, it's very likely that the point belongs to the same class as its neighbors.

Based on that, it's possible to classify a new point as belonging to one class or another.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

Process :

- **Provide a training set**

- A training set is a set of labelled data that's used for training a model. You can create the training set by manually labelling data or use the labelled databases available in public resources, like this one. A good practice is to use around 75% of the available data for training and 25% as the testing set.

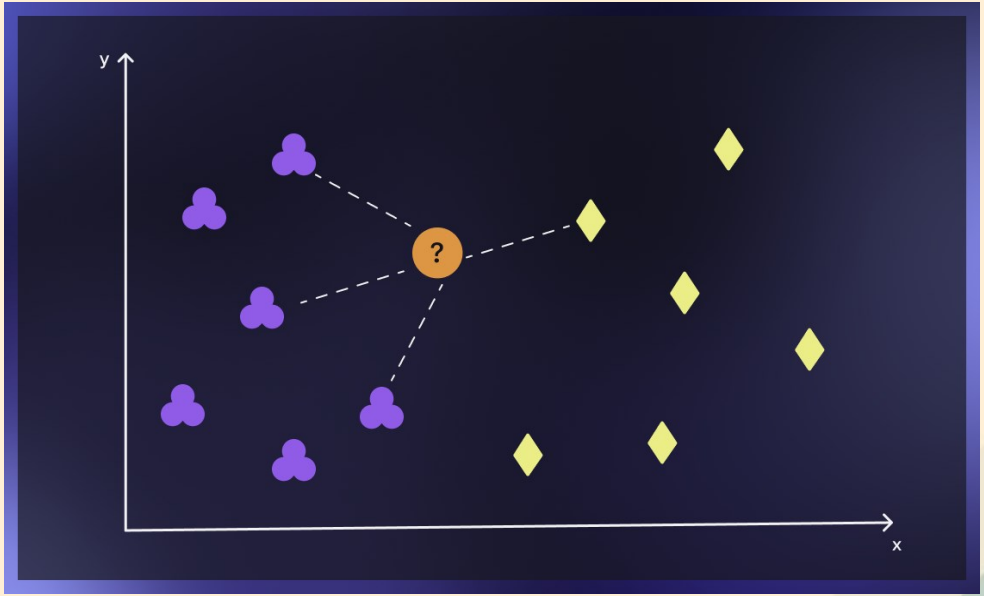
Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

Process :

❖ **Find K nearest**

- Finding k-nearest neighbors of a record means identifying those records which are most similar to it in terms of common features. This step is also known as similarity search or distance calculation.



Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

❖ **Classify Points :**

For classification problems, the algorithm assigns a class label based on a majority vote, meaning that the most frequent label that occurs in neighbors is applied. For regression problems, the average is used to identify the k nearest neighbors.

After the completion of the above steps, the model provides the results. Note that for classification problems, we use discrete values, so the output would be descriptive, for example “likes bright colors”.

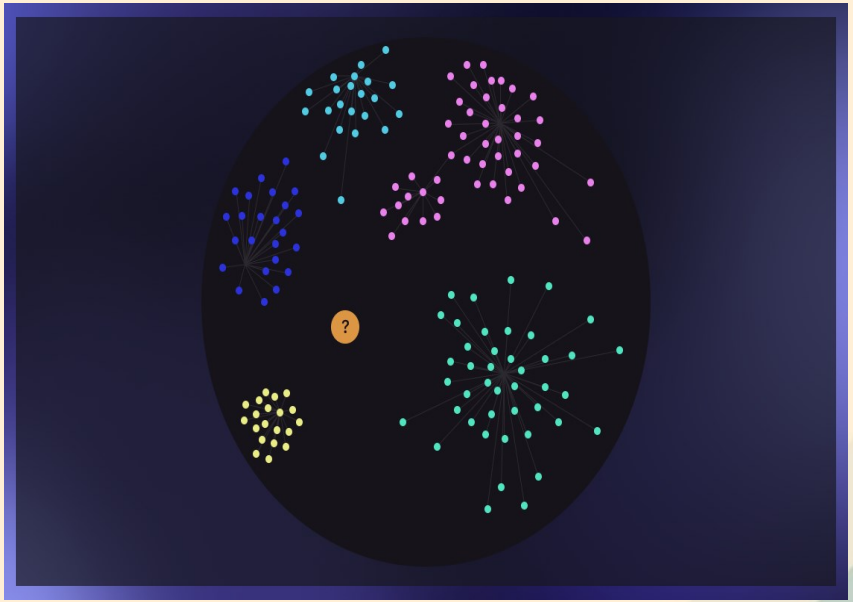
Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

