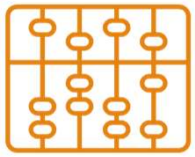
McMaster University | Integrated Biomedical Engineering & Health Sciences Program

IBEHS - 4QZ3
Modelling of Biological Systems

Lecture 12

TAYLOR DEVET MASC.
PHD. CANDIDATE BIOLOGICAL AND BIOMEDICAL ENGINEERING
MCGILL UNIVERSITY
SHRINERS HOSPITAL FOR CHILDREN

Todays Aims…



Linear Classifiers

Unsupervised Learning

Neural Networks

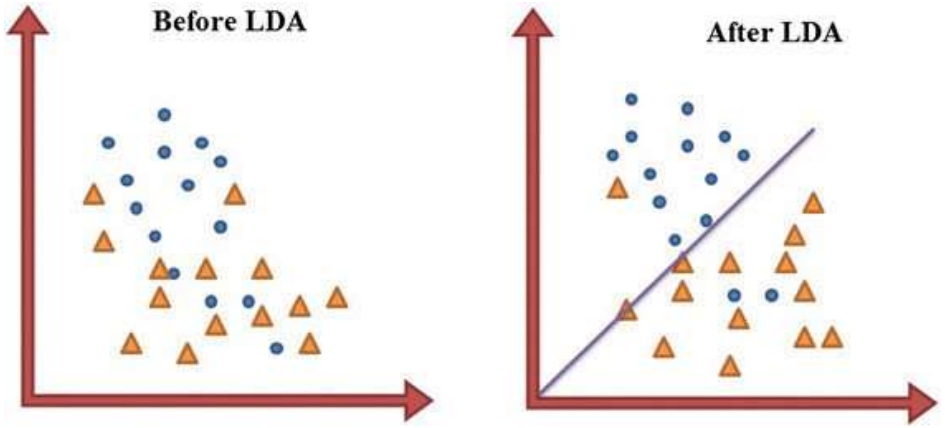
Linear Classifiers

Linear Discriminators

Linear classifiers use decision boundaries that are linear

- straight lines for two variables
- planes for three variables
- hyperplanes for four or more variables

These classifiers produce a **single boundary** so they can separate only **two classes** at a time.



https://www.researchgate.net/publication/288002528_Data_mining_EEG_signals_in_depression_for_their_diagnostic_value/figures?lo=1&utm_source=google&utm_medium=organic

Linear Discriminators

The class predicted by a linear discriminator is given by the output of a linear equation:

$$y = \sum_{i=1}^{M} x_i w_i + b$$

where M is the number of input variables

x_i are the input variables

w_i are the weights → orientation of separation plane/line

b is the bias or offset. → determines location of separation place/line

Linear Classifiers (cont)

The output of the linear classifier to any set of input variables is wholly determined by the weights w_i and bias, b.

$$y = \sum_{i=1}^{N} x_i w_i + b$$

The bias, b, can be included as one of the weights, by adding an input x(M+1) that is always 1.0

(i.e., the inputs become: xi = [x1, x2, x2....xM, 1])

$$y = \sum_{i=1}^{M+1} x_i w_i = \boldsymbol{X}\boldsymbol{w} \quad \text{where} \quad x_{(M+1)} = 1$$

Essentially: $w_{(M+1)} = b$

Linear Classifiers (cont)

If $y > 0.5$, then the classifier is predicting that the input data belong in class 1. Class 1 = 1

If $y \leq 0.5$ the classifier is predicting that the input data belong to class 0. Class 0 = 0.

"Class 0" and "class 1" are arbitrary names for the two classes. They stand for two categories such as "normal" and "diseased" or "malignant" and "benign."

Linear Classifiers (cont)

The weights (and bias) constitute the free parameters of this classification method.

Sometimes linear classifiers are optimal

Linear classifiers can be easily implemented and quickly trained.

Training Set Class Identification

The known class is specified by variable d where:

$$d = \begin{cases} 0 \text{ or } -1 & \text{Class } 0 \\ 1 & \text{Class } 1 \end{cases}$$

In the case where there is more than two classes, d is usually a vector of 0's and 1's

$$d = \begin{cases} 1000 & \text{Class } 0 \\ 0100 & \text{Class } 1 \\ 0010 & \text{Class } 3 \\ 0001 & \text{Class } 4 \end{cases}$$

This unusual structure makes programming easier as will become apparent later.

- These identifiers are an essential component of any training set.

Training Methods
Least Squares Algorithm

The sum of squares error between the output of the linear classifier and the correct class is given by:

$$\varepsilon^2(w) = \sum_{i=1}^{N+1} \left(d_i - x_i^T w\right)^2$$

In matrix notation:

$$\varepsilon^2(w) = \left(\boldsymbol{d} - \boldsymbol{X}\boldsymbol{w}\right)^T \left(\boldsymbol{d} - \boldsymbol{X}\boldsymbol{w}\right)$$

Minimizing *sse(w)* is done by differentiating *sse(w)* with respect to w, setting to zero, and solving for w. If X^TX is nonsingular, a unique solution is:

$$w = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1} \boldsymbol{X}^T\boldsymbol{d}$$

Example 1

Generate a test set consisting of two Gaussian distributions with centers 3.0 standard deviations apart.

The test set should include the correct classification vector d.

Apply the least squares method to classify these two data sets and then plot the results.

Plot the two classes as circles and squares and plot any misclassified data points filled in black.

Finally plot the decision boundary produced by the linear classifier.

Plotting the Results

The equation for the boundary can be determined from the basic equation and weights, wi. The boundary occurs at y =0.5:

$$y = \sum_{i=1}^{N+1} x_i w_i = \boldsymbol{X}\boldsymbol{w} = 0.5$$

For two inputs, this equation becomes:

$$w_1 x_1 + w_2 x_2 + w_3 = 0.5$$

Solving for x2 in terms of x1:

(y = mx + b)

$$x_2 = \left(\frac{-w_1}{w_2} \right) x_1 - \frac{0.5 - w_3}{w_2}$$

This equation is used in linear_eval to plot the boundaries

linear_eval: Routine to Evaluate Performance of a Linear Classifier

```
function [sensitivity, specificity] = linear_eval(X,d,w)
% Function to evaluate performance.
%  X    data set
%  d     Correct classification
%  w    Classifier weights (totally defines classifier)
threshold = .5;              % The decision threshold
% Initialize counters, etc.
[r,c] = size(X);             % Determine data set size
y = X*w;                                % Evaluate the output
% Plot the results
```

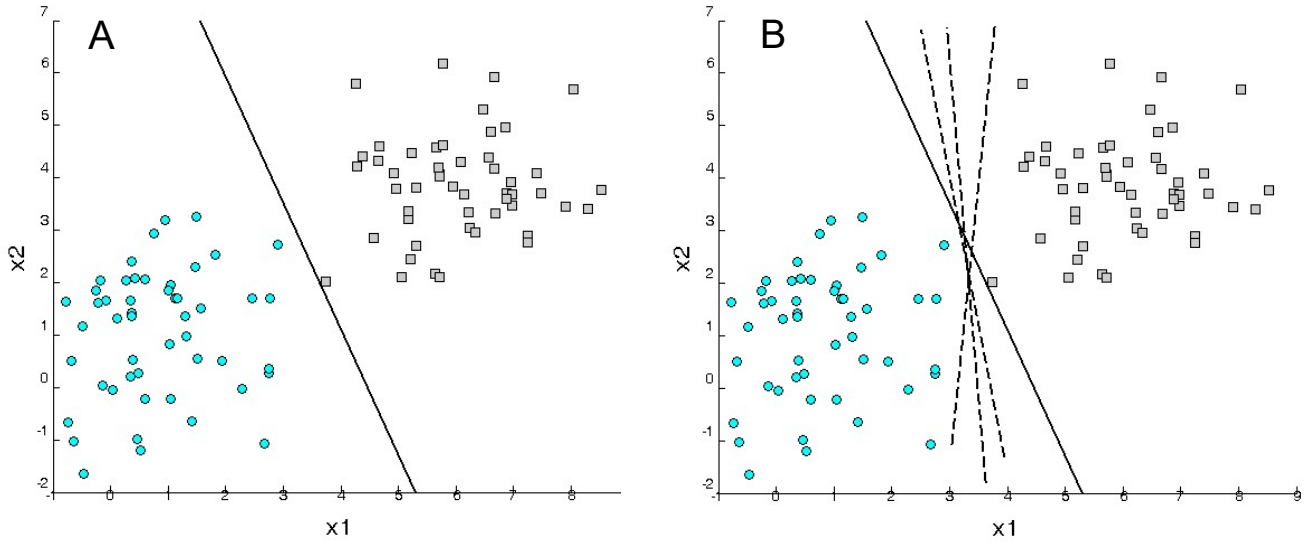
Plotting and Counting the Four Possibilities

```
% Assumes Class 0 is 0 and Class 1 is 1.
Threshold = 0.5

% Evaluates each point for all four
possibilities


for i = 1:r

    if d(i) > threshold & y(i) > threshold

        plot(X(i,1),X(i,2),'sqk')

        tn = tn + 1;

%1) True negative

    elseif d(i) > threshold & y(i) <=
threshold

        plot(X(i,1),X(i,2),'sqk')

        fp = fp + 1;

% 2) False positive

    elseif d(i) <= threshold y(i) <= threshold

        plot(X(i,1),X(i,2),'ok');

        tp = tp + 1;
 % 3) True positive

    elseif d(i) <= threshold & y(i) >
threshold

        plot(X(i,1),X(i,2),'ok');

        fn = fn + 1;
 % 4) False negative

    end

end
```

Plotting the Linear Decision Boundary

```
clf; hold on;
y_lim = get(gca,'YLim');                    % Used to reset axis
x_lim = get(gca,'XLim');
% Plot decision boundary
x1 = [min(X(:,1)),max(X(:,1))];                 % Construct x1 to span x1
x2 = -w(1)*x1/w(2) + (-w(3)+.5)/w(2);       % Calculate x2 using eq.
plot(x1,x2,'k');              % Plot boundary line
axis([x_lim, y_lim]);                       % Limit axis
%
% Evaluate sensitivity and specificity
specificity = (tn/(tn+fp))*100; % Specificity
sensitivity = (tp/(tp+fn))*100; % Sensitivity
```

Example 1 Results



The boundary found is shown on the left accurately separates the two classes in this training set.

But many boundaries can do that as shown on the right (an infinite number).

What is the best boundary? The one that will generalize best.

Sensitivity and Specificity

Sensitivity: Percent correct detections

$$Sensitivity = 100\frac{True\ Positives}{True\ Positives + False\ Negatives} = 100\frac{True\ Positives}{Total\ Abnormal}$$

Specificity: Percent correct rejections

$$Specificity = 100\frac{True\ Negatives}{True\ Negatives + FalsePositives} = 100\frac{True\ Negatives}{Total\ Normal}$$

The Receiver Operation Curve (ROC)

Sometimes it is possible to vary the decision boundary to increase or decrease the detection of abnormal values.

