Lecture 8: Black Box Methods: Support Vector Machines and Neural Networks

Outline

- Understanding Neural Networks
  - From biological to artificial neurons
  - Activation functions
  - Network topology
  - Training neural networks with backpropagation
- Understanding Support Vector Machines
  - Classification with hyperplanes
  - Using kernels for non-linear spaces
- Evaluation Metrics for Numeric Prediction

Black Box Methods

- Black box methods in data mining refer to a pair of methods that are extremely powerful, while their inner workings can be difficult to understand.
  - The mechanism that transforms the input into the output is obfuscated by an imaginary box.
  - Complex mathematics allowing them to function.
- Although they may not be easy to understand, it is dangerous to apply black box models blindly.

Black Box Methods

- Despite their complexity, black box methods can be applied easily to real-world problems.
  - How neural networks mimic the structure of animal brains to model arbitrary functions
  - How support vector machine use multidimensional surfaces to define the relationship between features and outcomes

Neural Networks

Understand Neural Networks

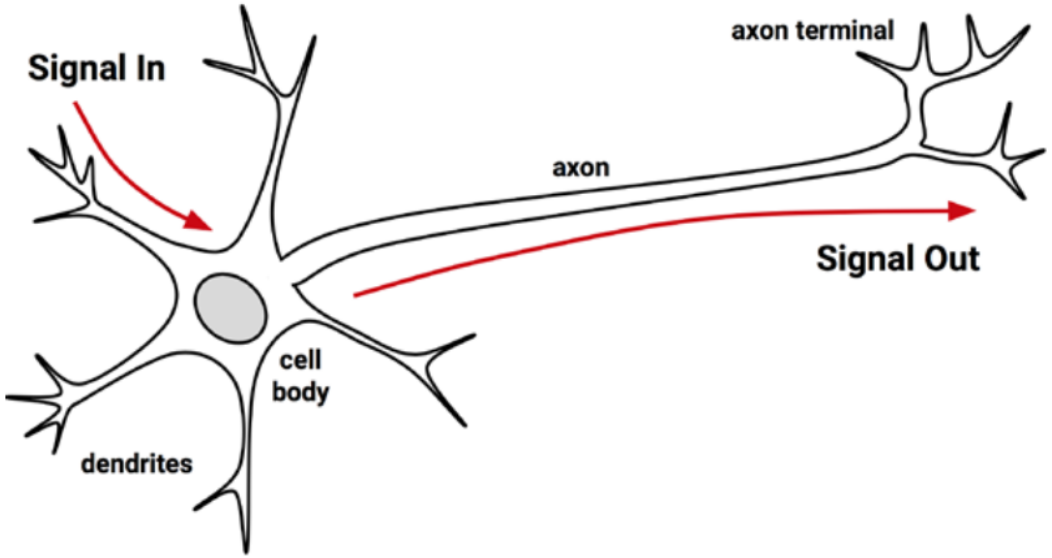
- Neural networks, are computational models inspired by biological neural networks, and are used to approximate functions that are generally unknown. Particularly, they are inspired by the behavior of neurons and the electrical signals they convey between input, processing, and output from the brain.

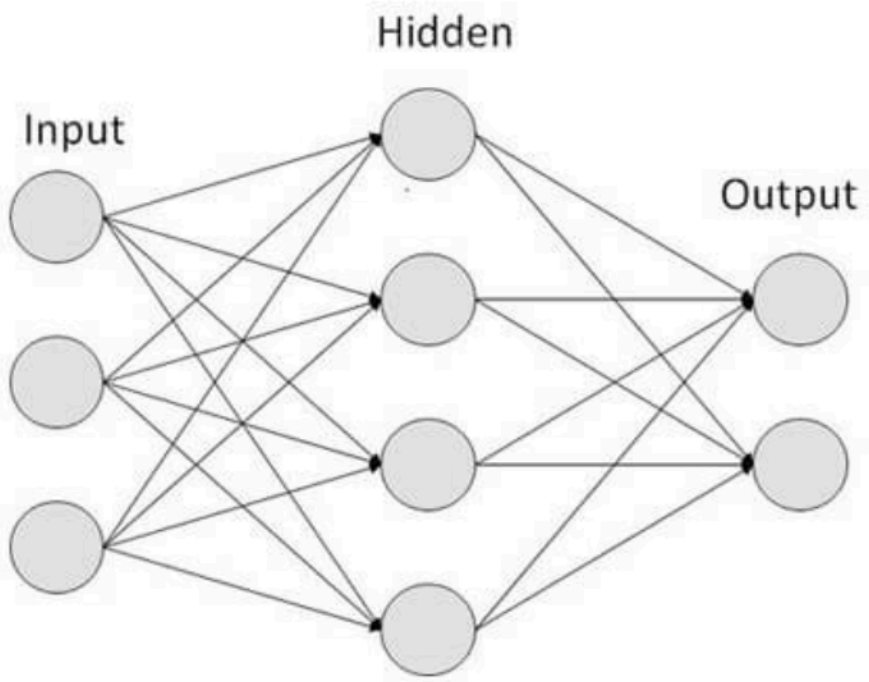
Understand Neural Networks

- An Artificial Neural Network (ANN) models the relationship between a set of input signals and an output signal
  - Derived from our understanding of how a biological brain responds to stimuli from sensory inputs
  - A brain uses a network of interconnected cells called **neurons** to create a massive parallel processor
  - ANN uses a network of **artificial neurons** or **nodes** to solve learning problems
- ANNs are versatile learners that can be applied to nearly any learning tasks
  - Classification, numeric prediction, and even unsupervised pattern recognition

From Biological to Artificial Neurons

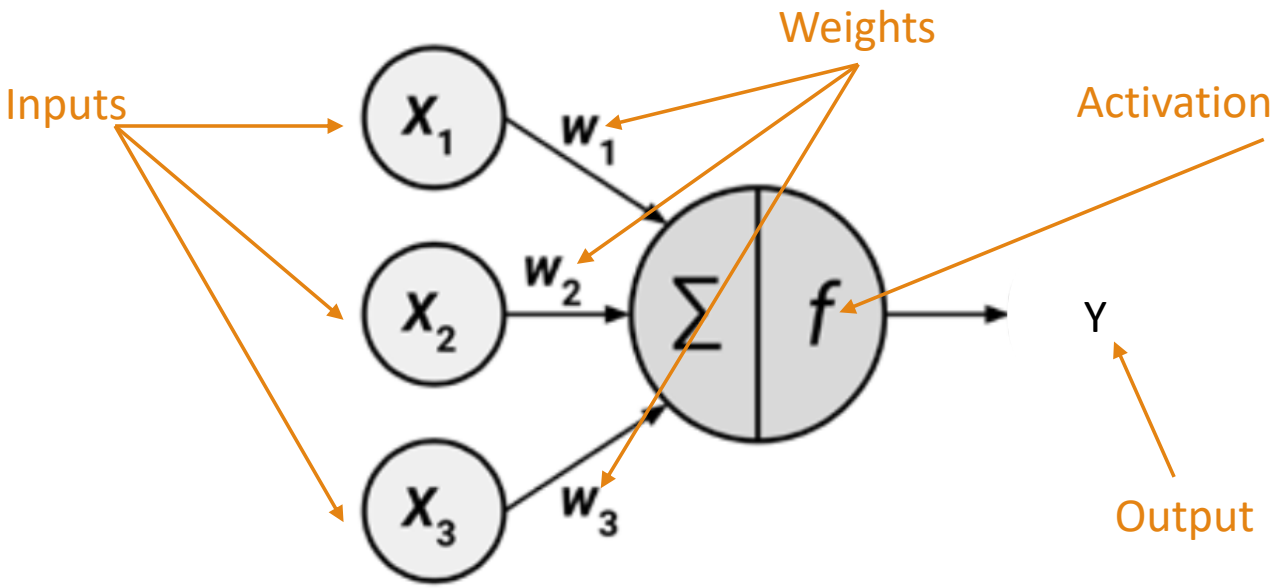
- Nature has inspired millions of innovations-the human brain has inspired the creation of neural networks.





