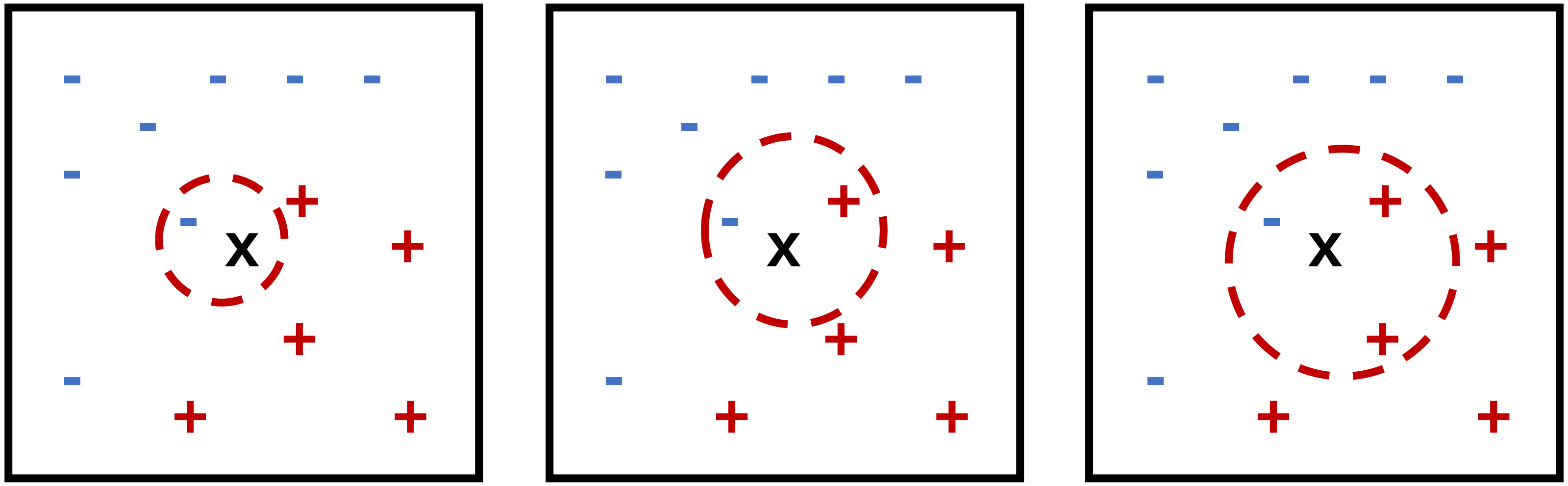
Classification:

More Classification Techniques

CS 418. Introduction to Data Science

k-Nearest Neighbors Classifier Preliminaries

- A **k-nearest neighbors** classifier assigns class labels to observations based on the class labels of the k "**most similar**" observations (**k nearest neighbors**).
- **"If it walks like a duck, quacks like a duck, and looks like a duck, then it's probably a duck."**
- ***Example***:



1-Nearest Neighbor **2-Nearest Neighbors** **3-Nearest Neighbors**

k-Nearest Neighbors Classifier Algorithm

- **Algorithm for k-nearest neighbors classifier:**
  1. Choose a **proximity metric** and a **number of nearest neighbors k**.
  2. For each observation z in the test set:
     1. Compute the **distance** or **similarity** between z and every observation in the training set.
     2. Select the k observations **most similar** to z (**k nearest neighbors**).
     3. Assign a class label to z based on the class labels of its **k nearest neighbors**.
        - Use **majority voting** or **weighted majority voting**.
        - If there is a **tie** between classes, randomly choose one of them.
- **How to choose k?**
  - If k is **too small**, then the classifier may be susceptible to **overfitting** due to noise.
  - If k is **too large**, then the classifier may misclassify z by considering observations that are **not similar** to z.

k-Nearest Neighbors Classifier Distance Metrics

- **Distance metrics** measure the **dissimilarity** between two observations.
  - For **continuous** attributes:
    - **Euclidean distance:**

      $$d(x, y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

      where p is the number of attributes and x_i and y_i are the ith attributes of observations x and y, respectively.
    - **Manhattan distance:**

      $$d(x, y) = \sum_{i=1}^{p} |x_i - y_i|$$

    - The **Euclidean** and the **Manhattan** distances can be generalized by the **Minkowski distance**:

      $$d(x, y) = \left(\sum_{i=1}^{p} |x_i - y_i|^r\right)^{1/r}$$

      where r is a parameter.