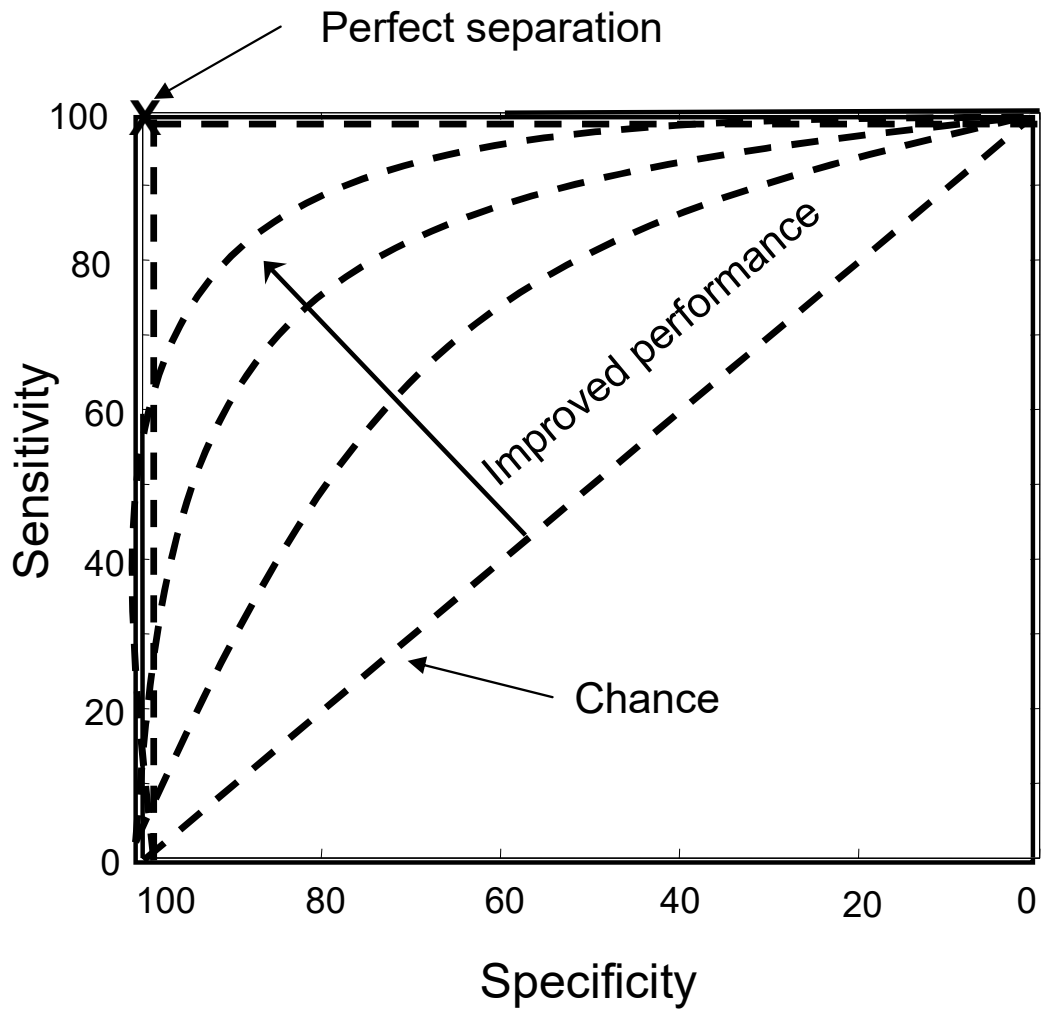
Increasing the detection of true positives will usually increase the number of false positives and lower the number of true negatives.

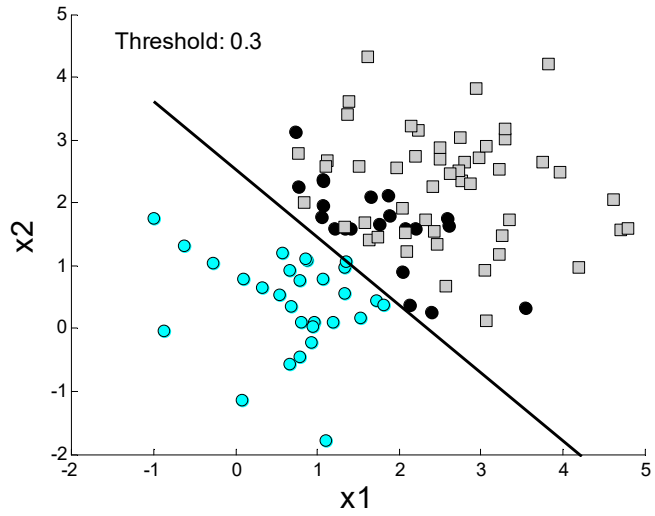
There is a trade-off between sensitivity and specificity.

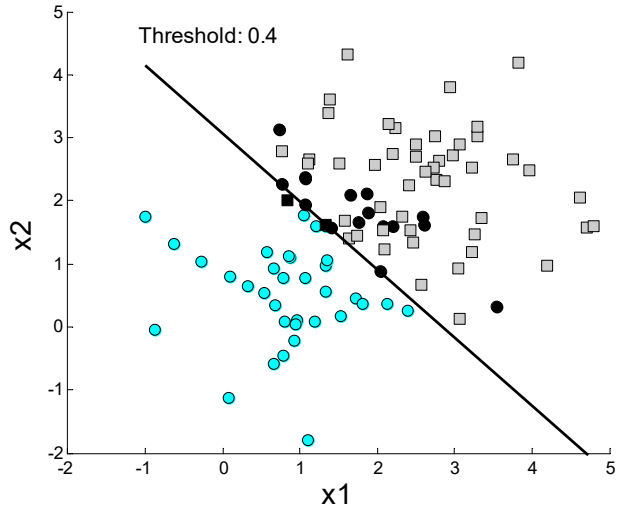
A curve showing this tradeoff between sensitivity and specificity is called the ROC (receiver operator characteristics) curve.

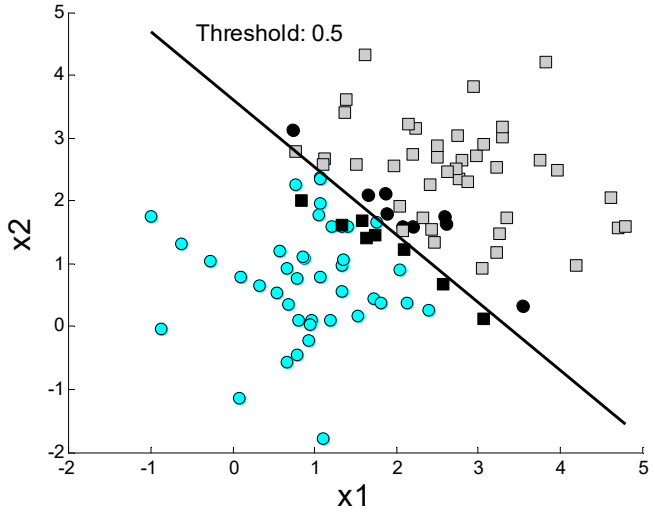
Example ROC Curve

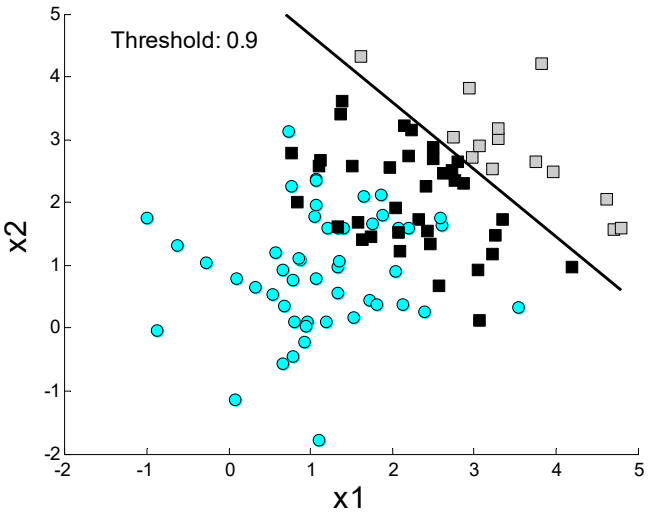
Note that Specificity is plotted in reverse





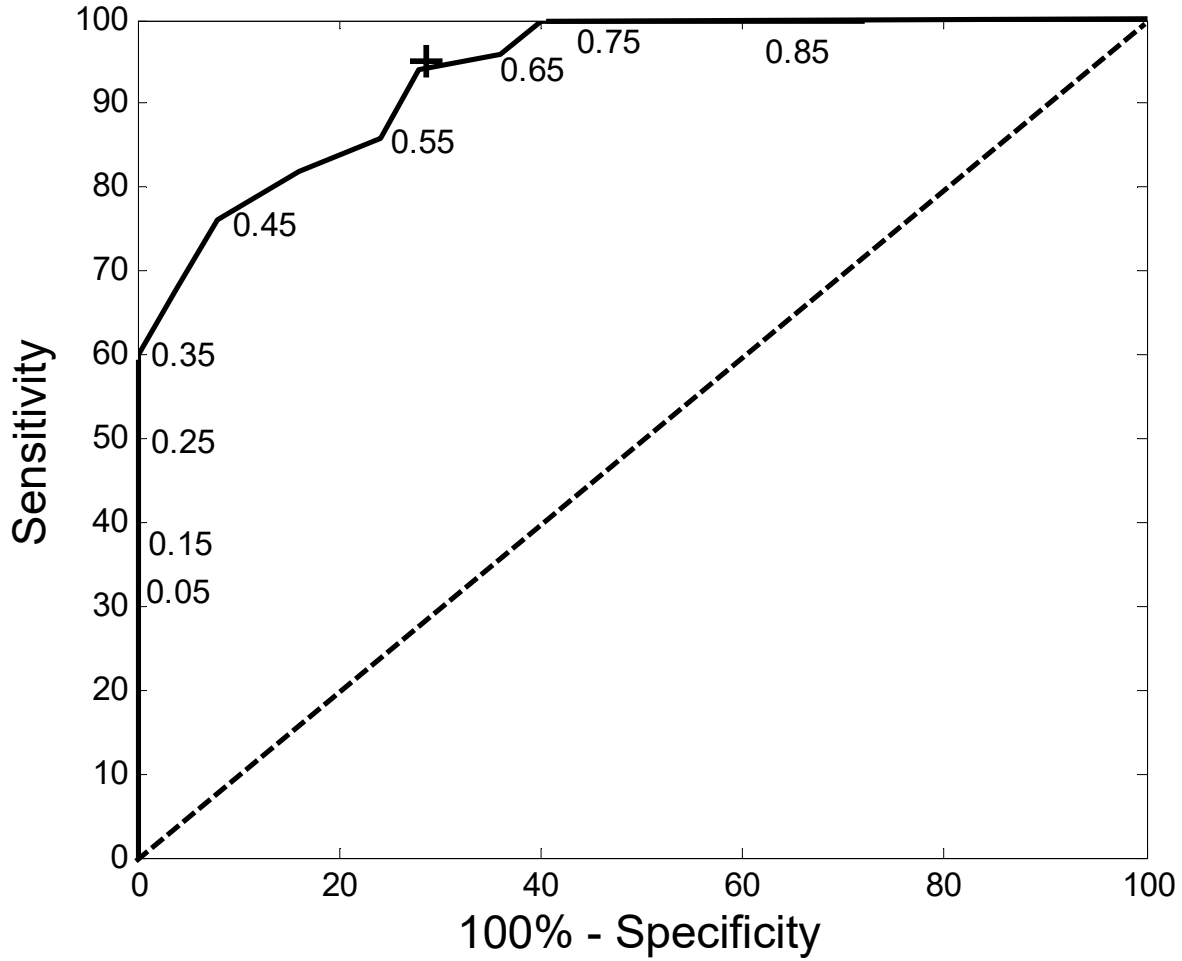






Circles are Diseased

Moving Threshold adjusts sensitivity specificity tradeoff



The data consisted of two closely spaced Gaussian distributions,

The classifier was linear least squares.