From Biological to Artificial Neurons

- The model of a single artificial neuron can be understood in terms very similar to the biological model.



$$y(x) = f\left(\sum_{i=1}^{n} w_i x_i\right)$$

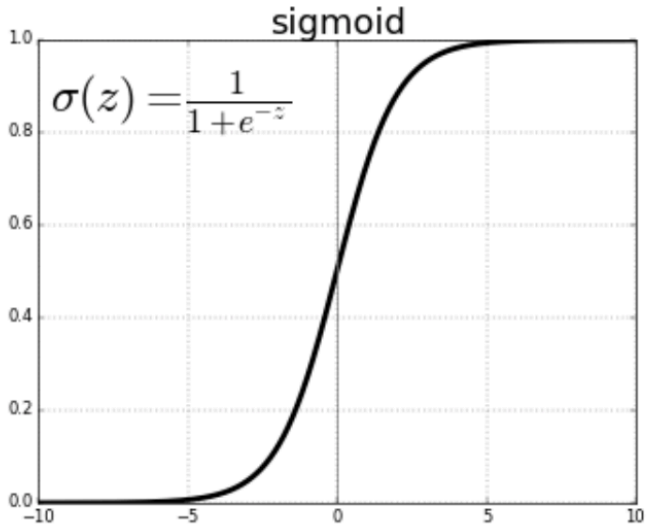
From Biological to Artificial Neurons

- **Neural networks** use **neurons** as building blocks to construct complex models of data.
- Characteristics of **neural networks**:
  - An **activation function**, which transforms a neuron's combined input signals into a single output signal to be broadcasted further in the network
  - A **network topology** (or architecture), which describes the number of neurons in the model as well as the number of layers and manner in which they are connected
  - The **training algorithm** that specifies how connection weights are set in order to inhibit or excite neurons in proportion to the input signal

Activation Functions

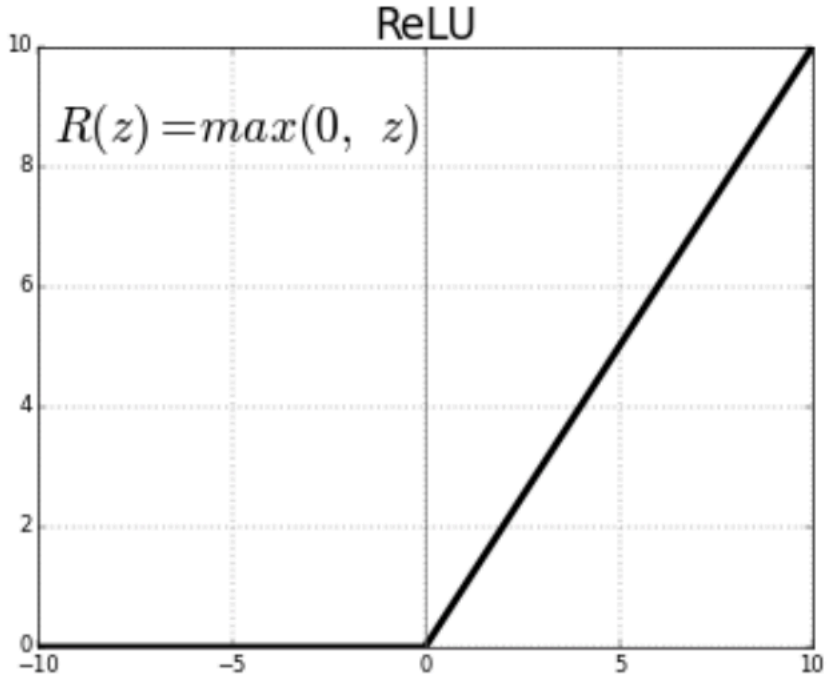
- Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it.
  - The purpose of the activation function is to **introduce non-linearity** into the output of a neuron.

Issues with Sigmoid activation functior
- Vanishing gradient
- Output is not zero centered



Activation Functions

- ReLU (Rectified Linear Unit) Activation Function
  - The ReLU is the most used activation function right now



Network Topology

- The ability of a neural network to learn is rooted in its **topology**, or the patterns and structures of interconnected neurons.
- The topology determines the complexity of tasks that can be learned by the network.
  - The number of layers
  - The number of nodes within each layer of the network
  - Whether information in the network is allowed to travel backward

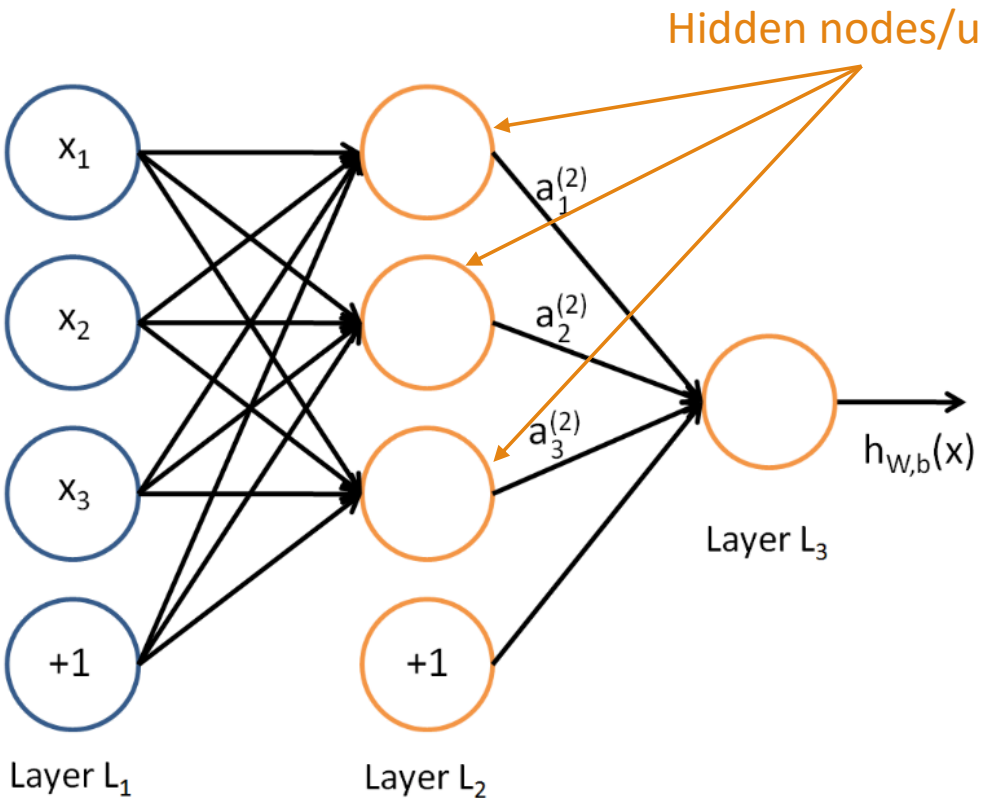
Network Topology

A neural network is put together by hooking together many of our simple "neurons," so that the output of a neuron can be the input of another.

- Leftmost layer is called **input layer**
- Rightmost layer is called **output layer**
- Middle layer of nodes is called **hidden layer**, since the value are not observed in the training set.
- The circles labeled "+1" are called **bias units**, and correspond to the intercept term.

$$a_1^{(2)} = f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)})$$
$$a_2^{(2)} = f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)})$$
$$a_3^{(2)} = f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)})$$
$$h_{W,b}(x) = f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)})$$

$f()$ is the **activation function**



3 layer structure

Network Topology

- Neural networks can in complexity by the number of layers and the number of nodes in each layer.
  - The number of input nodes is predetermined by the number of features in the input data.
  - The number of output nodes is predetermined by the number of outcomes to be modeled or the number of class levels in the outcome.
  - The number of hidden nodes is left to the user to decide prior to training the model.

Network Topology

- There is no reliable rule to determine the number of neurons in the hidden layer.
  - The appropriate number depends on the number of input nodes, the amount of training data, the amount of noisy data, and the complexity of the learning task, among many other factors.
  - The best practice is to use the fewest nodes that result in adequate performance in a validation dataset.
  - A greater number of neurons will result in a model that more closely mirrors the training data, but this runs a risk of **overfitting**. It may generalize poorly to future data. Large neural networks can also be computationally expensive and slow to train.