k-Nearest Neighbors Classifier
Similarity Metrics

- **Similarity metrics** measure the **similarity** between two observations.
  - For **binary** attributes:
    - **Jaccard coefficient:**

$$S(x,y) = \frac{N_{11}}{N_{01} + N_{10} + N_{11}}$$

    - **Simple matching coefficient:**

$$S(x,y) = \frac{N_{00} + N_{11}}{N_{00} + N_{01} + N_{10} + N_{11}}$$

      where N_{11} is the number of attributes where both x and y have a value of 1, N_{00} is the number of attributes where both x and y have a value of 0, N_{01} is the number of attributes where x has a value of 0 and y has a value of 1, and N_{10} is the number of attributes where x has a value of 1 and y has a value of 0.

- For more **distance** and **similarity metrics** available on Python, see `sklearn.neighbors.DistanceMetric`.

k-Nearest Neighbors Classifier Advantages and Disadvantages

- What are some of the **advantages** of ***k*-nearest neighbors**?
  - Does not require training a model.
  - **Nonparametric**. Makes no assumptions about the probability distribution of the data.
  - Applicable to **categorical and numeric attributes**.
  - **Flexibility**. Can produce decision boundaries of any shape.
- What are some of the **disadvantages** of ***k*-nearest neighbors**?
  - Classifying each observation from the test set can be **computationally expensive**.
  - **Easy to overfit** due to noise or high dimensionality.
  - **Sensitive to redundant or irrelevant attributes**.
  - **Sensitive to the scale** of the data.
  - **No descriptive value**. Does not provide a model of the relationship between the class label and the attributes.

Naïve Bayes Classifier Preliminaries (I)

- A **Naïve Bayes classifier** is a **probabilistic model** that uses **Bayes' Theorem** for classification.

The **probability** of observing class label y given attribute value X is

$$P(y|X) = \frac{P(X|y)\,P(y)}{P(X)} = \frac{P(X|y)\,P(y)}{P(X|y)\,P(y) + P(X|\neg y)\,P(\neg y)}$$

- ***Example***: Suppose that we want to classify emails as **spam** or **non-spam** based on the appearance of the word "free" in the email. Using **Bayes' Theorem** we can compute:

$$P(spam|free) = \frac{P(free|spam)\,P(spam)}{P(free|spam)\,P(spam)+P(free|\neg spam)\,P(\neg spam)}$$

$$P(\neg spam|free) = \frac{P(free|\neg spam)\,P(\neg spam)}{P(free|spam)\,P(spam)+P(free|\neg spam)\,P(\neg spam)}$$

If $P(spam|free) > P(\neg spam|free)$, we classify the email as **spam**.

Naïve Bayes Classifier Preliminaries (II)

- **What if we have more than one attribute?**

$$P(y|X_1, X_2, \dots, X_p) = \frac{P(X_1, X_2, \dots, X_p|y)\, P(y)}{P(X_1, X_2, \dots, X_p)}$$

- Note that:

$$P(X_1, X_2, \dots, X_p|y) = \frac{P(X_1, X_2, \dots, X_p, y)}{P(y)}$$

$$= \frac{P(X_1, X_2, \dots, X_p, y)}{P(X_2, X_3, \dots, X_n, y)} \cdot \frac{P(X_2, X_3, \dots, X_p, y)}{P(X_3, X_4, \dots, X_p, y)} \cdot \dots \cdot \frac{P(X_p, y)}{P(y)}$$

$$= P(X_1|X_2, \dots, X_p, y)\, P(X_2|X_3, \dots, X_p, y) \dots P(X_p|y)$$

Let X_1, X_2, and Y be random variables. X_1 is **conditionally independent** of X_2 given Y if

$$P(X_1|X_2, Y) = P(X_1|Y)$$

Naïve Bayes Classifier Assumption

- **Naïve Bayes classifiers** assume that **all attributes are conditionally independent**. That is, if the class label $\boldsymbol{y}$ is known, then we consider the attributes to be **independent of each other**. That is:

$$P(X_1, X_2, \dots, X_p | y) = P(X_1|y)\, P(X_2|y) \dots P(X_p|y) = \prod_{i=1}^{p} P(X_i|y)$$

- Then, the probability of observing a class label $\boldsymbol{y}$ is given by:

$$P(y|X_1, X_2, \dots, X_p) = \frac{P(y) \prod_{i=1}^{p} P(X_i|y)}{P(X_1, X_2, \dots, X_p)}$$

- Since $P(X_1, X_2, \dots, X_p)$ is constant for every class label, then:

$$P(y|X_1, X_2, \dots, X_p) \propto P(y) \prod_{i=1}^{p} P(X_i|y)$$

- Thus, it is sufficient to choose the class label that **maximizes** $P(y) \prod_{i=1}^{p} P(X_i|y)$.