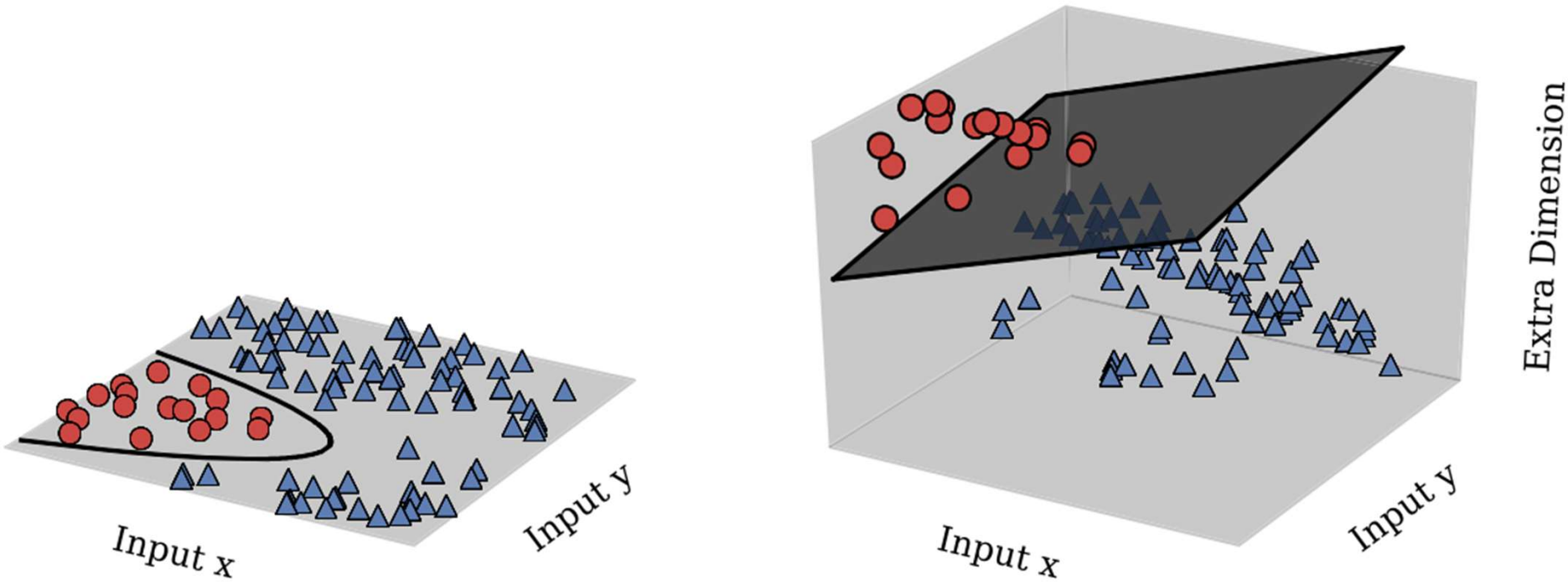
Higher Dimensions – Kernel machines

Many classification problems involve data that are separable, but not by a single straight line

In more complex data sets, it is still possible to separate the classes using a linear boundary if the data are transformed into a higher-dimensional space.

Cover's Theorem (1965)

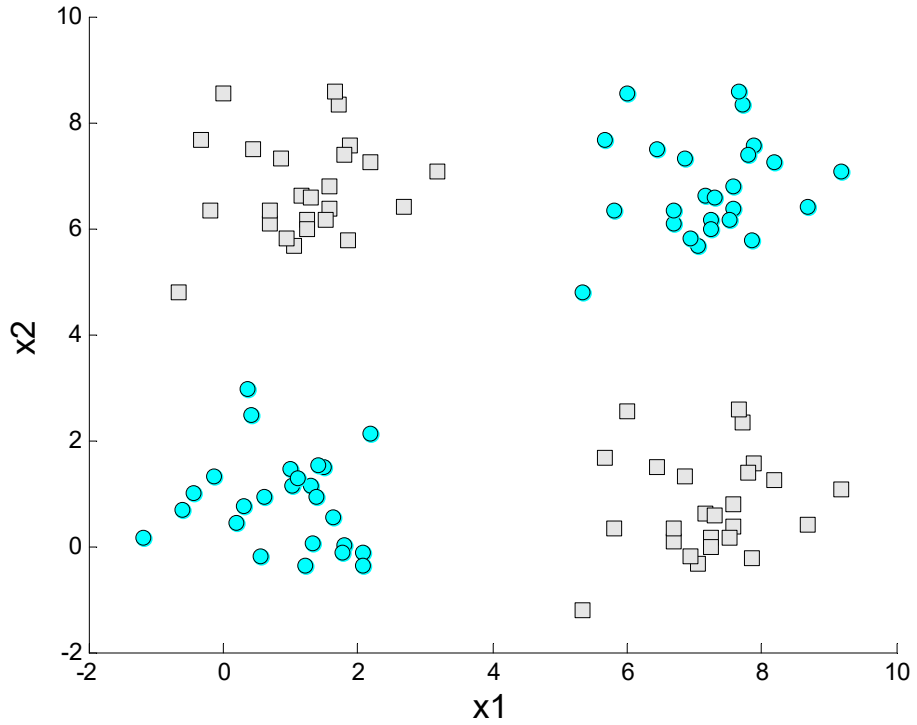
- if the number of dimensions is high enough, you can always find a linear boundary that will separate the data without error
- hyperplane



https://i.stack.imgur.com/fylvr.png

Example

A two-class data set that is clearly separable, but not by a single linear boundary.



Example

Classify the data in the last slide using the least squares linear classifier after the data are transformed into a higher dimensional space.

Solution: The quadratic kernel can be used to separate these data.

Take it from 2 features (x1, x2) to 5:

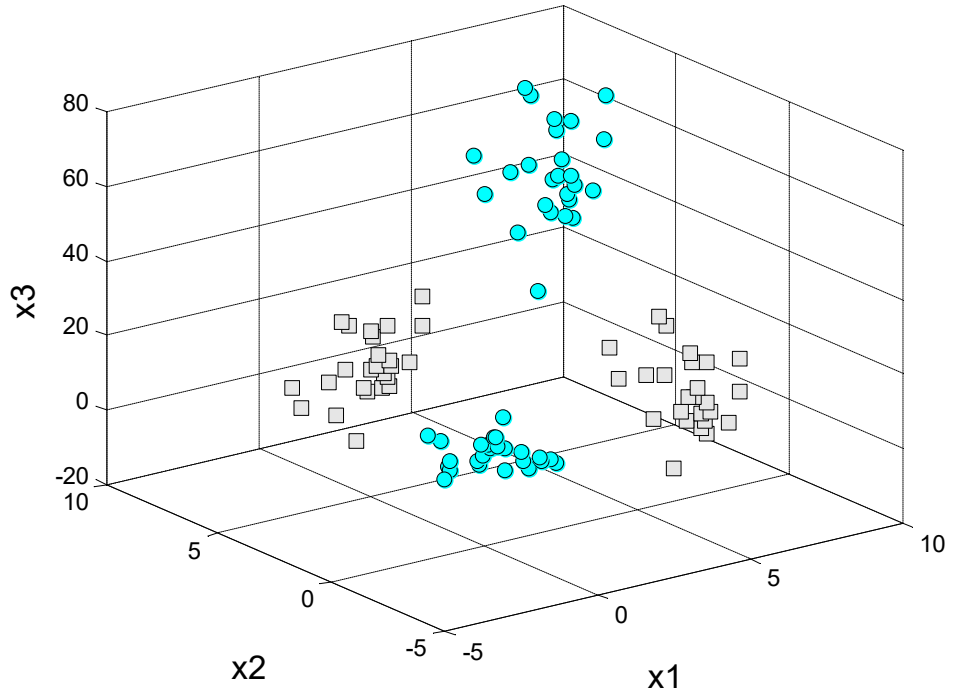
$$x = x1 + x2 + x1 * x2 + x1^2 + x2^2$$

Since the data are so widely spaced it is only necessary to use the cross product term, x1*x2, to obtain perfect separation.

This allows us to implement the classifier in three dimensions (as opposed to five) which is easier to demonstrate.

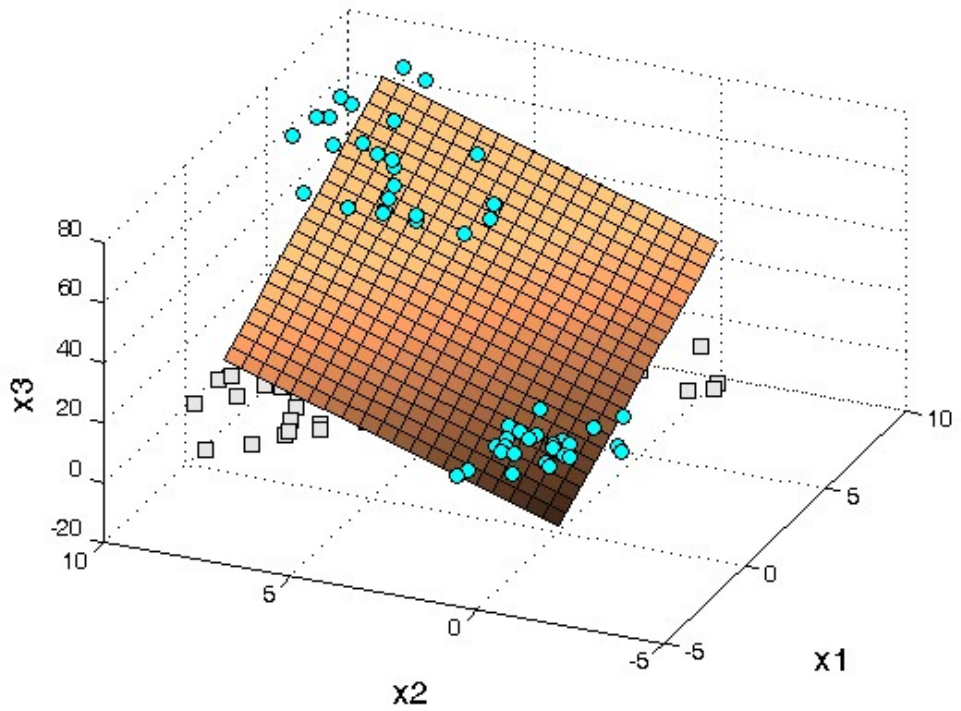
Example

The data in the previous slide transformed into three-dimensional feature space by adding a cross-product term.