Network Topology

- Generally, larger and more complex (more layers & more nodes in each layer) networks are capable of identifying more subtle patterns and complex decision boundaries.
  - The number of layers
  - The number of nodes within each layer of the network
- However, the power of a network is not only a function of the network size, but also the way units are arranged.
  - Whether information in the network is allowed to travel backward

Network Topology

- Networks in which the input signal is fed continuously in one direction from connection to connection until it reaches the output layer are called **feedforward** networks.
  - A simple example of feedforward neural network is Multilayer perceptron (MLP)

Network Topology

Multilayer perceptron (MLP)

- Consist of multiple layers of nodes in a directed graph
- Each layer fully connected to the next one
- Except the input nodes, each node is a neuron with nonlinear activation function
- Supervised learning with backpropagation

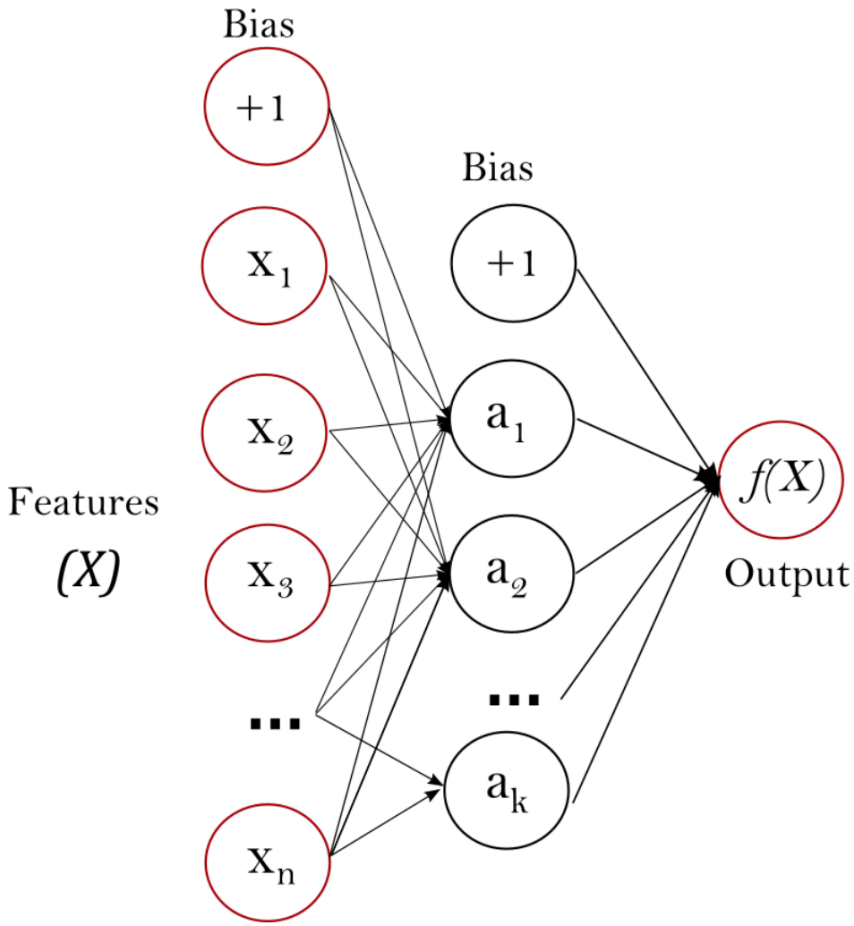
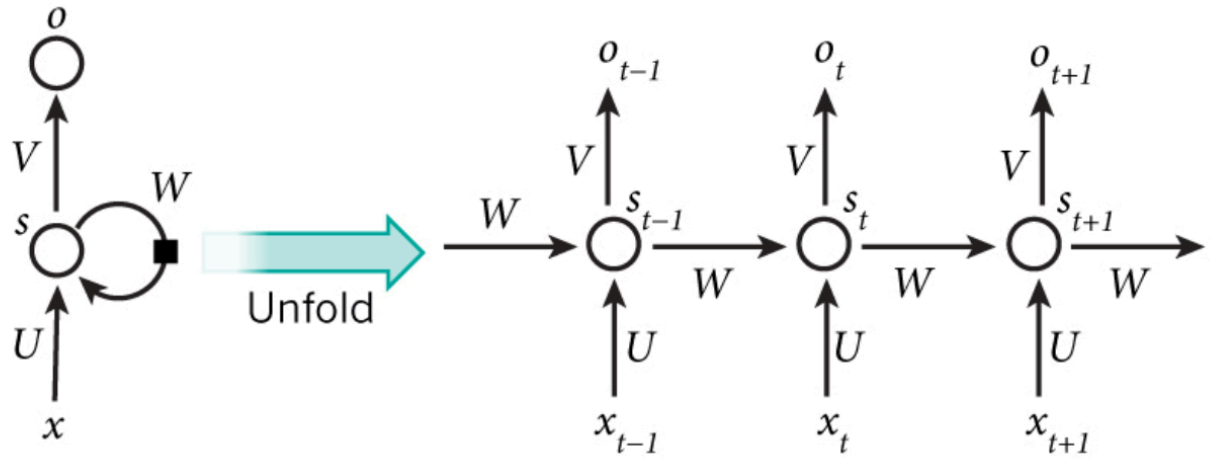


Figure 1 : One hidden layer MLP.

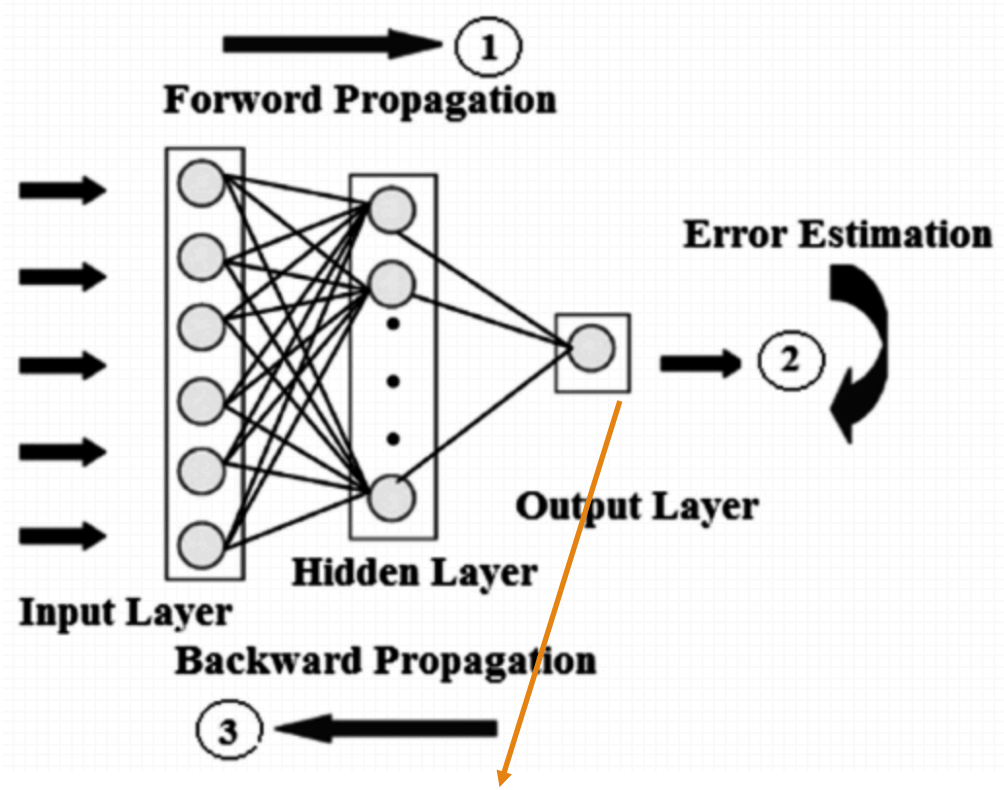
Network Topology

- In contrast, a **recurrent network** (or **feedback network**) allows signals to travel in both directions using loops.
  - The idea behind RNN is to make use of sequential information.
  - Before we assume all inputs are independent of each other.
  - RNN performs the same task for every element of sequence, with the output being depended on the previous computations.
  - Have the "memory" which captures information about what has been calculated so far.



