CS M146: Introduction to Machine Learning
k-Nearest Neighbors

Aditya Grover

The instructor gratefully acknowledges Sriram Sankararaman (UCLA) for some of the materials and organization used in these slides, and many others who made their course materials freely available online.

https://aditya-grover.github.io/ @adityagrover_

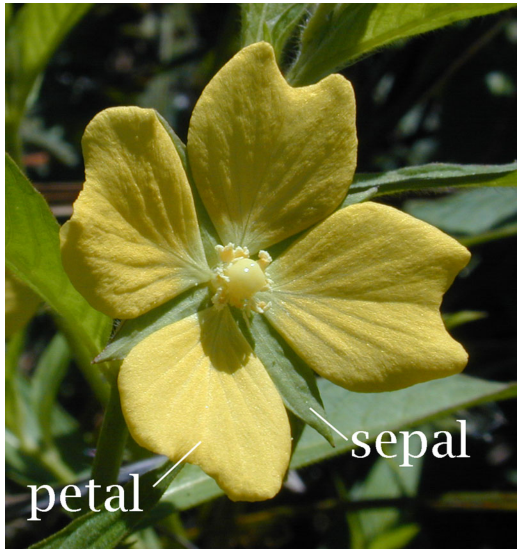
Example: Recognizing flowers

3 types of iris (classes): setosa, veriscolor, virginica

Measuring the properties of flowers

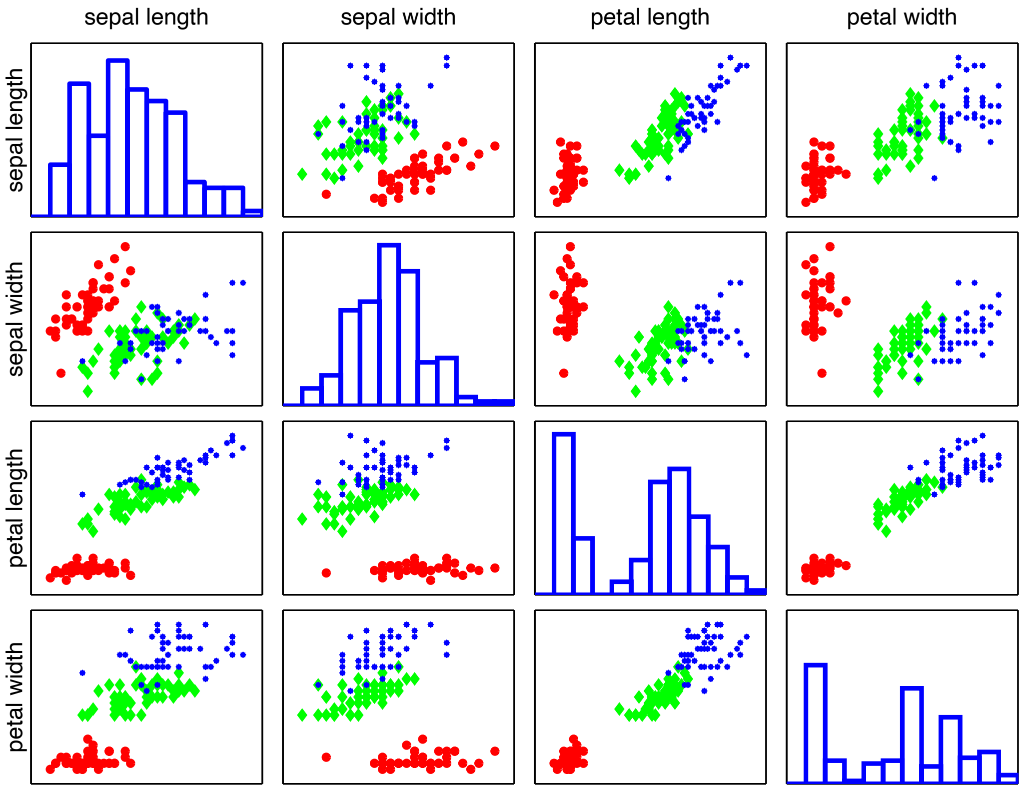
Features: the widths and lengths of sepal and petal