❖ **Find K nearest**

For regression problems, continuous values are applied, meaning that the output would be a floating-point number.

We can evaluate the accuracy of the results by looking at how close the model's predictions and estimates match the known classes in the testing set.



Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

- **How to determine the k value in the k-neighbors classifier?**

1. Select a random k value. In practice, k is usually chosen at random between 3 and 10, but there are no strict rules. A small value of k results in unstable decision boundaries. A large value of k often leads to the smoothening of decision boundaries but not always to better metrics. So it's always about trial and error.
2. Try out different k values and note their accuracy on the testing set.
3. Choose k with the lowest error rate and implement the model.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

❖ Picking k in more complicated cases

In the case of anomalies or a large number of neighbors closely surrounding the unknown point, we need to find different solutions for calculating k that are both time- and cost-efficient.

1. Square root method

The optimal K value can be calculated as the square root of the total number of samples in the training dataset. Use an error plot or accuracy plot to find the most favourable K value. KNN performs well with multi-label classes, but in case of the structure with many outliers, it can fail, and you'll need to use other methods.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
✉ trupesh1991@gmail.com

K-nearest Neighbor(KNN)

2. Cross Validation method (Elbow Method)

Begin with k=1, then perform cross-validation (5 to 10 fold – these figures are common practice as they provide a good balance between the computational efforts and statistical validity), and evaluate the accuracy.

Keep repeating the same steps until you get consistent results. As k goes up, the error usually decreases, then stabilizes, and then grows again. The optimal k lies at the beginning of the stable zone.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

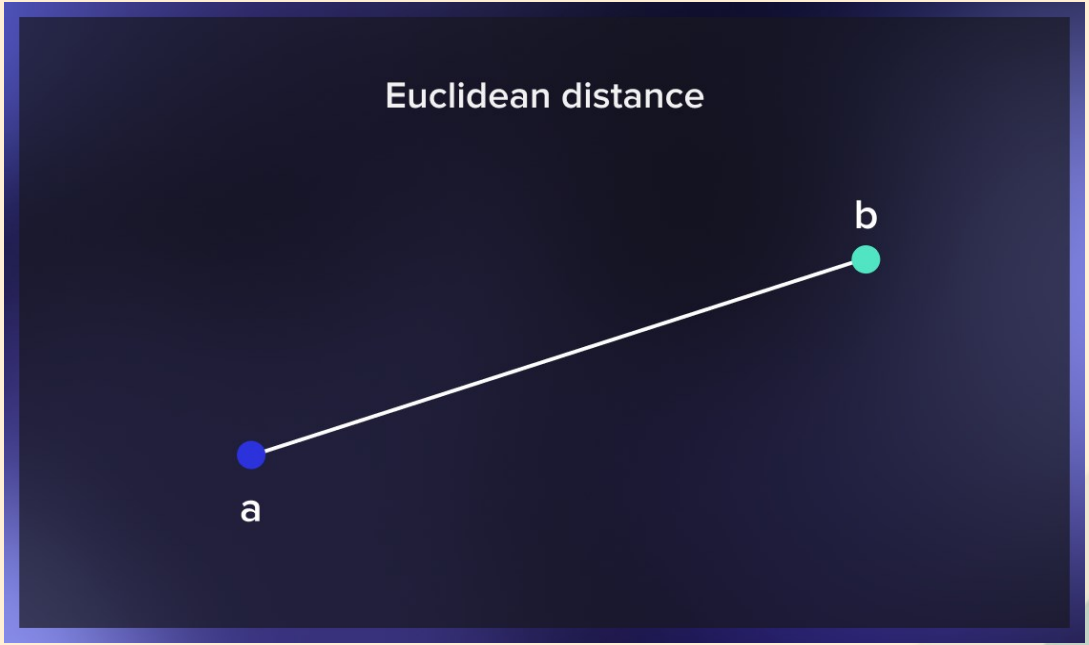
K distance calculation :