Training Neural Networks with Backpropagation

- **Feedforward and backward propagation**
- **Input *x***: Set of input variables
- **Feedforward**: Information flows from the input layer to the output layer
- **Output error**: Compute the error with the cost function of the output value and real value
- **Backpropagate the error**: Calculate the error for each neuron from the output layer to the input layer
- **Update parameters**: Adjust the weight according to the backward propagation error



- For classification, ANN outputs class distribution
- For numeric prediction, ANN outputs predictive number

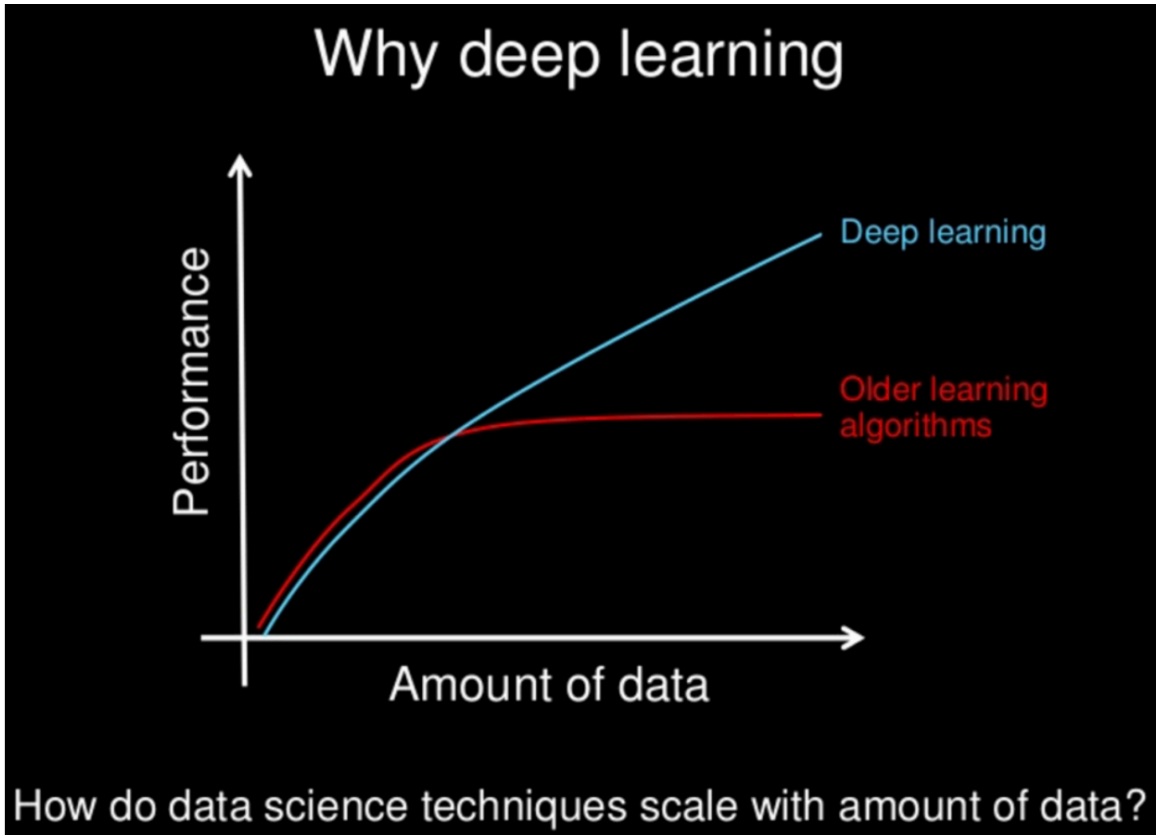
When Might ANN Be a Good Choice?

- Never hurt to try it with your application
- Studies have reported the following conditions under which ANN has outperformed other methods
  - Sufficient time and processing power to train and tune ANN models
  - Stake is high – high benefits of model accuracy – e.g., mission critical, high-prize competition
  - High-dimensional input, e.g., image, video, audial, textual input, big data
  - Possibly noisy data
  - Model interpretability is not necessary, e.g., for automated applications of dynamically learned models without human interventions
- ANN can be used for both classification and numeric prediction.

When Might ANN Be a Good Choice?

- Advantages of deep learning:
  - Scalability (see picture)
  - Automatic feature selection

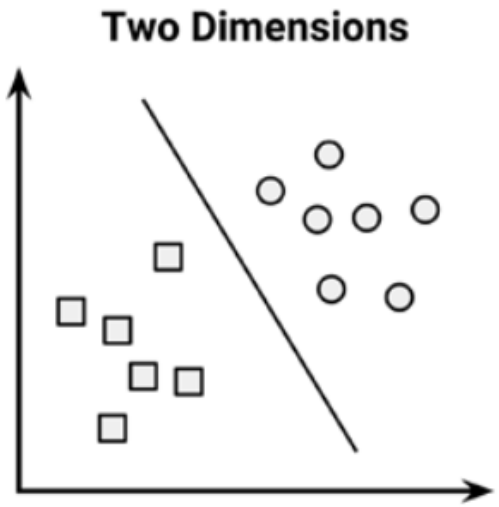
- Results get better with
  - More data
  - Bigger models
  - More computation

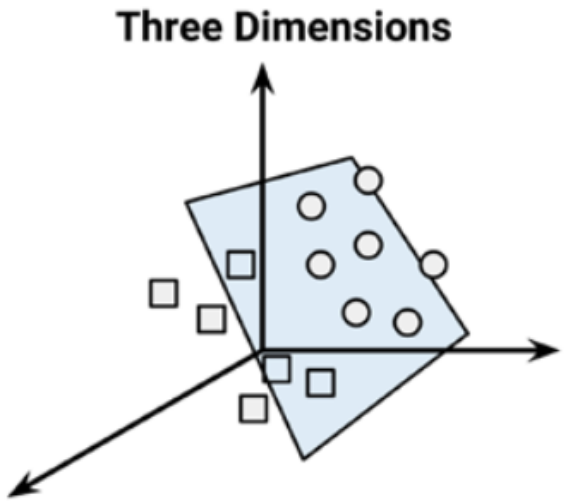


Support Vector Machines

Understanding Support Vector Machines

- A Support Vector Machine (SVM) can be imagined as a surface that creates a boundary between points of data plotted in multidimensional that represent examples and their feature values.
- The goal of a SVM is to create a flat boundary called a hyperplane, which divides the space to create fairly homogeneous partitions on either side.





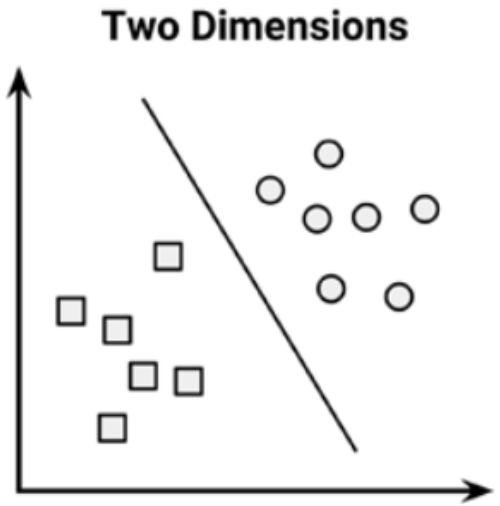
Understanding Support Vector Machines

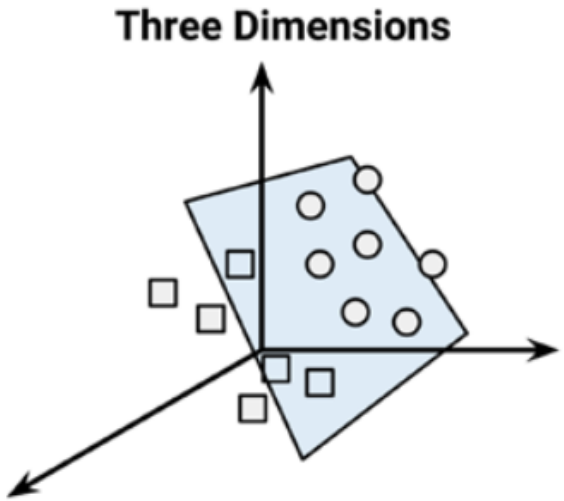
- SVMs can be adapted for use with nearly any type of learning task, including both classification and numeric prediction.
  - SVMs are most easily understood when used for binary classification, which is how the method has been traditionally applied.