4 features (sepal width, sepal length, petal width, petal length)



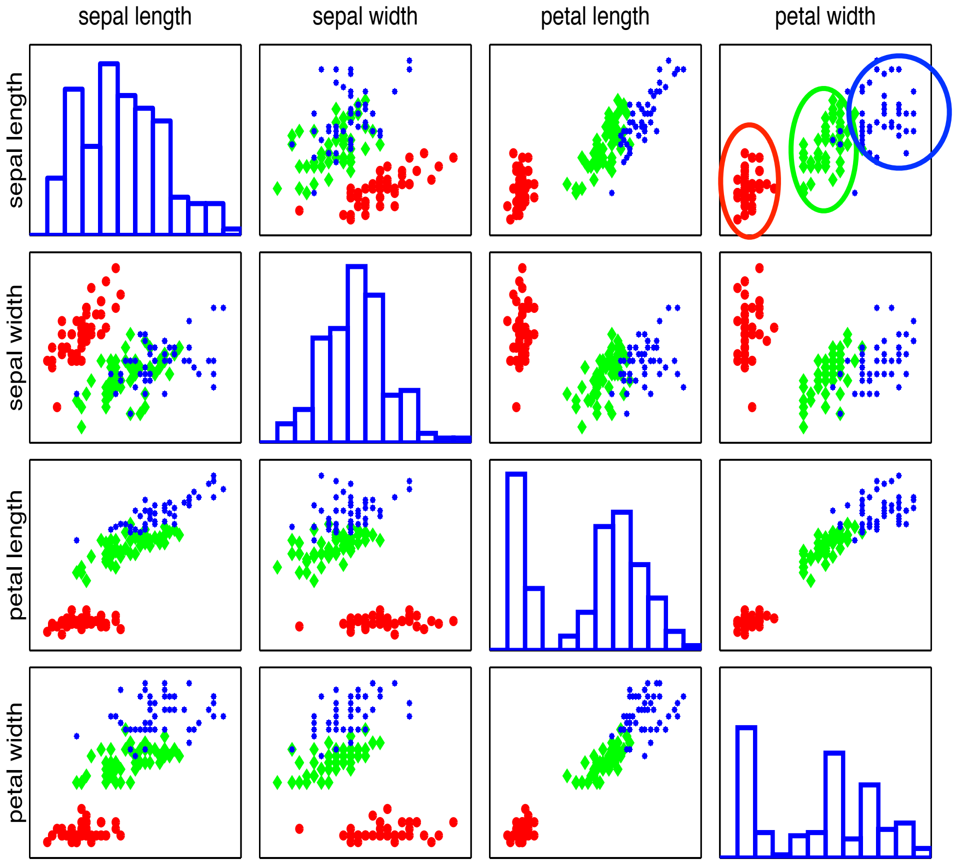
Data Visualization

Each colored point is an instance of a flower: setosa, versicolor, virginica



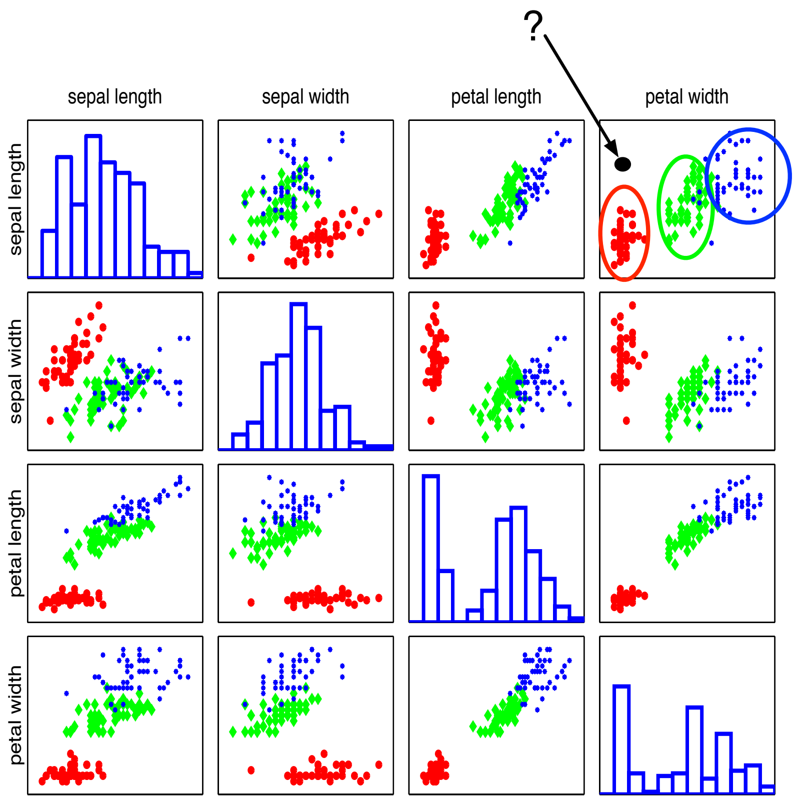
Different types seem well-clustered

Using two features: petal width and sepal length



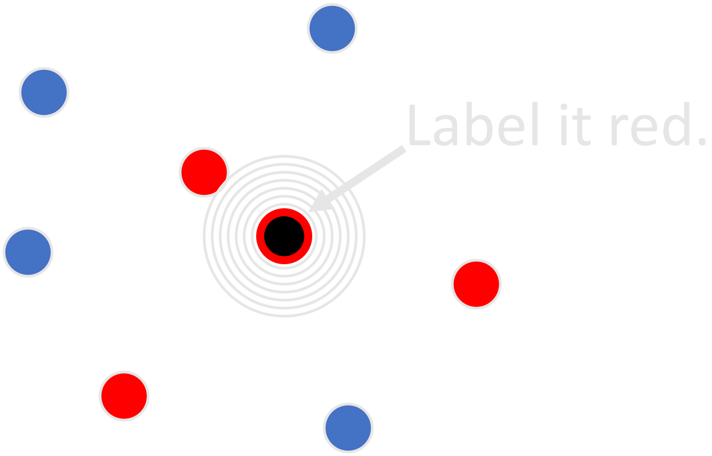
Labeling an unknown flower type

Close to red cluster: so labeling it as setosa



1-Nearest Neighbor

- One of the simplest of all machine learning classifiers
- Idea: label a new point the same as the closest known point



1-Nearest Neighbor

- **Nearest neighbor** is an index to a training instance

$$nn(\boldsymbol{x}) = [i] \text{ where } i \in \{1, 2, \cdots, n\}$$

- To compute nearest neighbor, we need a notion of **distance** between two points, e.g., squared Euclidean (or ℓ_2) distance

$$nn(\boldsymbol{x}) = \underset{i \in [n]}{\operatorname{argmin}} \sum_{j=1}^{d} (x_j - x_j^{(i)})^2$$