Example Results

A plane gives perfect separation of the transformed data set.



Support Vector Machines

Support Vector Machines

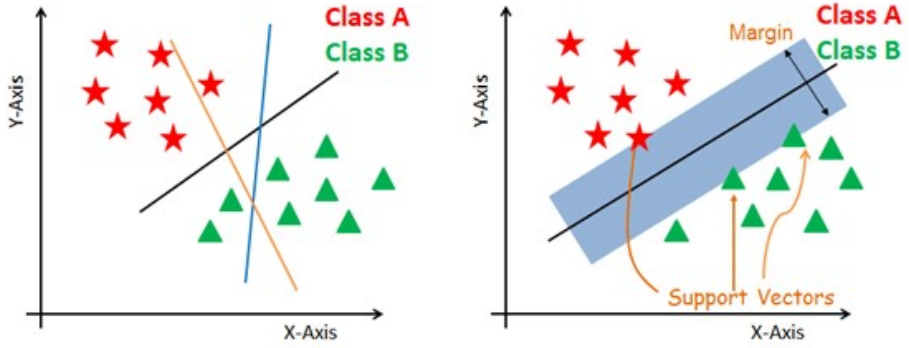
- One of the most robust prediction methods
- Uses binary linear classification
- SVM maximize the distance between the critical data points (support vectors)
- Looks at points in space and tries to maximize the gap between groups
- New data is sorted based on which side of the gap they are on
- Can use kernels to map in higher dimensions'
  - Kernel – similarity function

Support Vector

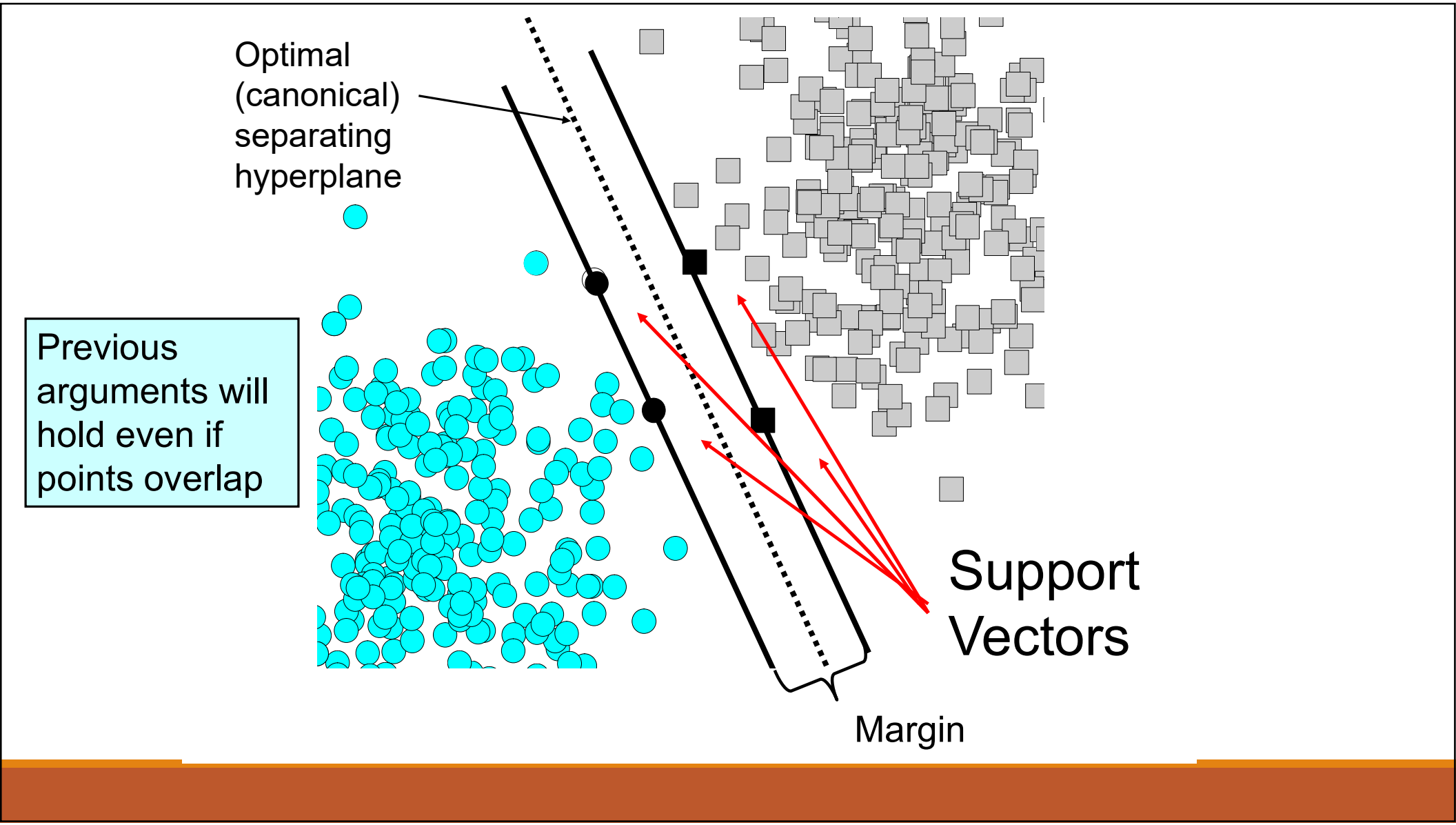
The data points that lie closest to the decision surface/line/plane

They are the points that are the most difficult to classify

They have a direct impact on the location of the decision boundary



https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python


Optimal
(canonical)
separating
hyperplane
Previous
arguments will
hold even if
points overlap
Support
Vectors
Margin

Determining the margin

To determine the equation for the margin, M, note that the distance of any hyperplane, xiw + b = 0, to the origin is:

where ||w|| is the norm of w and is equal to:

or in matrix notation:

If the hyperplane is equal to ±1, then the distance to the origin is just:

Boundary Equations

For the line separating class 1: $y_i \geq 1$, the distance to the origin is:

$$d_o = \frac{(1-b)}{\|\boldsymbol{w}\|}$$

For the line separating class -1: $y_i \leq -1$, the distance to the origin is:

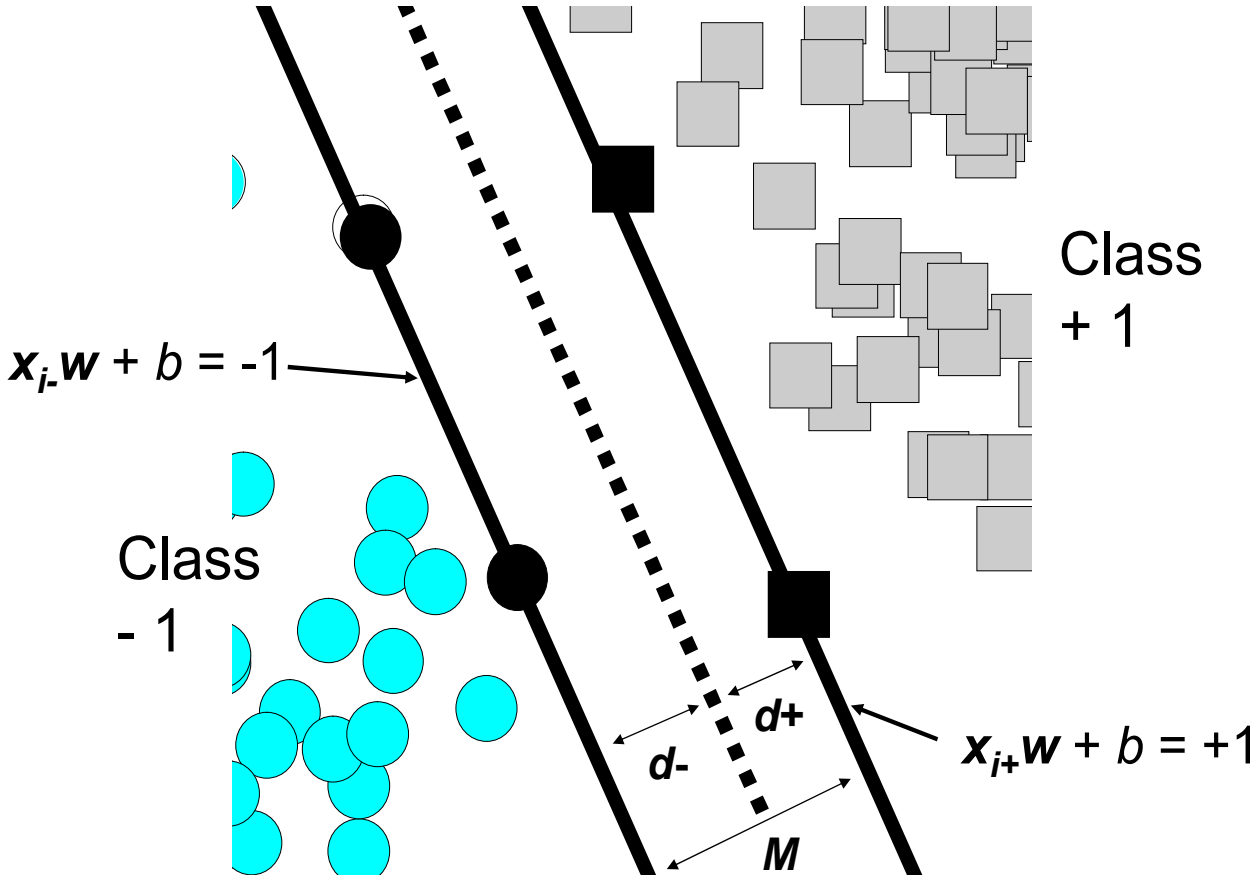
$$d_o = \frac{(-1-b)}{\|\boldsymbol{w}\|}$$

The difference between the two lines, which is the margin, is obtained by subtracting:

$$M = \frac{(1-b)}{\|\boldsymbol{w}\|} - \frac{-(1-b)}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|}$$

The two lines passing though the support vectors mark the boundary for $y \geq \pm 1$ and this determines the equation for these lines.

$$M = \frac{2}{\|\boldsymbol{w}\|}$$



SVM Margin Maximization

- The maximum margin is obtained by minimizing $||w||$ (or by minimizing $||w||^2$) which is equal to $w^T w$ and is somewhat easier to perform.
- The minimization must be done subject to the constraint imposed by $y_i(x_i w+b) \geq 1$ which ensures that the boundaries are on the correct side.
- This type of minimization problem is known as a quadratic programming or *QP* optimization problem and there are a number or routines available to solve this.