K-distance is the distance between data points and a given query point. To calculate it, we have to pick a distance metric.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

Euclidean distance : The Euclidean distance between two points is the length of the straight line segment connecting them. This most common distance metric is applied to real-valued vectors.

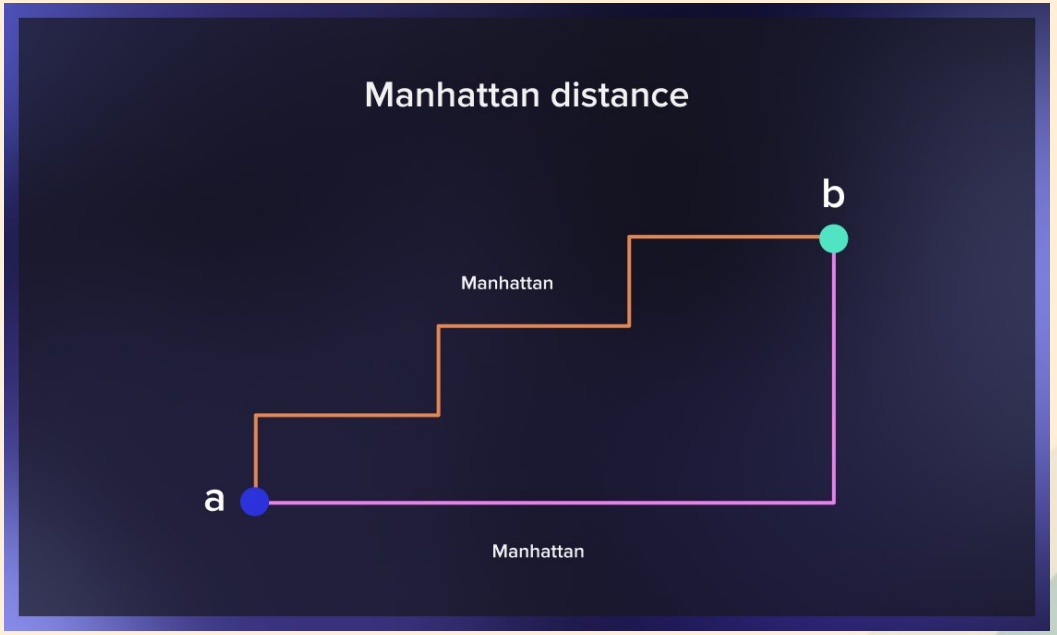


Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

Manhattan distance : The Manhattan distance between two points is the sum of the absolute differences between the x and y coordinates of each point.