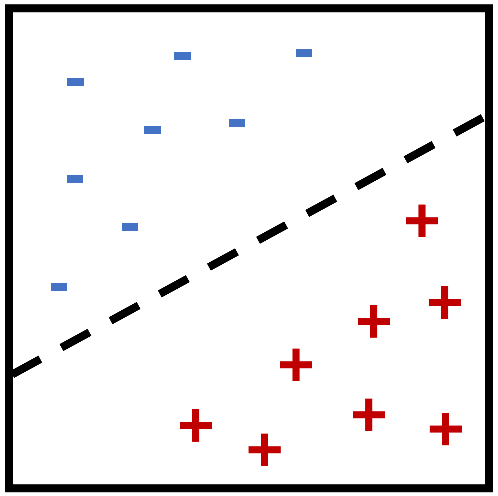
Naïve Bayes Classifier
Advantages and Disadvantages

- What are some of the **advantages** of **Naïve Bayes**?
  - **Descriptive value**. Provides a probabilistic model of the relationship between the class label and the attributes, as well as probabilities that quantify the uncertainty in predictions.
  - Applicable to **categorical and numeric attributes**.
  - **Robust to irrelevant attributes**.
  - **Robust to noise**.
- What are some of the **disadvantages** of **Naïve Bayes**?
  - **Sensitive to redundant attributes**.
  - Assumes that **attributes are conditionally independent**. This assumption often does not hold in practice.
- We can represent other forms of conditional independence among attributes using **Bayesian Networks**, which are **probabilistic graphical models** that do not assume that attributes are conditionally independent.

Support Vector Machines Preliminaries (I)

- A **hyperplane** is given by the equation $\mathbf{w}^T\mathbf{x} + b = 0$ where $\mathbf{x}$ are the attribute values and $\mathbf{w}$ and b are the parameters of the **hyperplane**.
- An observation can belong to either side of the **hyperplane** depending on the sign of $\mathbf{w}^T\mathbf{x} + b$.
- A dataset is **linearly separable** if there exists a **hyperplane** that can perfectly separate observations from different classes.

if $\mathbf{w}^T\mathbf{x}_i + b > 0$, then $y_i = -$



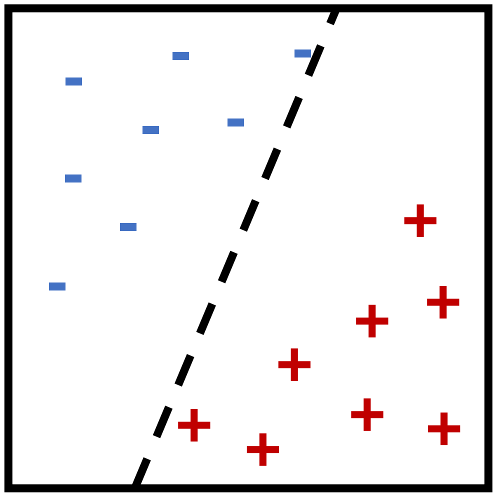
if $\mathbf{w}^T\mathbf{x}_i + b < 0$, then $y_i = +$

Classification Exercise 14.1

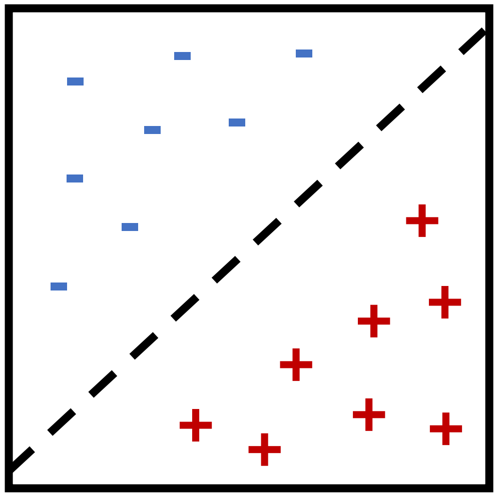
THINK PAIR SHARE

There can be infinitely many hyperplanes to separate the classes.