Support Classifier Algorithm Development

Find the boundaries that maximize the margins

Find the hyperplane that maximizes the margin

Constrain data points to the appropriate side of the boundary.

Classes are assumed to be ± 1 as this simplifies the mathematics.

Support Classifier Algorithm Development (cont)

Since the decision boundary is at y = 0 (between ± 1, the equation for the boundary is:

$$y = \sum_{i=1}^{N} w_i x_i + b = \boldsymbol{x_i w} + b = 0$$

where xi are the input values

w is the weight vector

b is the offset or bias

(recall w and b define the classifier)

Support Classifier Algorithm Development (cont)

Since the two classes are defined by y = ± 1, the value of y must be ± 1 at the closest points (i.e., the support vectors)

So the equations for the lines that go through those support vector points must be:

$$\boldsymbol{x_i w} + b \geq 1 \quad \text{when} \quad y = +1$$

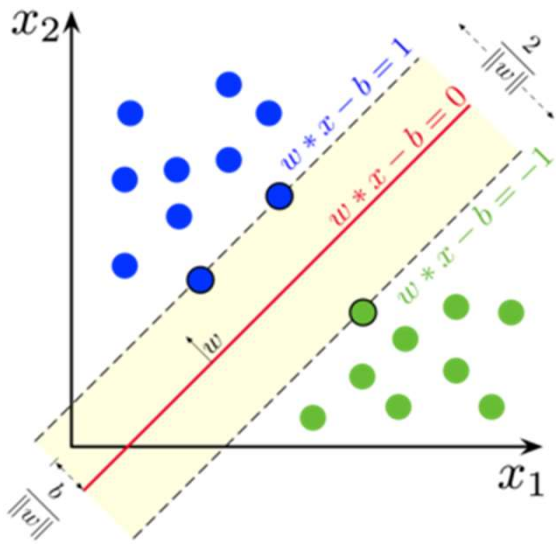
and

$$\boldsymbol{x_i w} + b \leq 1 \quad \text{when} \quad y = -1$$

These can be combined into a single equation:

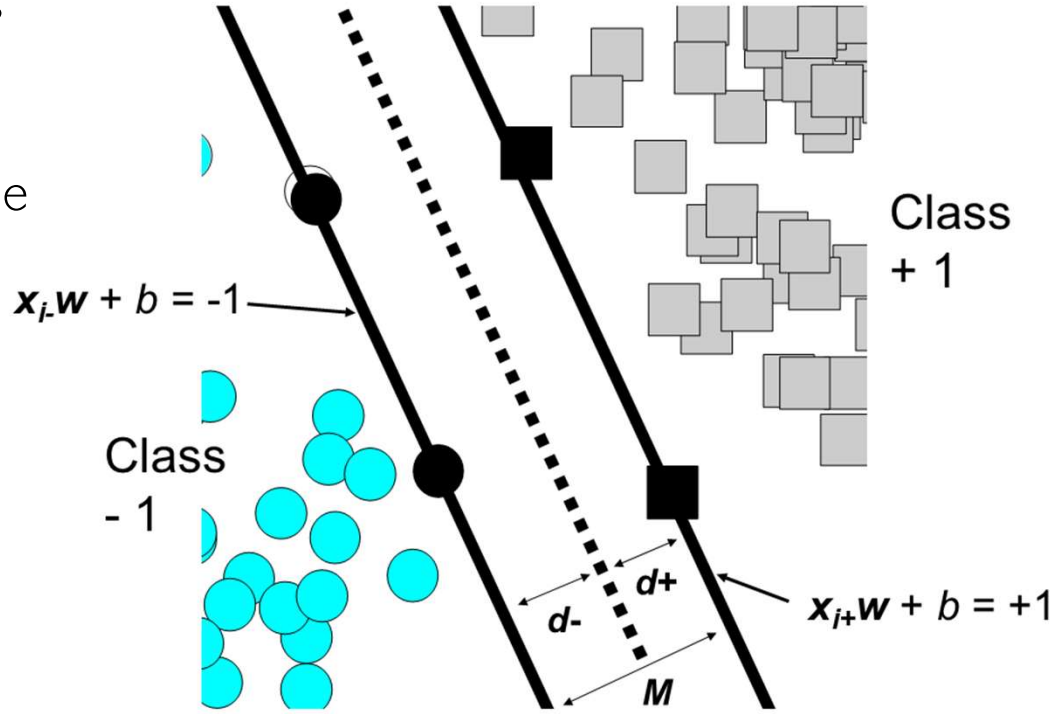
$$y_i\left(\boldsymbol{x_i w} + b\right) \geq 1$$

This equation is interpreted in the next slide.



Support Classifier Algorithm Development (cont)

The equation simply states that *w* and *b* should be such that the two classes fall on opposite sides of the support vector lines.



Support Vector Machines

If the data are not linearly separable:

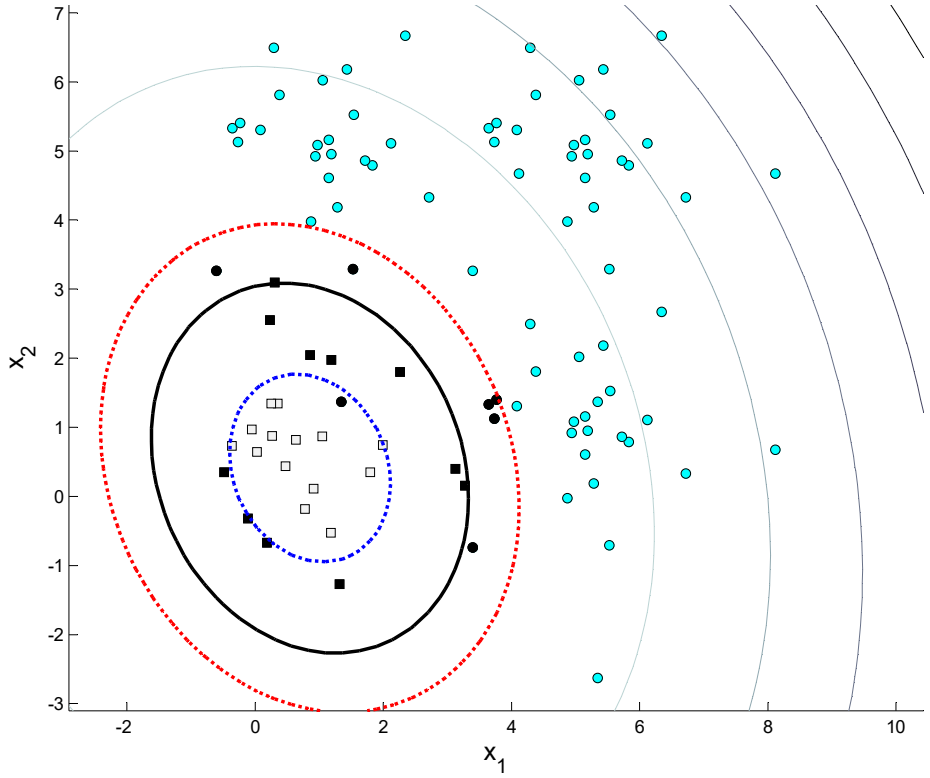
- the points overlap
- the optimization process still maximizes M,
- relaxed constraint so that some of the points can be on the wrong side of the boundary.

This approach can be used to produce linear boundaries and is termed linear support vector machines (LSVM).

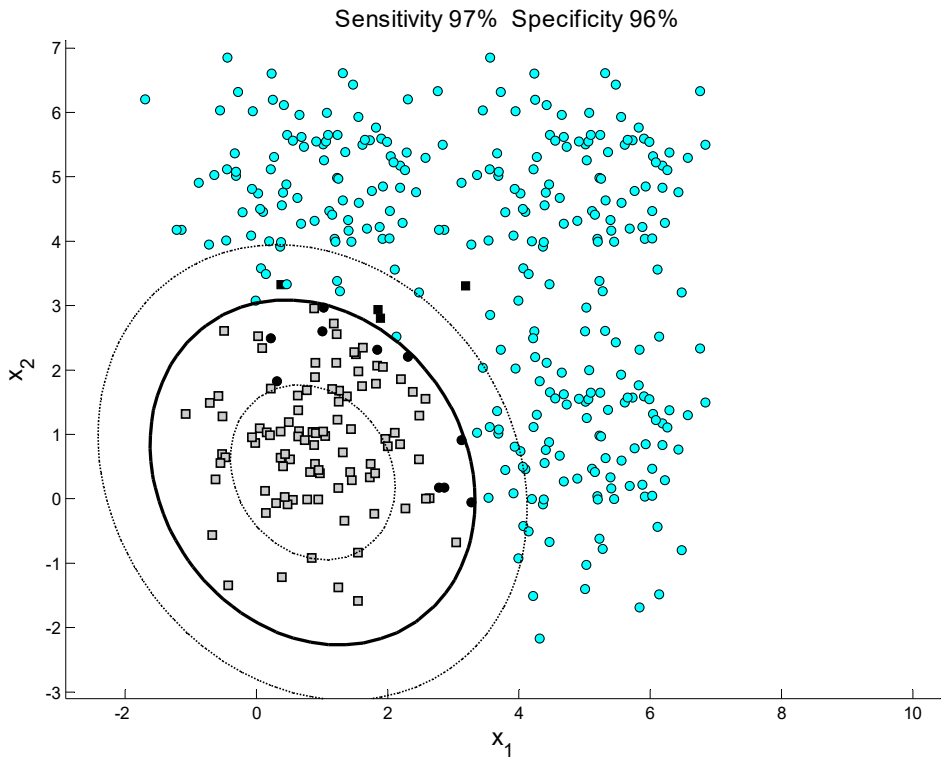
To generate nonlinear boundaries:

- a kernel can be used to transform the data into higher dimensions
- these classifiers are termed support vector machines (SVM).

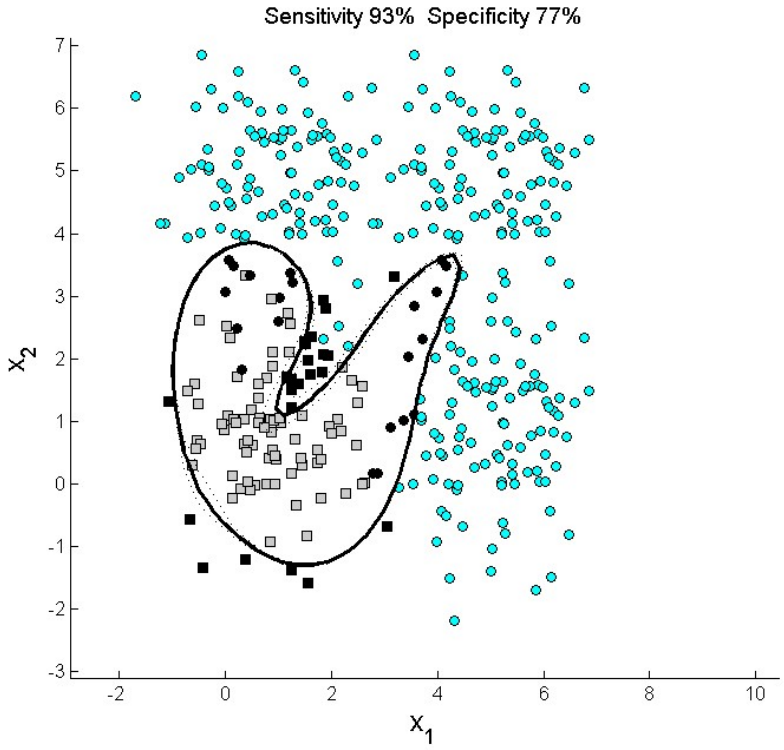
Example Kernels



Example Kernels



Example Kernels: Overtraining



Too Much Machine Capacity

The capacity of the classifier used above was increased by increasing the polynomial order to 6.