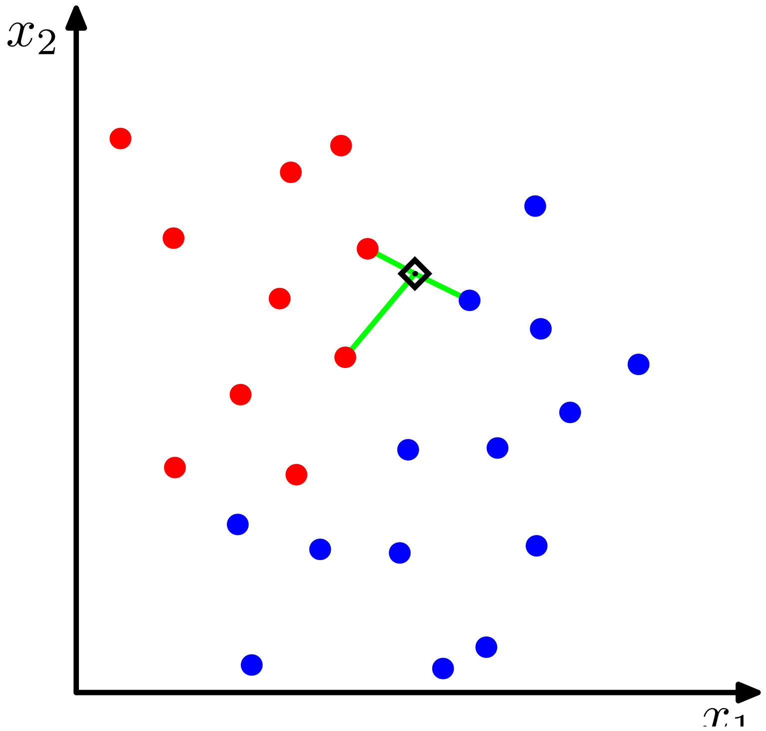
- Classification rule assigns the label of the nearest neighbor

$$h(\boldsymbol{x}) = y_{nn(\boldsymbol{x})}$$

- A type of **non-parametric classifier** (no parameters!)

Visual example

In this 2-dimensional example, the nearest point to $\boldsymbol{x}$ is a red training instance. Thus, $\boldsymbol{x}$ will be labeled red.



Example: classify iris with two features

Training data

ID (n)	petal width (x_1)	sepal length (x_2)	category (y)
1	0.2	5.1	setosa
2	1.4	7.0	versicolor
3	2.5	6.7	virginica

Flower with unknown category

petal width = 1.8, sepal length = 6.4

Calculating distance

ID	distance
1	1.75
2	0.72
3	0.76

Thus, the category is versicolor

How to measure distances?

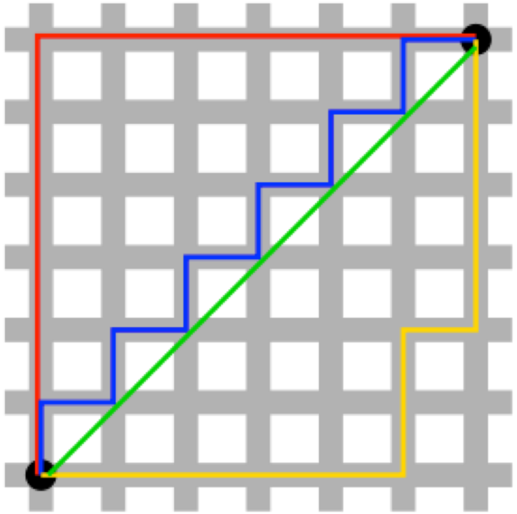
Previously, we used squared Euclidean distance.

$$nn(\boldsymbol{x}) = \underset{i\in[n]}{\operatorname{argmin}} \sum_{j=1}^{d}(x_j - x_j^{(i)})^2$$