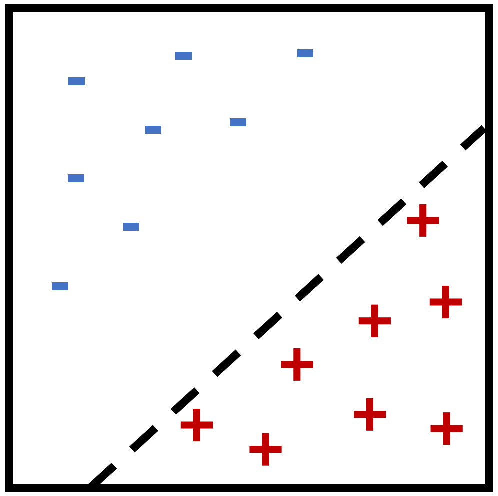
Which of these hyperplanes would you choose? Why?



Option A



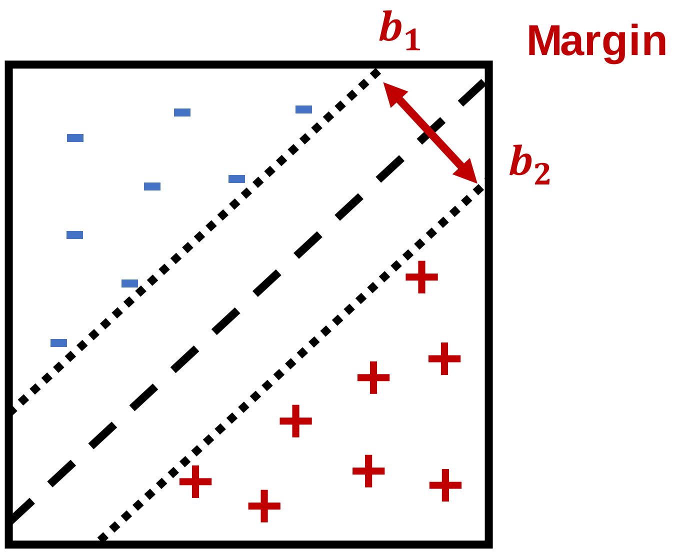
Option B



Option C

Support Vector Machines
Preliminaries (II)

- We would like to choose a **hyperplane** that is **robust to small perturbations in the data**. This can be achieved by choosing the **hyperplane** with the **maximum margin**.
- For every **hyperplane** B, there is a pair of **parallel hyperplanes** b_1 and b_2 that touch the **closest observations** of both classes.
- The **distance between b_1 and b_2** is known as the **margin** of **hyperplane** B.



What happens if the margin is too small?

Support Vector Machines
Definition

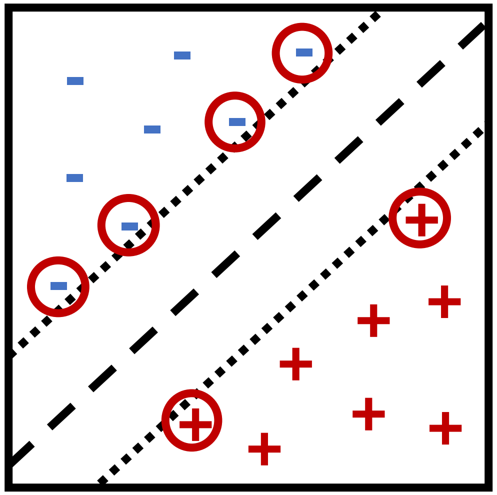
- A **support vector machine** (**SVM**) is a classifier that finds the **separating hyperplane** with the **maximum margin** by solving the following **optimization problem**:

$$\max_{\mathbf{w},b} M$$
$$\text{subject to } y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq M||\mathbf{w}||$$

  where $\mathbf{w}$ and b are the parameters of the **hyperplane**, $2M$ is the margin, and $y_i \in \{-1, 1\}$.