The boundary is complicated due to overtraining on the training set.

Test set performance is not nearly as good.

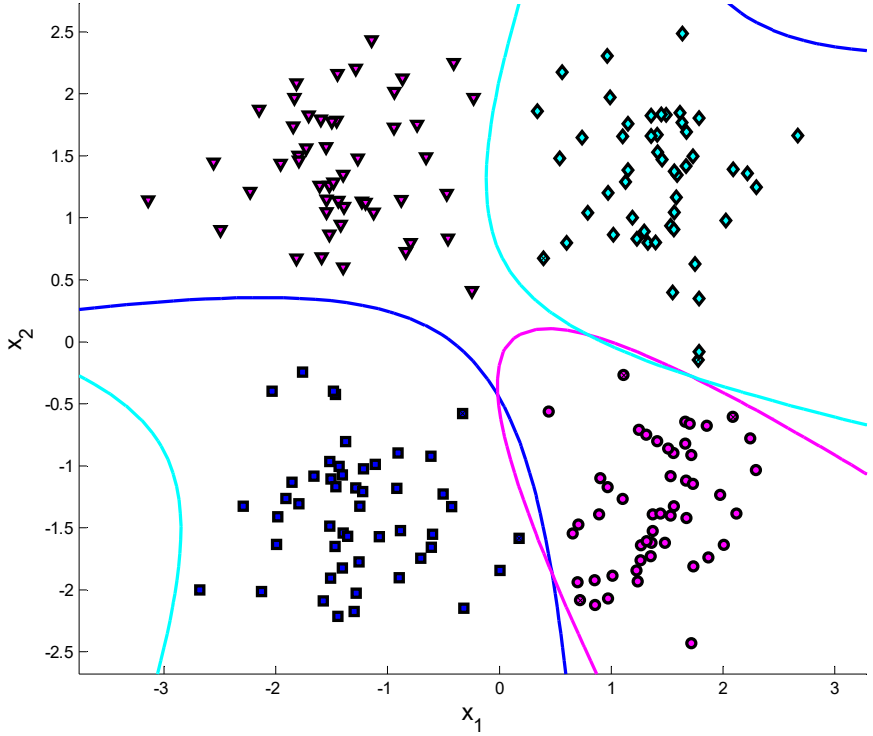
Multiple Classes

To classify data with more than two classes, implement the classifier on each class separately and combine results.

With multiple classes

- the correct classification is indicated by a matrix.
- Each class is indicated by a column which contains +1 if the associated pattern indicates that class.

Example Results: All classes are correctly identified.



K-nearest neighbour

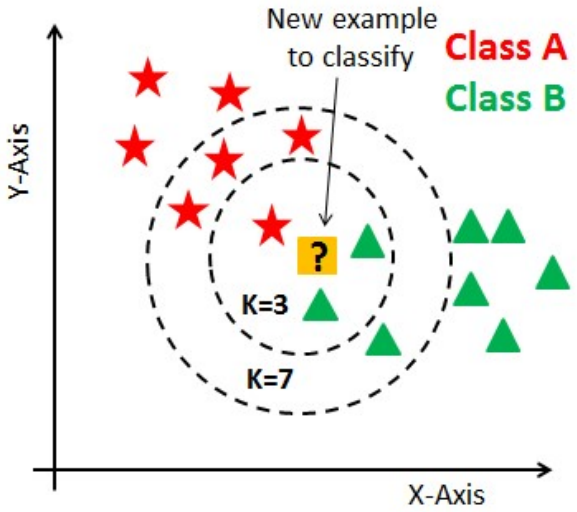
k-nearest neighbor classifier

Operates directly off the testing set and does not need prior training.

Takes each test set point and determines the distance to the k nearest training points, where k is a constant.

It then takes the class values of these nearest points and assigns the test point to the majority class.

This approach can be used for any number of classes and any number of input variables.



https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

Distance Measurement

Distances between points can be measured using a variety of metrics but the most common and straightforward is the Euclidean distance. The Euclidean distance between two points is:

$$d(a,b) = ||a-b||$$
$$d(a,b) = \sqrt{(a-b)\ 2}$$
$$d(a,b) = \sqrt{(ax-bx)2+(ay-by)\ 2)}$$

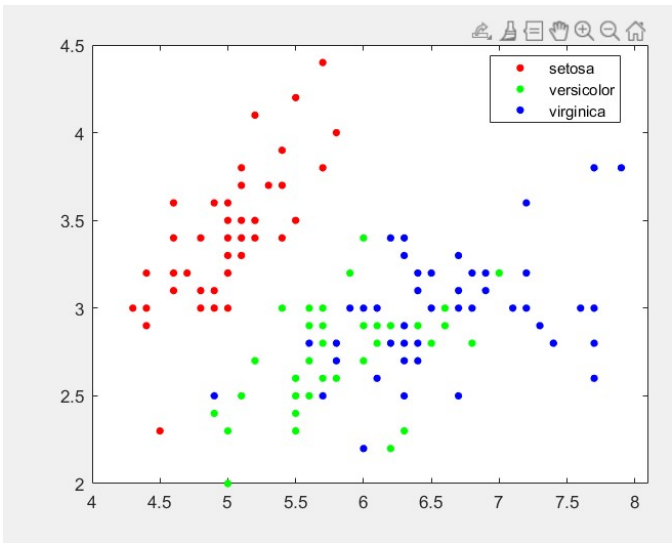
where a and b are vectors and ||a-b|| is the norm of the vector that results after subtraction.

Example

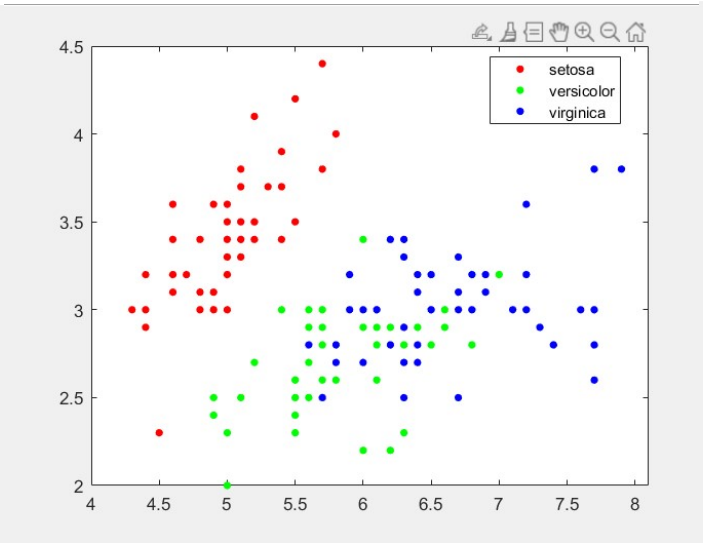
```
load fisheriris;

X = meas;
Y = species;

Mdl = fitcknn(X,Y,'NumNeighbors',5,'Standardize',1)
Mdl.ClassNames

figure();

gscatter(X(:,1),X(:,2),species)

figure();

label = predict(Mdl,X);

gscatter(X(:,1),X(:,2),label)

figure
hold on;

correct = 0;incorrect = 0;
CorrSet = 0; CorrVer = 0; CorrVirg = 0;
InCorrSet = 0; InCorrVer = 0; InCorrVirg = 0;
```

Example Results – k=5

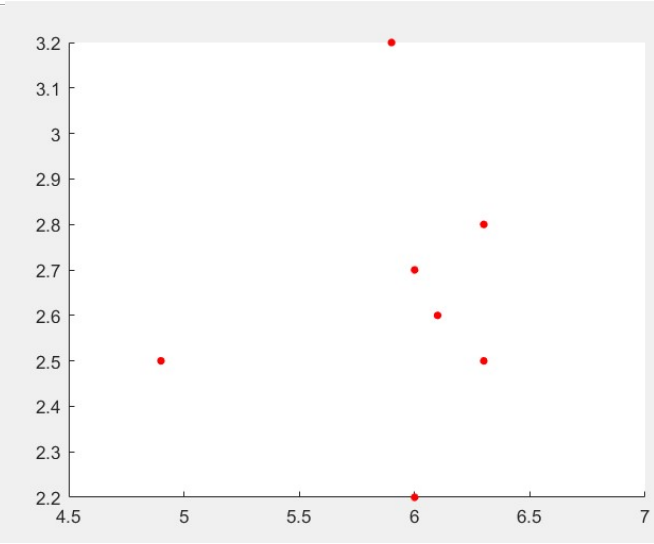
Setosa	Versicolor	Virginica
50	47	46
0	4	3