Classification with Hyperplanes

- SVMs use a boundary called a hyperplane to partition data into groups of similar class values.

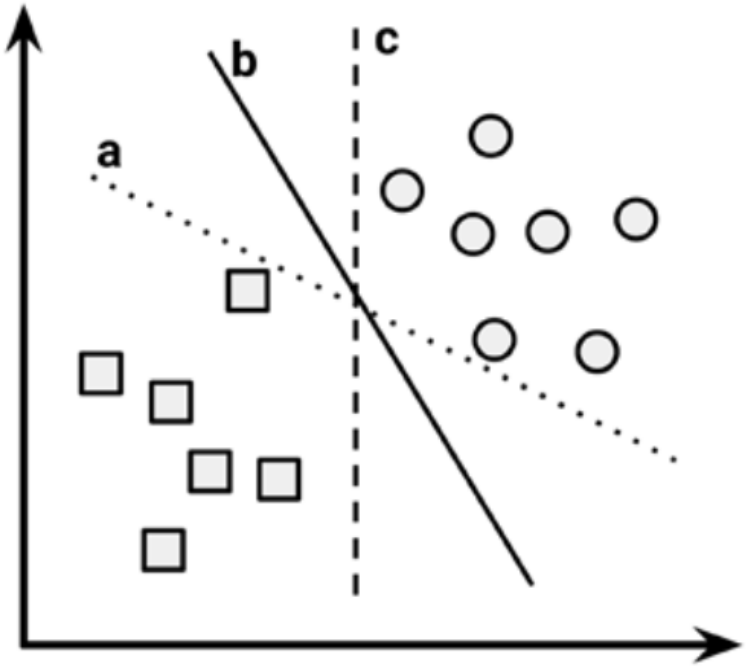
linearly separable: the circles and squares can be separated perfectly by the straight line or flat surface





Classification with Hyperplanes

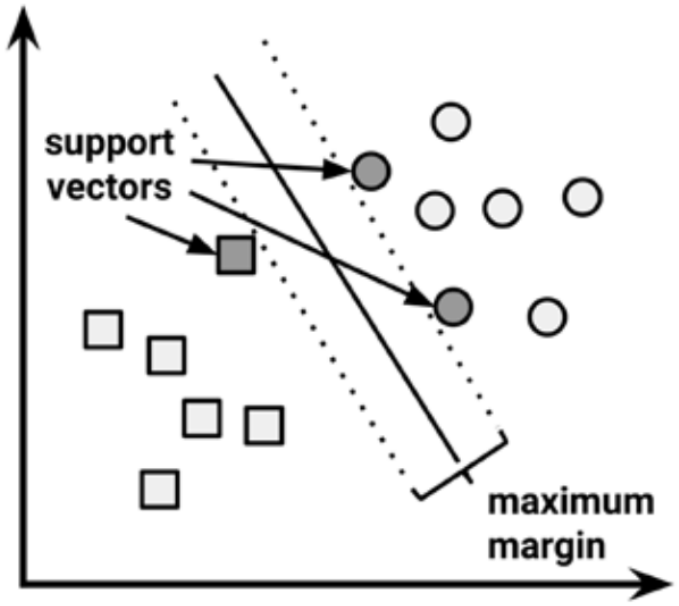
- In two dimensions, the task of the SVM algorithm is to identify a line that separates the two classes.



- There is more than one choice of dividing line between the groups of circles and squares.
- Three such possibilities are labeled a, b, and c.
- How does the algorithm choose?

Classification with Hyperplanes

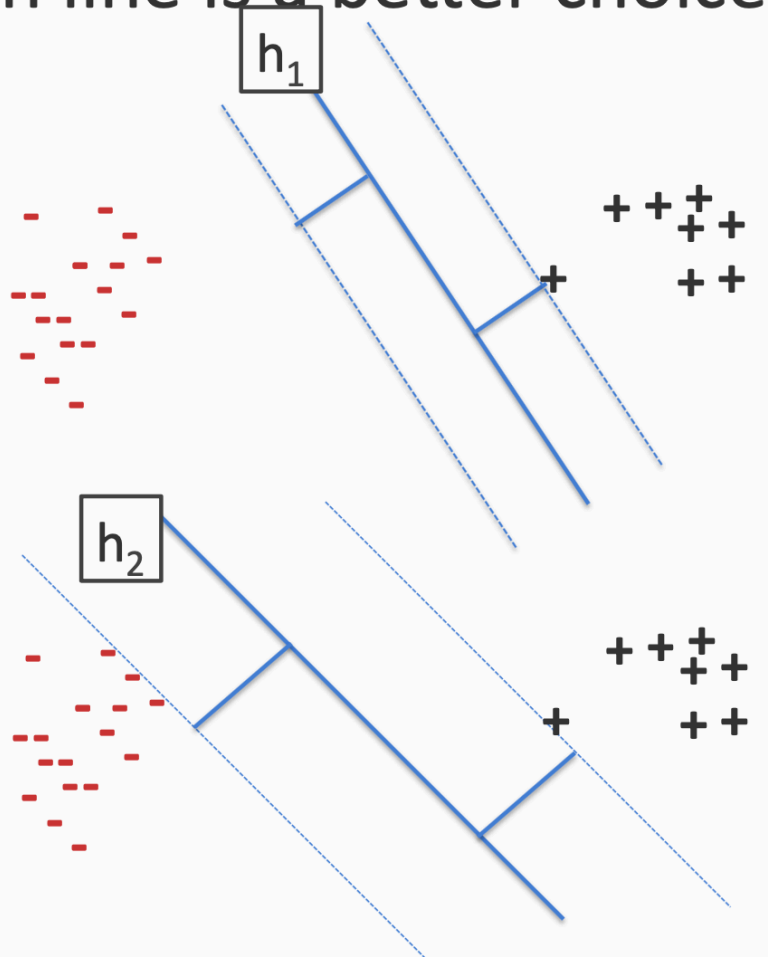
- Search for the **Maximum Margin Hyperplane** (MMH) that creates the greatest separation between the two classes.
  - The maximum margin will improve the chance that, in spite of random noise, the points will remain on the correct side of the boundary.



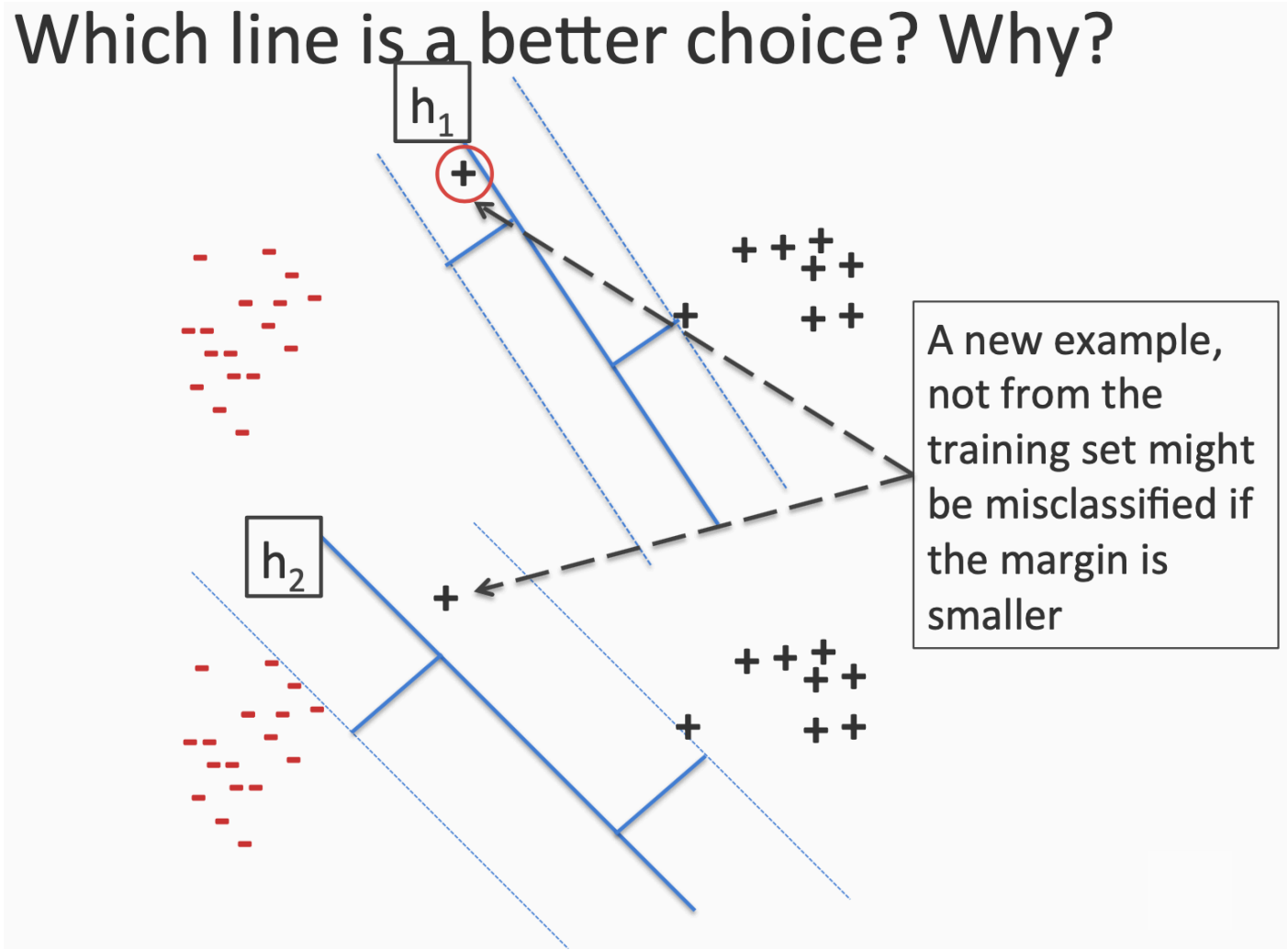
- The **support vectors** (indicated by arrows in the figure that follows) are the points from each class that are the closest to the MMH
- Each class must have at least one support vector, but it is possible to have more than one.
- Using the support vectors alone, it is possible to define the MMH.
- ***The margin of a hyperplane is the distance between the hyperplane and the data point nearest to it.***