Alternative distances

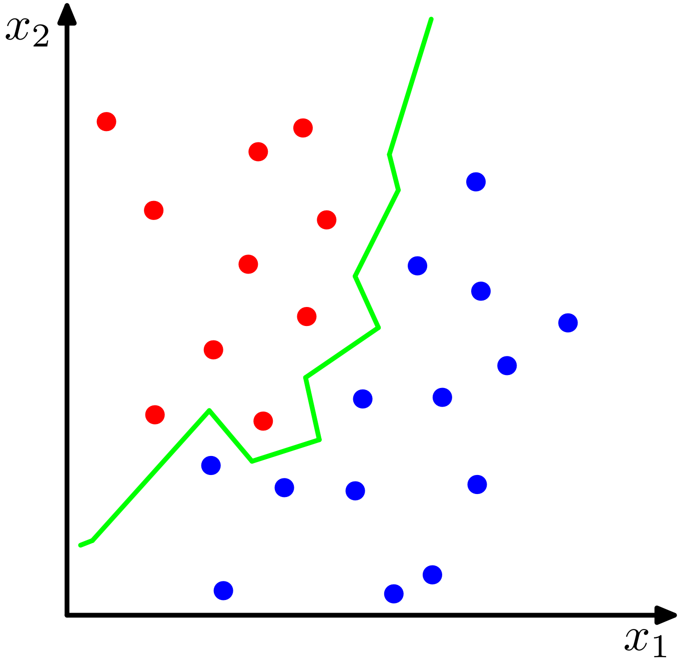
L1 distance

$$nn(\boldsymbol{x}) = \underset{i\in[n]}{\operatorname{argmin}} \sum_{j=1}^{d} | x_j - x_j^{(i)} |$$



Decision boundary

For every point in the space, we can determine its label using the nearest neighbor rule. This gives rise to a decision boundary that partitions the space into different regions.



(b)

k-Nearest Neighbor (kNN) classification

Increase the number of neighbors

1st nearest neighbor

$$nn_1(\boldsymbol{x}) = \underset{i \in [n]}{\operatorname{argmin}} \left\| \boldsymbol{x} - \boldsymbol{x}^{(i)} \right\|_2^2$$

2nd nearest neighbor

$$nn_2(\boldsymbol{x}) = \underset{i \in [n] - nn_1(\boldsymbol{x})}{\operatorname{argmin}} \left\| \boldsymbol{x} - \boldsymbol{x}^{(i)} \right\|_2^2$$

3rd nearest neighbor

$$nn_3(\boldsymbol{x}) = \underset{i \in [n] - nn_1(\boldsymbol{x}) - nn_2(\boldsymbol{x})}{\operatorname{argmin}} \left\| \boldsymbol{x} - \boldsymbol{x}^{(i)} \right\|_2^2$$

The set of k nearest neighbors

$$knn(\boldsymbol{x}) = \{nn_1(\boldsymbol{x}), nn_2(\boldsymbol{x}), \cdots, nn_k(\boldsymbol{x})\}$$

k-Nearest Neighbor (kNN) classification

Classification rule

- Every neighbor votes
- Use majority vote

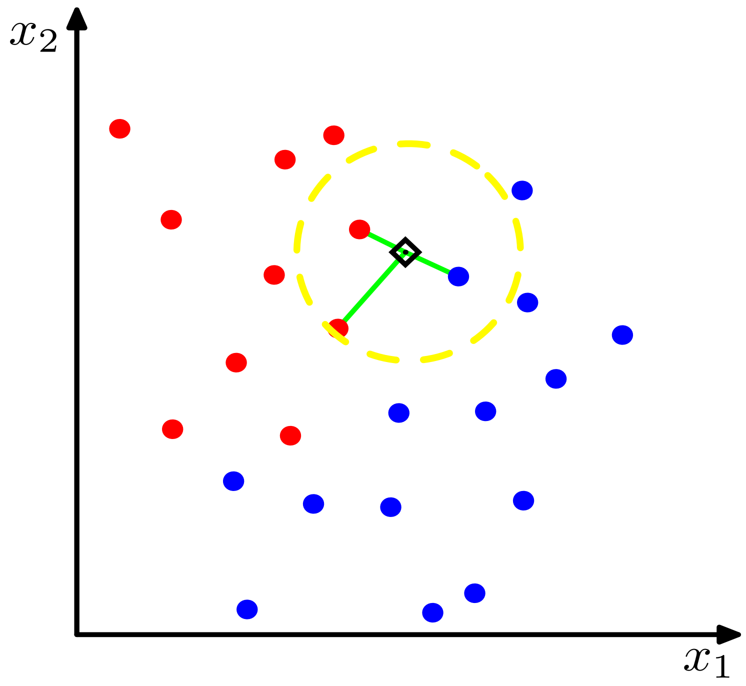
$$h(x) = \text{majority}\{y_{nn_1(\boldsymbol{x})}, y_{nn_2(\boldsymbol{x})}, \cdots, y_{nn_k(\boldsymbol{x})}\}$$