Original



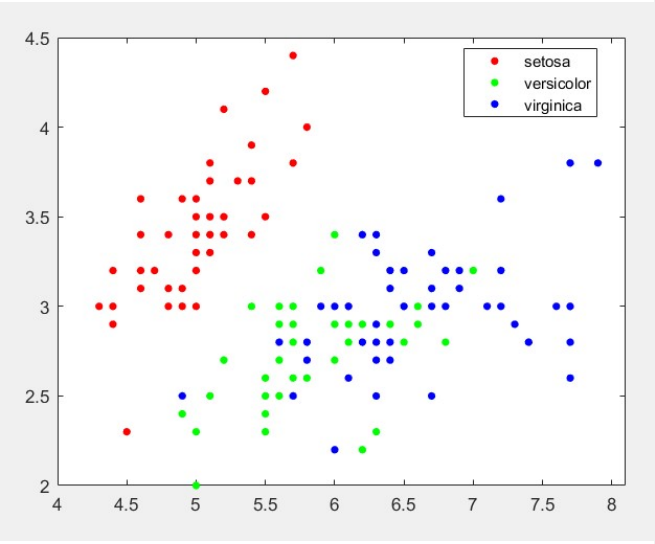
Classification



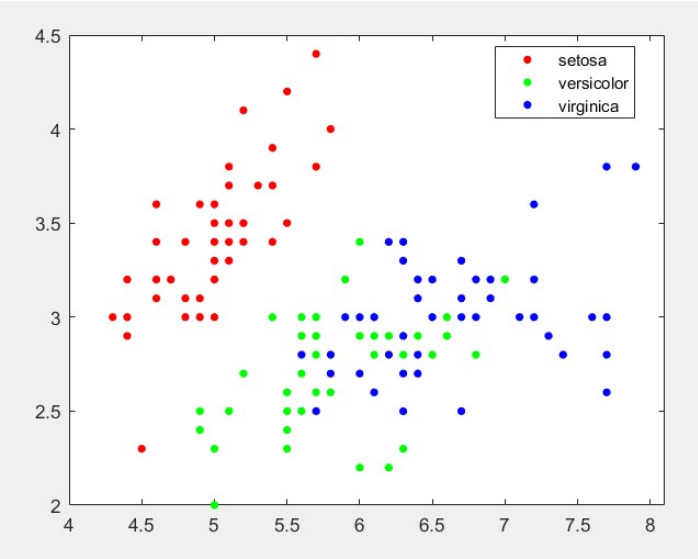
Error

Example Results – k=20

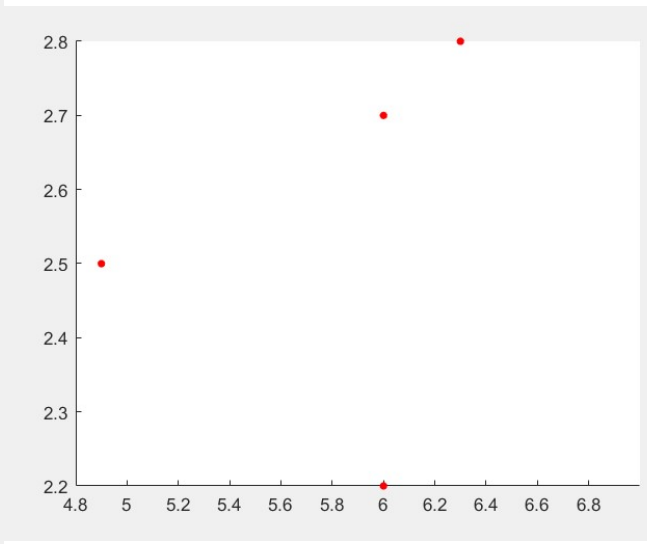
Setosa	Versicolor	Virginica
50	49	47
0	3	1



Original



Classification



Error

Confusion Matrix from Example

	Setosa	Versicolor	Virginica
Correct	50	47	46
Incorrect	0	4	3

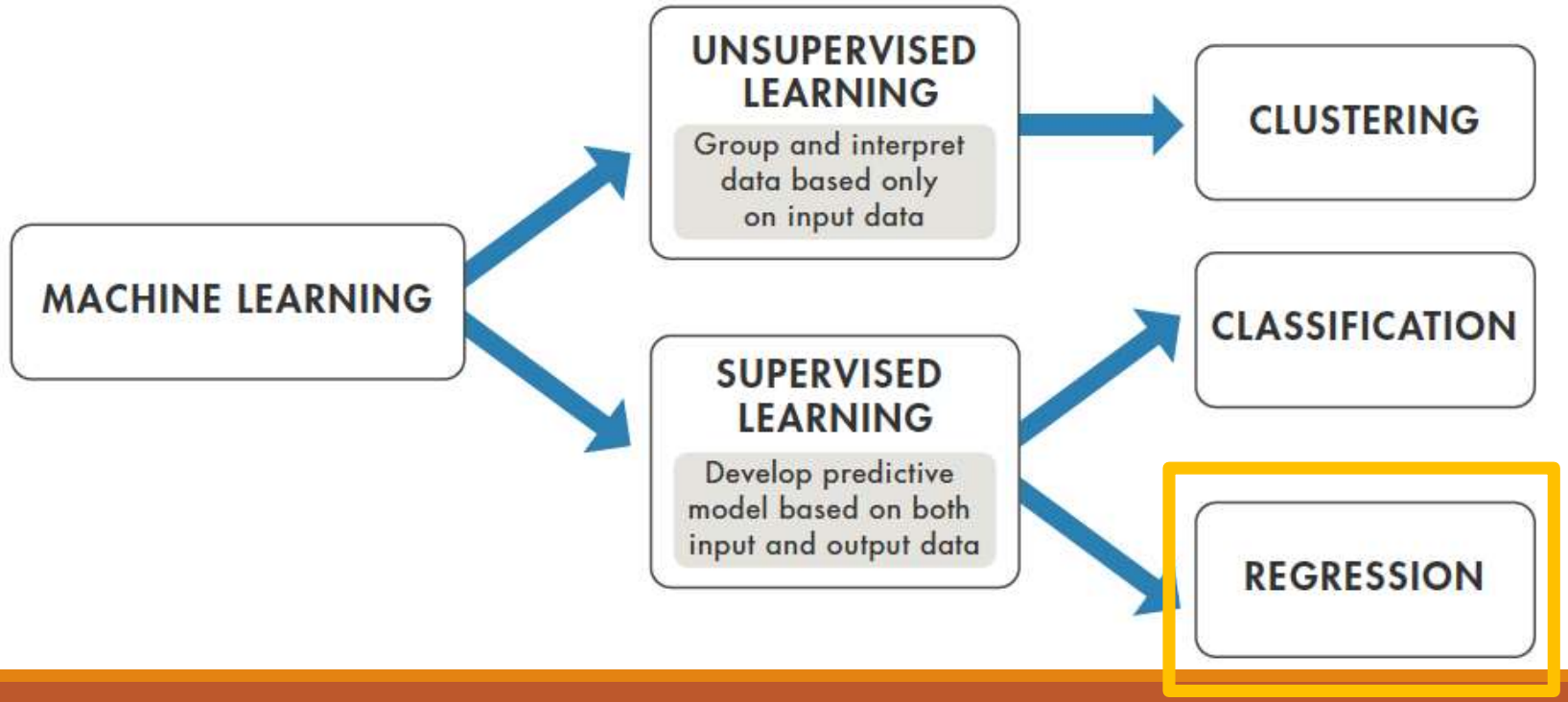
	Setosa	Versicolor	Virginica
Correct	50	49	47
Incorrect	0	3	1

The performance is slightly better when $k = 20$.

A larger k improves generality but, if k is too large, boundary points can be misclassified if they are close to a large group of the other class.

Smaller values of k can lead to misclassification due to a small number of outliers in the other class.

Types of Learning



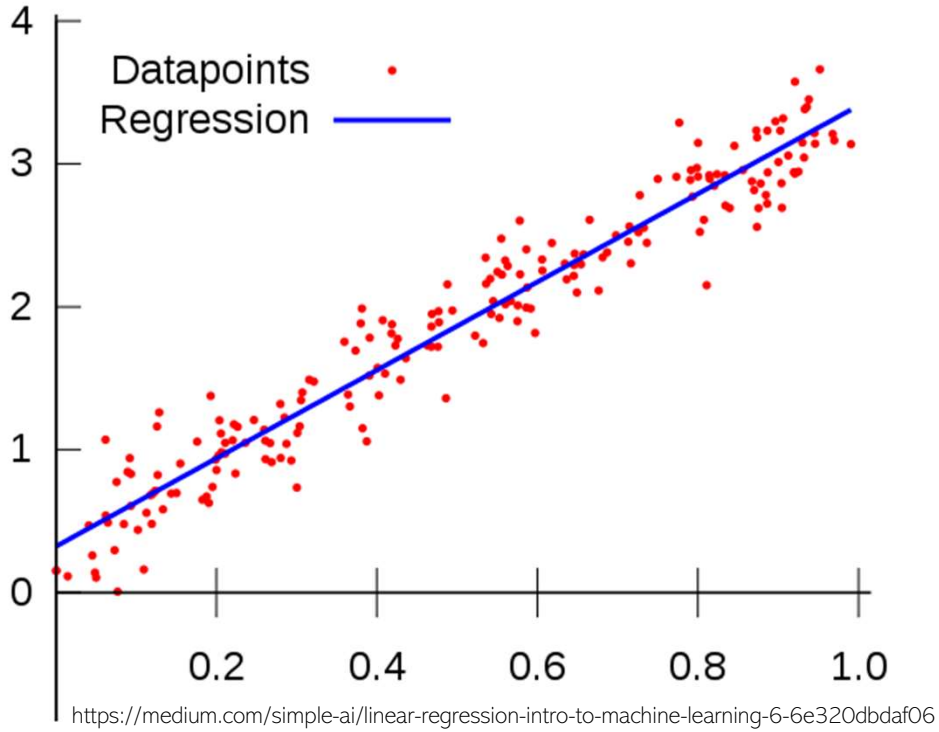
Regression

An algorithm that attempts to predict outcomes based on predictors

Can be linear, logistic, polynomial, ridge, lasso etc.

Important to consider

1. Multicollinearity
   - Predictors are correlated
2. Outliers
   - May have to pre filter for outliers
3. Heteroscedasticity
   - Unequal variability



https://medium.com/simple-ai/linear-regression-intro-to-machine-learning-6-6e320dbdaf06

Decisions Trees

Each node has different classification feature

Outputs of node are the possible outcomes of that classification feature

Outcome can be:

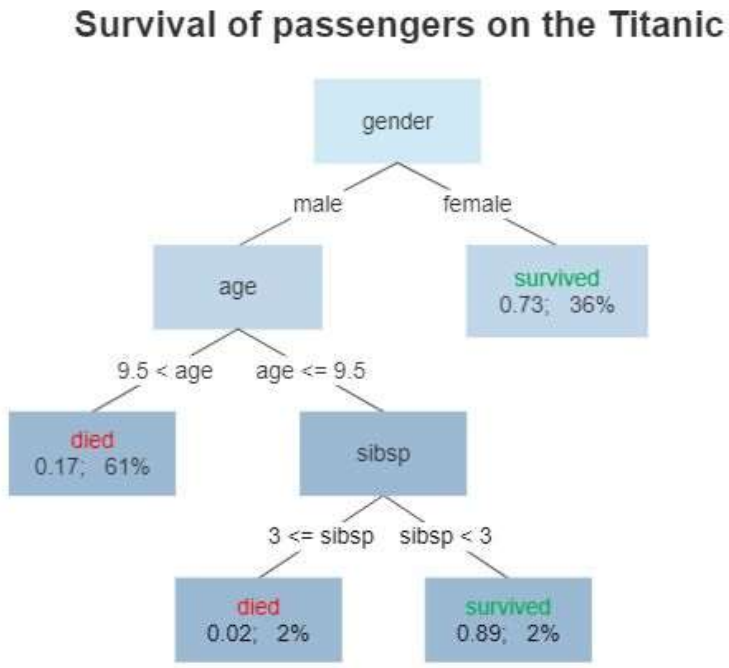
- Decision based on different classification feature
- Leaf with final classification and probability distribution

Classification Trees

- Predict discrete class i.e. yes, no
- Unordered values using dependent variables
- Titanic passenger survival