Classification with Hyperplanes

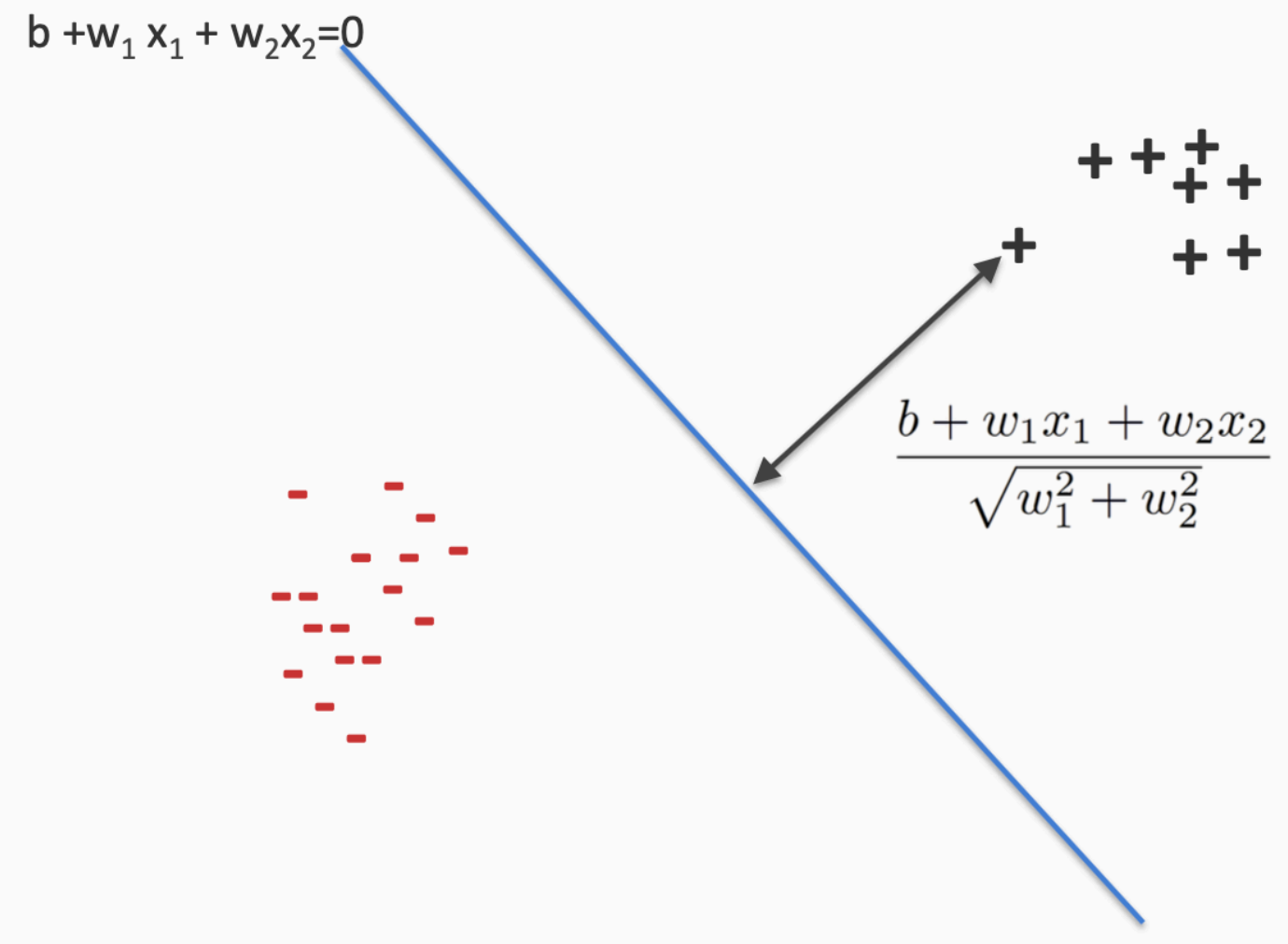
Which line is a better choice? Why?



Classification with Hyperplanes



Classification with Hyperplanes



- Learning strategy of SVM
  - Find the linear separator that maximizes the margin

$$\min_{\mathbf{w}} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w}$$
$$\text{s.t.} \forall i, \quad y_i\mathbf{w}^T\mathbf{x}_i \geq 1$$

Classification with Hyperplanes

- The case of nonlinearly separable data
  - A cost value (denoted as *C*) is applied to all points that "break into the margin"

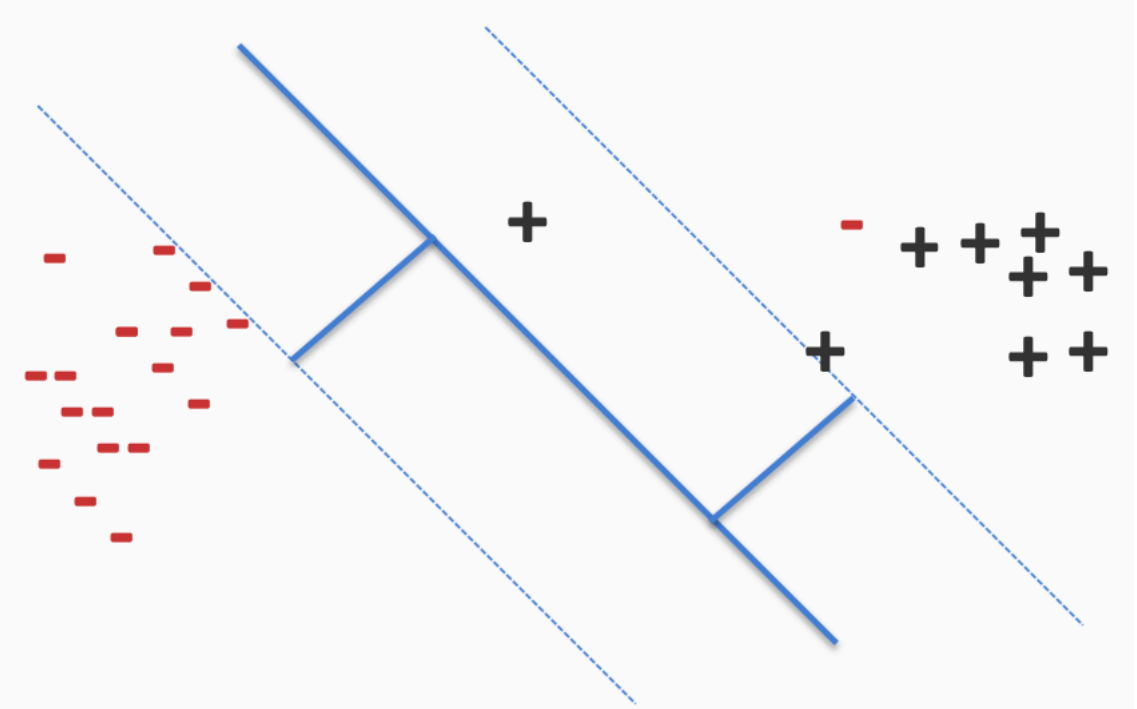
Maximize margin

Tradeoff between the two terms

$$\min_{\mathbf{w},\xi} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_i \xi_i$$