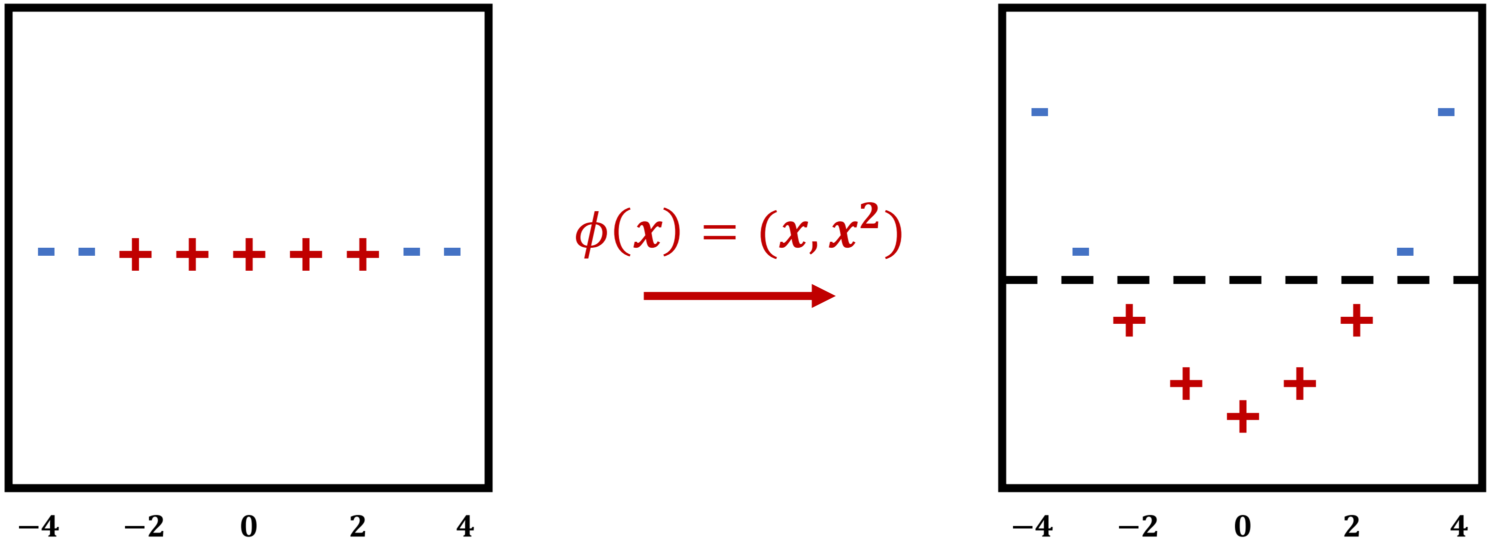
- **SVMs** represent this **hyperplane** using only a subset of the observations in the training set that are **most difficult to classify** (known as the **support vectors**).

- If the number of **support vectors** is **too large**, the **SVM** is more likely to **overfit**.



Support Vector Machines
Nonlinear SVM

- The previous definition of **SVM** learns a **linear decision boundary** to separate the classes.
- We can learn **nonlinear decision boundaries** by transforming the data from the original attribute space $\mathbf{x}$ into a new space $\phi(\mathbf{x})$ where a **linear hyperplane** can be used to separate the classes.



Not linearly separable

Linearly separable

Support Vector Machines
Kernel Trick

- Computing the **transformation** $\phi(\mathbf{u})$ for every observation $\mathbf{u}$ can be **computationally expensive**.
- However, in order to solve the **SVM** optimization problem, we only need the **inner products** $\langle\phi(\mathbf{u}), \phi(\mathbf{v})\rangle$.
- Therefore, we can use a **kernel function** $K(\mathbf{u}, \mathbf{v})$ to compute the **inner products** $\langle\phi(\mathbf{u}), \phi(\mathbf{v})\rangle$ without explicitly computing the **transformations** $\phi(\mathbf{u})$ and $\phi(\mathbf{v})$.

$$K(\mathbf{u}, \mathbf{v}) = \langle\phi(\mathbf{u}), \phi(\mathbf{v})\rangle = \boldsymbol{f}(\mathbf{u}, \mathbf{v})$$

- This method is known as the **kernel trick**.
- Some commonly used **kernel functions** include:
  - **Polynomial kernel:** $K(\mathbf{u}, \mathbf{v}) = \left(\mathbf{u}^T\mathbf{v} + 1\right)^p$
  - **Radial Basis Function (RBF) kernel:** $K(\mathbf{u}, \mathbf{v}) = e^{-\left|\left|\mathbf{u}-\mathbf{v}\right|\right|^2/(2\sigma^2)}$

Support Vector Machines
Advantages and Disadvantages

- What are some of the **advantages** of **SVMs**?
  - Uses optimization algorithms to find the **optimal decision boundary**.
  - **Flexibility.** Can learn both linear and nonlinear decision boundaries.
  - Applicable to **numeric and categorical attributes** (by using dummy variables).
  - **Generally robust to irrelevant attributes** (by assigning them a weight of zero).
  - **Generally robust to redundant attributes** (by assigning them a similar weight).
- What are some of the **disadvantages** of **SVMs**?
  - Training the model can be **computationally expensive**.

Classification
References

- Joel Grus. *Data Science from Scratch* (2015).
- Cathy O'Neil and Rachel Schutt. *Doing Data Science* (2013).
- Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin Kumar. *Introduction to Data Mining* (2019).