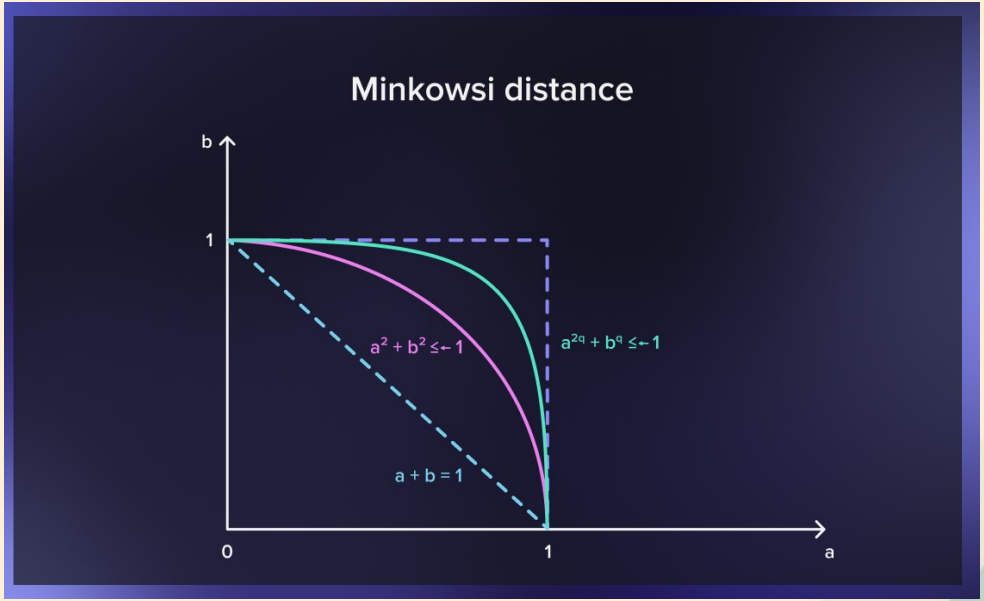
Used to measure the minimum distance by summing the length of all the intervals needed to get from one location to another in a city, it's also known as the taxicab distance.



Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

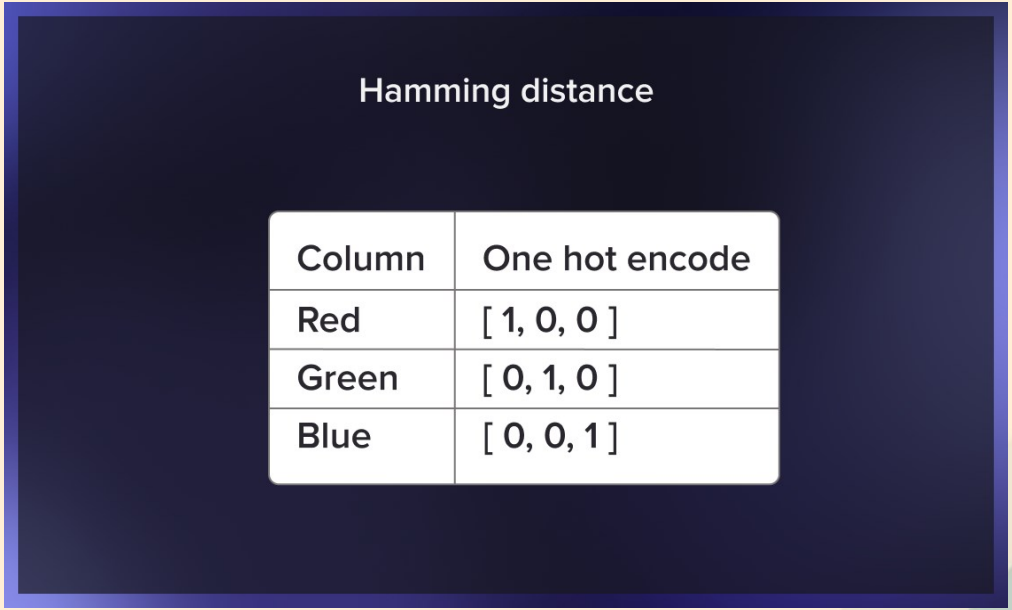
Minkowski distance : Minkowski distance generalizes the Euclidean and Manhattan distances. It adds a parameter called “order” that allows different distance measures to be calculated. Minkowski distance indicates a distance between two points in a normed vector space



Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

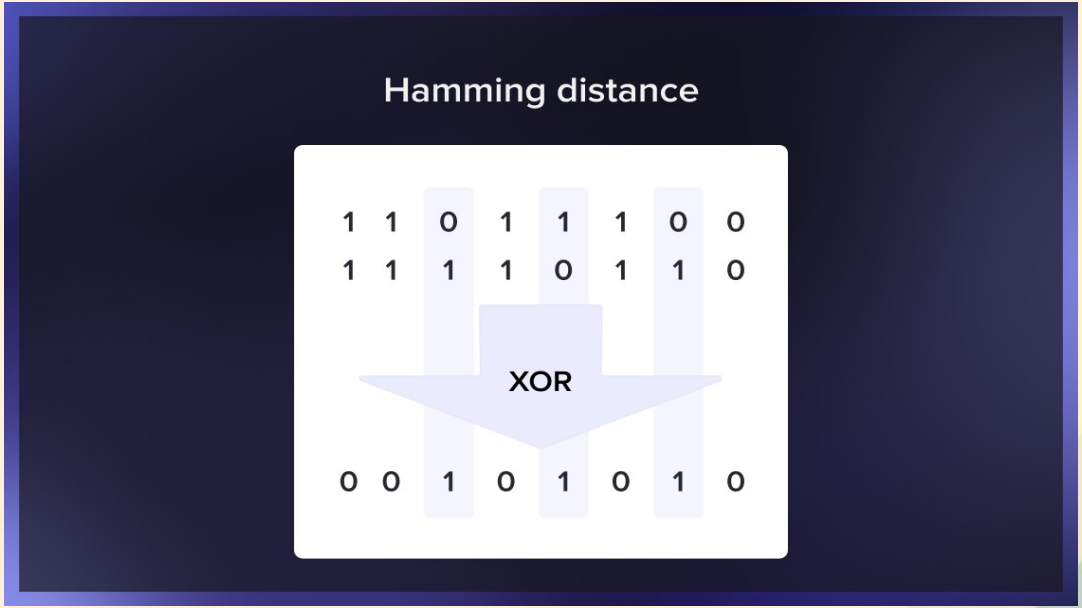
Hamming distance : Hamming distance is used to compare two binary vectors (also called data strings or bitstrings). To calculate it, data first has to be translated into a binary system.



Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

Hamming distance : Hamming distance is used to compare two binary vectors (also called data strings or bitstrings). To calculate it, data first has to be translated into a binary system.



Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

K-nearest Neighbor(KNN)

- Merits

- it’s fast to develop and simple to interpret.
- It can be used for both classification and regression problems.
- kNN is easy to understand and implement and does not require a training period.
- The researcher does not need to have initial assumptions about how similar or dissimilar two examples are.
- K-nearest neighbors only builds a model once a query is performed on the dataset.
- The algorithm can be quickly and effectively paralleled in inference time, which is especially useful for large datasets.

- Demerits

- kNN is more memory-consuming than other classifying algorithms as it requires you to load the entire dataset to run the computation, which increases computation time and costs.
- The k-nearest neighbors algorithm performs worse on more complex tasks such as text classification.
- It requires feature scaling (normalization and standardization) in each dimension.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Practical Applications

Car manufacturer

- An automaker has designed prototypes of a new truck and sedan. To determine their chances of success, the company has to find out which current vehicles on the market are most similar to the prototypes.

- Their competitors are their "nearest neighbors." To identify them, the car manufacturer needs to input data such as price, horsepower, engine size, wheelbase, curb weight, fuel tank capacity, etc., and compare the existing models.

- The kNN algorithm classifies complicated multi-featured prototypes according to their closeness to similar competitors' products.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Practical Applications

E – commerce

- K-nearest neighbors is an excellent solution for cold-starting an online store recommendation system, but with the growth of the dataset more advanced techniques are usually needed.

- The algorithm can select the items that specific customers would like or predict their actions based on customer behaviour data. For example, kNN will quickly tell whether or not a new visitor will likely carry out a transaction.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com

Practical Applications

Education

- Another kNN application is classifying groups of students based on their behaviour and class attendance. With the help of the k-nearest neighbors analysis, it is possible to identify students who are likely to drop out or fail early.

- These insights would allow educators and course managers to take timely measures to motivate and help them master the material.

Prof. Trupesh Patel
Assistant Professor, Computer Engineering Department
trupesh1991@gmail.com