Minimize total slack (i.e allow as few examples as possible to violate the margin)

$$\text{s.t.} \quad \forall i, \quad y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i$$
$$\forall i, \quad \xi_i \geq 0.$$

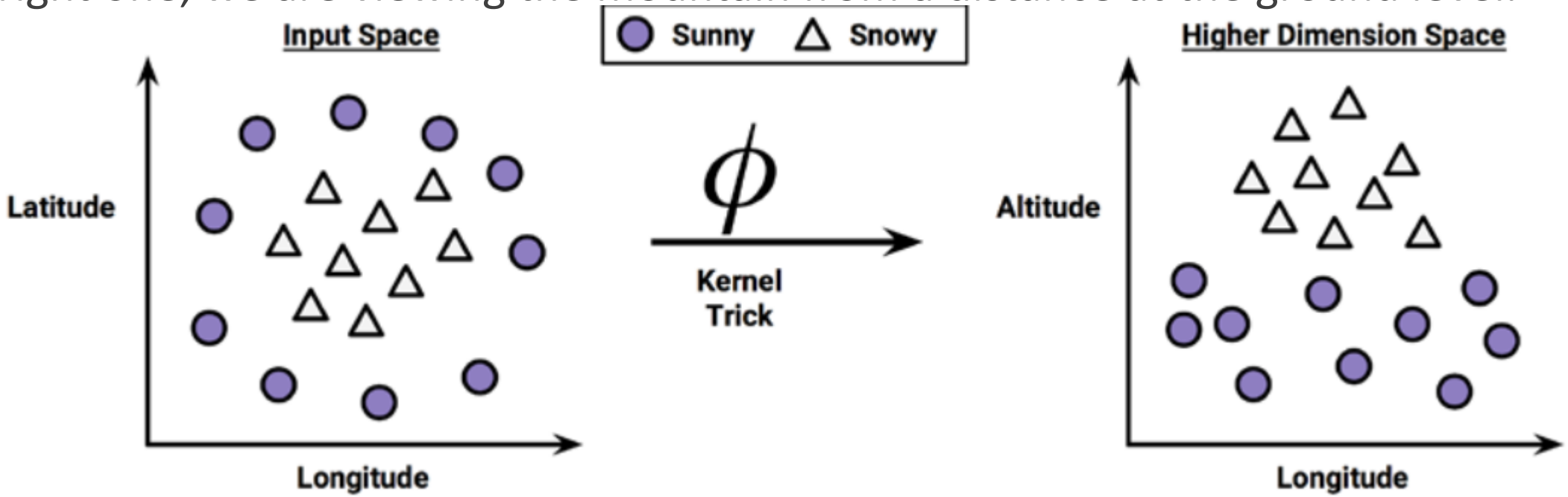


Classification with Hyperplanes

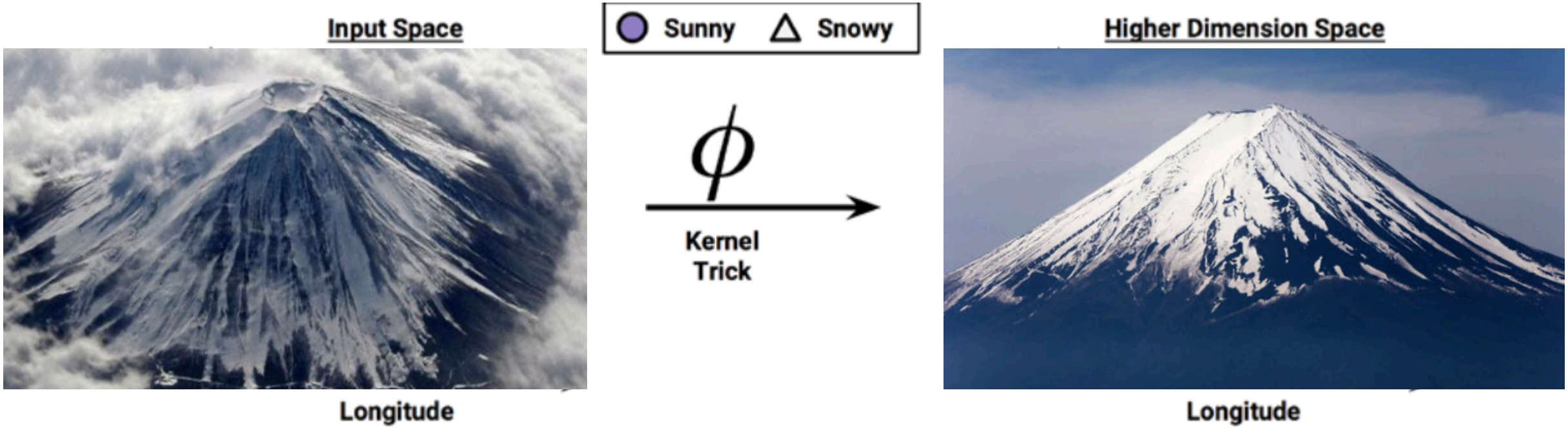
- In many real-world applications, the relationships between variables are nonlinear.
  - Another key feature of SVMs is their ability to map the problem into a higher dimension space using a process known as the **kernel trick**.

Using Kernels for Non-linear Spaces

- In the following figure, the scatterplot on the left depicts a nonlinear relationship between a weather class (sunny or snowy) and two features: latitude and longitude.
  - In the left figure, we are viewing the mountain from a bird's eye view, while in the right one, we are viewing the mountain from a distance at the ground level.



Using Kernels for Non-linear Spaces



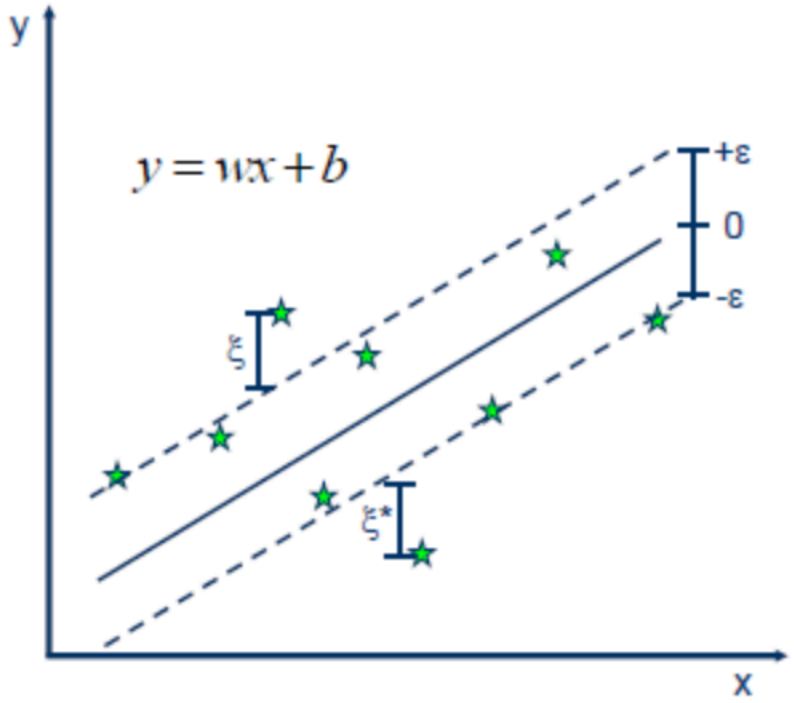
Using Kernels for Non-linear Spaces

- SVMs with nonlinear kernels add additional dimensions to the data in order to create separation in this way.
- The kernel trick involves a process of constructing new features that express mathematical relationships between measured characteristics.
  - For instance, the altitude feature can be expressed mathematically as an interaction between latitude and longitude—the closer the point is to the center of each of these scales, the greater the altitude.
- This allows SVM to learn concepts that were not explicitly measured in the original data.
  - Common kernels functions include the linear kernel, polynomial kernel, sigmoid kernel, and Gaussian RBF kernel.