Note:

- kNN assumes all features are equally useful for classification
- Scale of measurement matters
  - Different nearest neighbors if you have {petal width, sepal length} vs {5 petal width, sepal length}

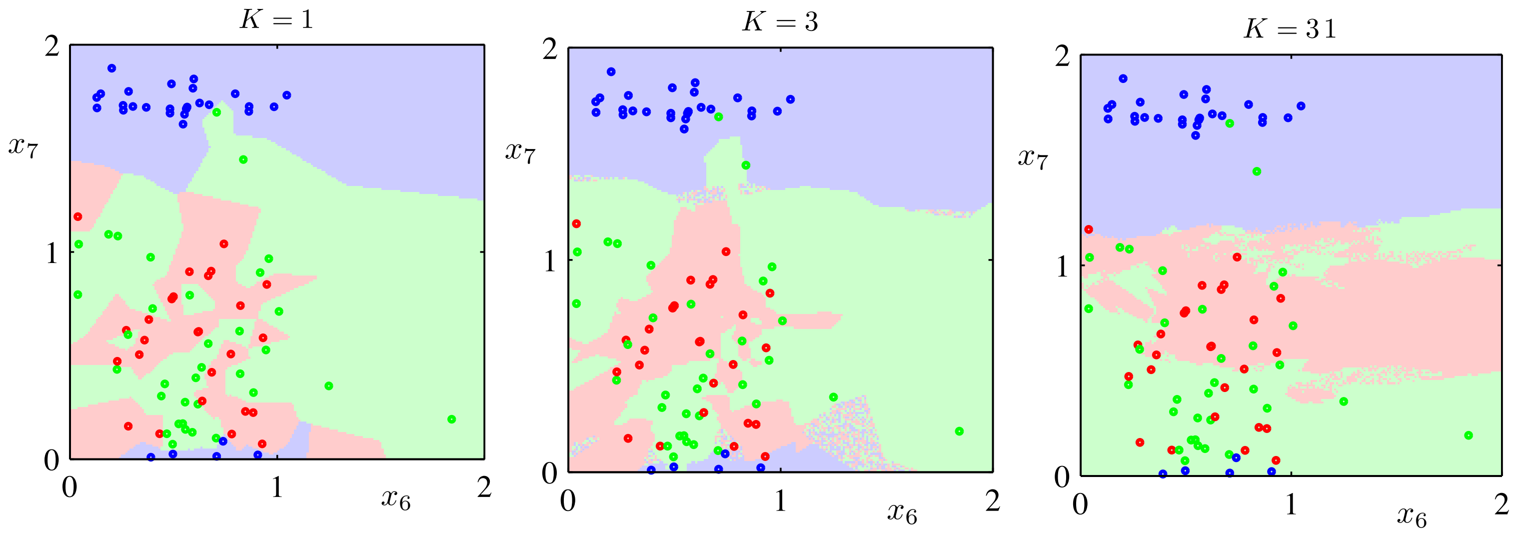
Example

$k = 1$, Label: Red $\quad k = 3$, Label: Red $\quad k = 5$, Label: Blue



How to choose an optimal k? Treat as hyperparameter and validate

Voronoi Diagrams for kNN



- Assign colors to training points based on their class labels
- For every other point, color it with the same color as its k nearest neighbors

Summary

Advantages of kNN

- conceptually simple and easy to implement – just computing the distance
- **non-parameteric** (no parameters, no optimization needed)

Disadvantages

Computationally intensive for large-scale (high dimensionality, high training instances problems

- $O(nd)$ for labeling a data point

Memory intensive

- Need to store the training data even during testing

Needs a suitable notion of distance for computing nearest neighbors

- not always easy for certain data types, e.g., images