Support Vector Regression

- The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences.
  - Output is a real number
  - A margin of tolerance (epsilon) is set in approximation to the SVM
  - To minimize error, individualizing the hyperplane which maximizes the margin

Support Vector Regression



- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon + \xi_i$$

$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$

It also has **kernel trick** to map the data into a higher dimension space.

Evaluation Metrics for Numeric Prediction

- **Mean Absolute Error (MAE)**
  - The **mean absolute error (MAE)** is a quantity used to measure how close forecasts or predictions are to the eventual outcomes.
  - This is known as a scale-dependent accuracy measure and therefore cannot be used to make comparisons between series using different scales

$$\mathrm{MAE} = \frac{1}{\mathrm{n}}\sum_{t=1}^{n}|y_t - \widehat{y_t}|$$

- **Root Mean Squared Error (RMSE)**
  - RMSE gives a relatively high weight to large errors. This means the RMSE should be more useful when large errors are particularly undesirable.

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{t=1}^{n}(y_t - \widehat{y_t})^2}{n}}$$

- **Mean absolute percentage error (MAPE)**
  - The **mean absolute percentage error** (**MAPE**) measures this accuracy as a percentage.
  - It cannot be used if there are zero values because there would be a division by zero.

$$\mathrm{MAPE} = \frac{1}{\mathrm{n}}\sum_{t=1}^{n}\left|\frac{y_t - \widehat{y_t}}{y_t}\right|$$

Not depend on the scale

- **Relative absolute error (RAE)**
  - The **relative absolute error** takes the total absolute error and normalizes it by dividing by the total absolute error of the mean estimator.

$$\mathrm{RAE} = \frac{\sum_{t=1}^{n}|y_t - \widehat{y_t}|}{\sum_{t=1}^{n}|y_t - \overline{y_t}|}$$

Evaluation Metrics for Numeric Prediction

- **Mean Absolute Error (MAE)**
  - Measures **scale-dependent accuracy** by **averaging the magnitudes of the forecast errors**.

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} |y_t - \hat{y}_t|$$

Assume that you will lose each dollar your model's prediction misses due to an over-estimation or under-estimation.

- MAE tells you how much money you will lose.

```
> mmetric(datTrain2$expenses,prediction_on_train6,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2355.32319 4571.88052   21.96882   25.65733
> mmetric(datTest2$expenses,prediction_on_test6,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2766.02470 5064.75568   24.68685   31.13553
```

Evaluation Metrics for Numeric Prediction

- **Root Mean Squared Error (RMSE)**
  - The RMSE is easy for most people to interpret because of its similarity to the basic statistical concept of a standard deviation
  - **RMSE gives a relatively high weight to large errors**. This means the RMSE should be more useful when large errors are particularly undesirable.

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{n}}$$

Assume that the penalty for an erroneous prediction increases with the difference between the actual and predicted values.

- RMSE is help you assess how much money you will lose

RMSE will be equal or greater than MAE

```
> mmetric(datTrain2$expenses,prediction_on_train6,metric=c("MAE","RMSE","MAPE","RAE"))
        MAE         RMSE         MAPE          RAE
2355.32319 4571.88052    21.96882    25.65733
> mmetric(datTest2$expenses,prediction_on_test6,metric=c("MAE","RMSE","MAPE","RAE"))
        MAE         RMSE         MAPE          RAE
2766.02470 5064.75568    24.68685    31.13553
```

Evaluation Metrics for Numeric Prediction

- **Mean absolute percentage error (MAPE)**
  - The **mean absolute percentage error (MAPE)** **measures predictive accuracy as a percentage**.
  - It cannot be used if there are zero values because there would be a division by zero.

$$\text{MAPE} = \frac{1}{\text{n}} \sum_{t=1}^{n} \left| \frac{y_t - \hat{y}_t}{y_t} \right|$$

MAPE can be used to compare model performances on different target variables

- Compare performances in predicting medical expenses in UT and CA

```
> mmetric(datTrain2$expenses,prediction_on_train6,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2355.32319 4571.88052   21.96882   25.65733
> mmetric(datTest2$expenses,prediction_on_test6,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2766.02470 5064.75568   24.68685   31.13553
```

Evaluation Metrics for Numeric Prediction

- **Relative absolute error (RAE)**
  - RAE is normalized MAE. RAE takes the total absolute error and normalizes it by dividing by the total absolute error of the mean estimator.

RAE indicates model effectiveness compared to a simple mean estimator

$$\text{RAE} = \frac{\sum_{t=1}^{n} |y_t - \hat{y}_t|}{\sum_{t=1}^{n} |y_t - \bar{y}_t|}$$

→ total absolute error of current predictive model

→ total absolute error of the mean estimator

```
> mmetric(datTrain2$expenses,prediction_on_train6,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2355.32319 4571.88052   21.96882   25.65733
> mmetric(datTest2$expenses,prediction_on_test6,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2766.02470 5064.75568   24.68685   31.13553
```

RAE and MAE are always consistent

Evaluation Metrics for Numeric Prediction

- SVM regression with different C values
  - C=1

```
> mmetric(datTrain2$expenses,prediction_on_train6,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2355.32319 4571.88052   21.96882   25.65733
> mmetric(datTest2$expenses,prediction_on_test6,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2766.02470 5064.75568   24.68685   31.13553
```

  - C = 5

```
> mmetric(datTrain2$expenses,prediction_on_train7,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2107.13391 4271.72790   20.41174   22.95372
> mmetric(datTest2$expenses,prediction_on_test7,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2714.76124 4982.99992   24.21993   30.55849
```

  - C = 10

```
> mmetric(datTrain2$expenses,prediction_on_train8,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2002.93488 4099.88096   20.29481   21.81864
> mmetric(datTest2$expenses,prediction_on_test8,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2784.50102 5055.22912   25.74966   31.34351
```

Which model has better performances?

Evaluation Metrics for Numeric Prediction

- SVM regression with C=5

```
> mmetric(datTrain2$expenses,prediction_on_train7,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2107.13391 4271.72790   20.41174   22.95372
> mmetric(datTest2$expenses,prediction_on_test7,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2714.76124 4982.99992   24.21993   30.55849
```

- ANN with 2 hidden layers (16, 8)

```
> mmetric(datTrain2$expenses,prediction_on_train5,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2345.88167 4585.39501   19.71702   25.55448
> mmetric(datTest2$expenses,prediction_on_test5,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2667.92899 5012.17672   23.24695   30.03132
```

Which model has better performances?

Evaluation Metrics for Numeric Prediction

- SVM regression with C=5

```
> mmetric(datTrain2$expenses,prediction_on_train7,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2107.13391 4271.72790   20.41174   22.95372
> mmetric(datTest2$expenses,prediction_on_test7,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2714.76124 4982.99992   24.21993   30.55849
```

- ANN with 2 hidden layers (8, 4)

```
> mmetric(datTrain2$expenses,prediction_on_train6,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2489.76175 4507.40811   25.21162   27.12181
> mmetric(datTest2$expenses,prediction_on_test6,metric=c("MAE","RMSE","MAPE","RAE"))
       MAE       RMSE       MAPE        RAE
2795.43547 4930.50234   27.64945   31.46659
```

Real value: $y_1 = 1, y_2 = 10$

Prediction1: $\widehat{y_1} = 2, \widehat{y_2} = 8$

- $e_1 = 1,\ e_2 = 2$
- MAE = 1.5, RMSE = 1.58, MAPE = 60

Prediction2: $\widehat{y_1} = 3, \widehat{y_2} = 9$

- $e_1 = 2,\ e_2 = 1$
- MAE = 1.5 , RMSE = 1.58
- MAPE = 105