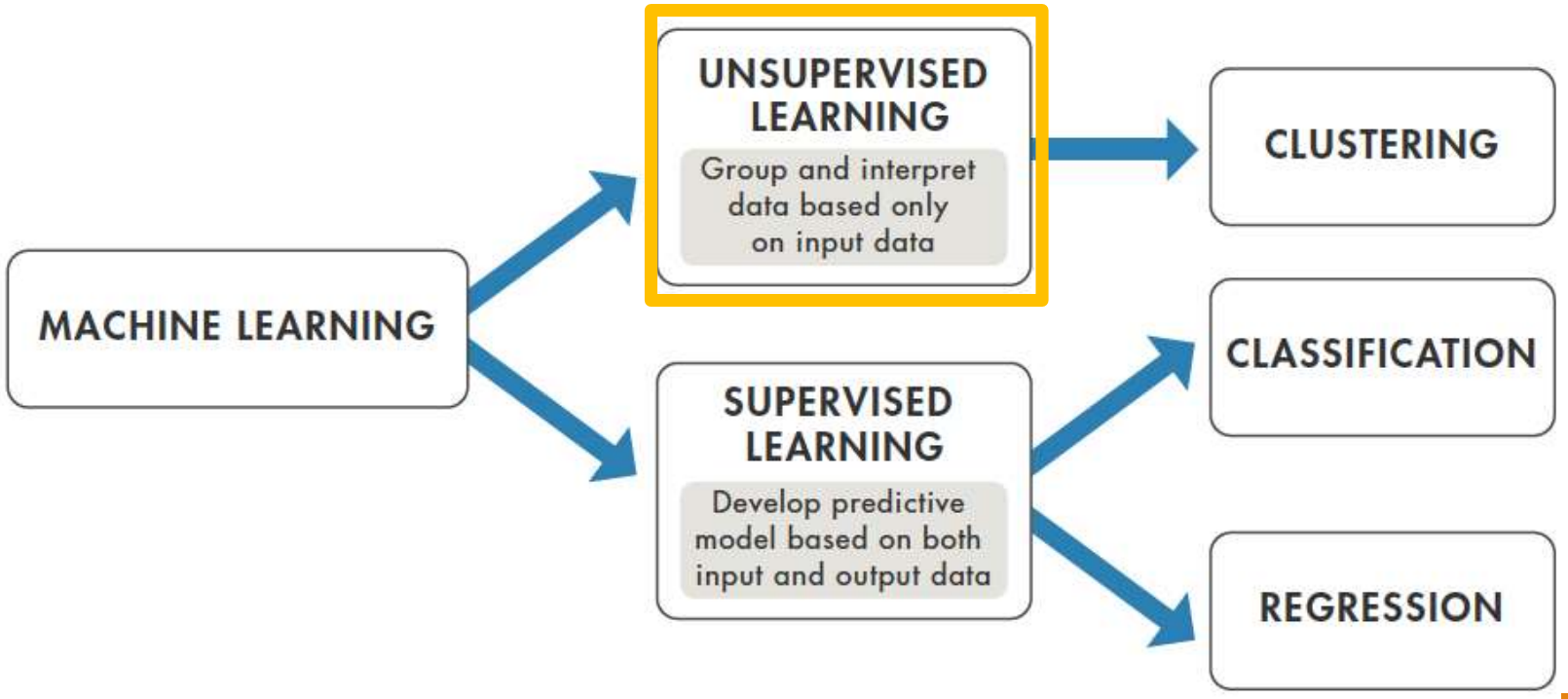
Regression Trees

- Predict value on a continuous scale
- Ordered values
- E.x. Guess house price based on size, area etc.

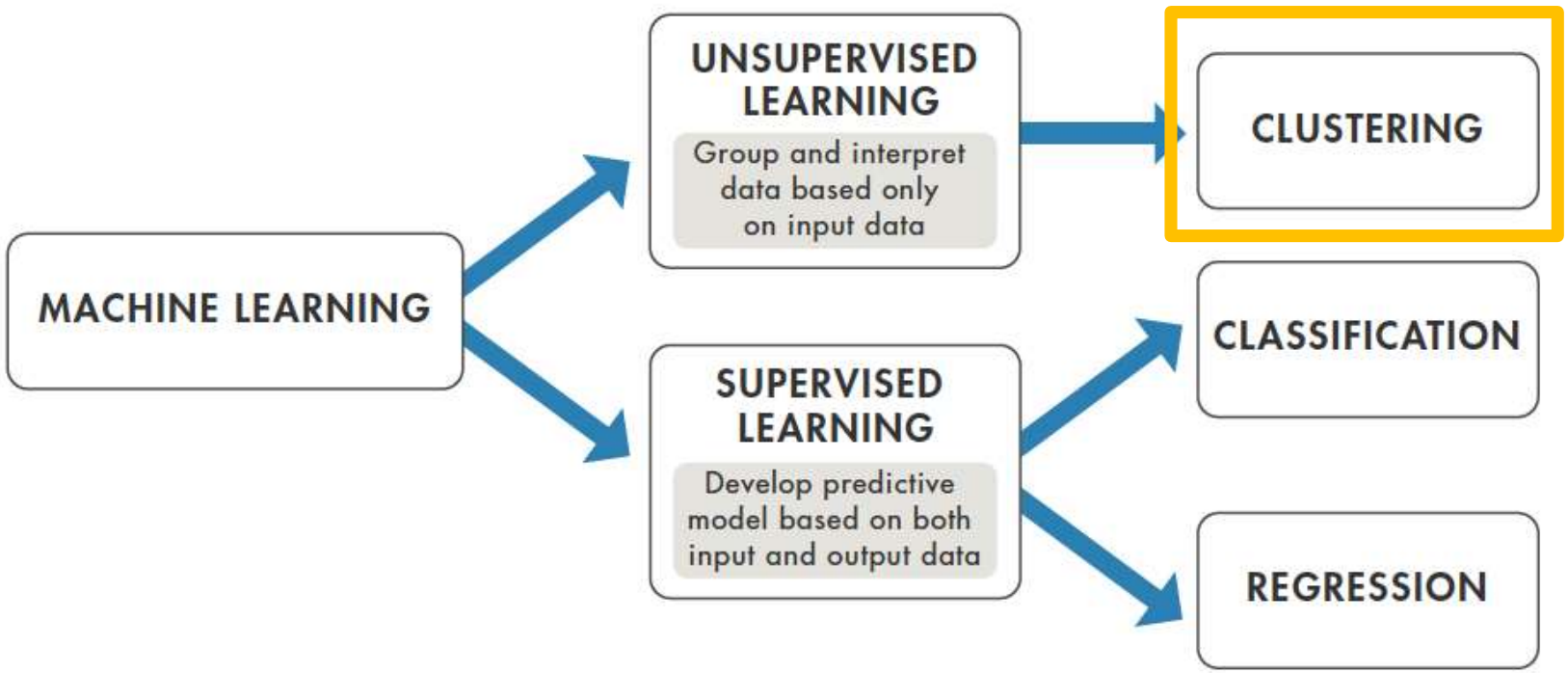


https://commons.wikimedia.org/wiki/File:Decision_Tree.jpg

Types of Learning



Types of Learning

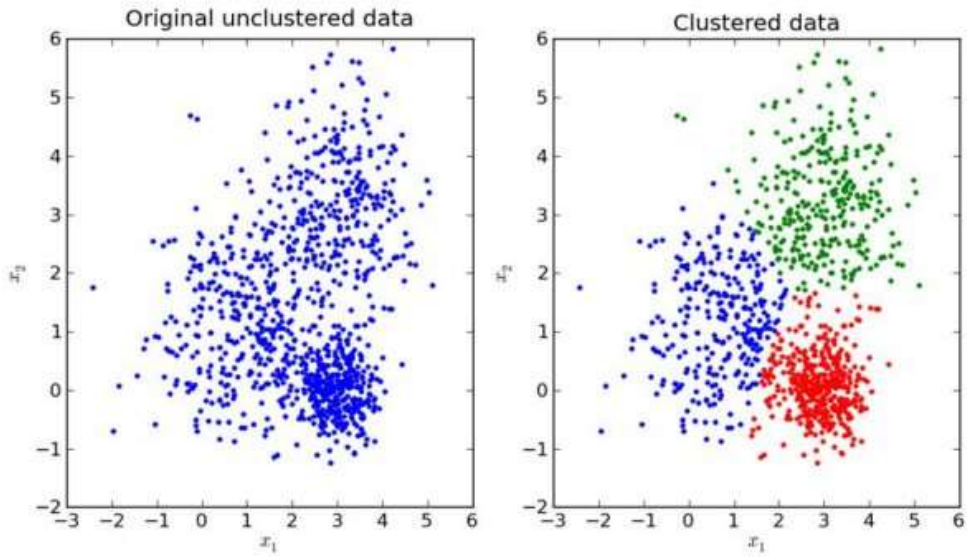


Clustering

Used to group data that is untagged/unlabeled

Try to make groups such that members of the group are more similar to each other than the other groups

Can use a variety of algorithms to achieve this



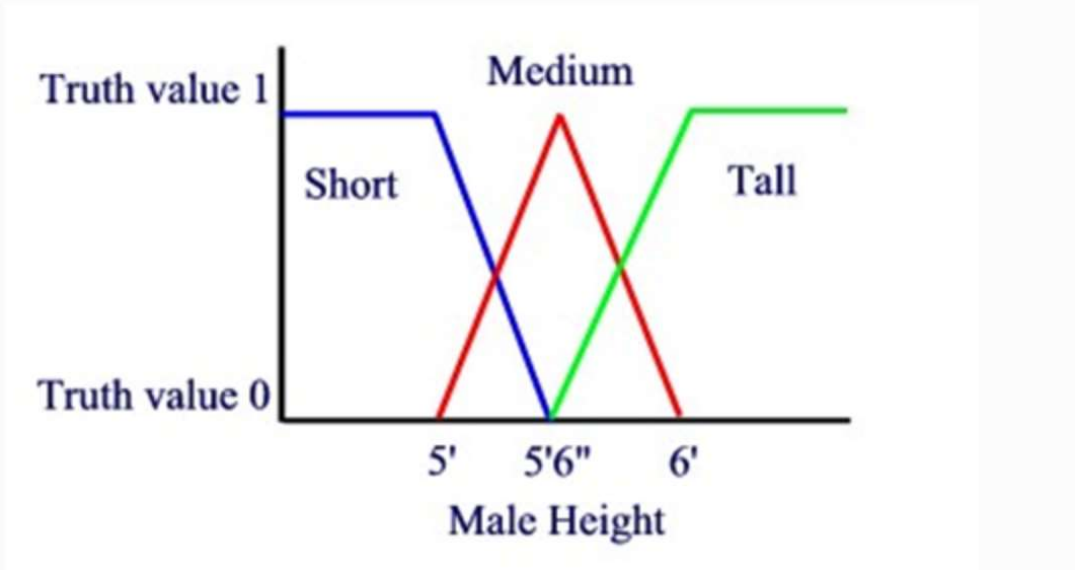
https://www.kdnuggets.com/2017/05/must-know-most-useful-number-clusters.html

Fuzzy Means

Data can belong to one or more clusters

Each point gets a probability to fitting into a cluster

The closer a point is to the middle of a cluster the more likely it is to belong to that group



https://medium.com/analytics-vidhya/fuzzy-sets-fuzzy-c-means-clustering-algorithm-ac5c4386396b

The k-means Clustering Classifier

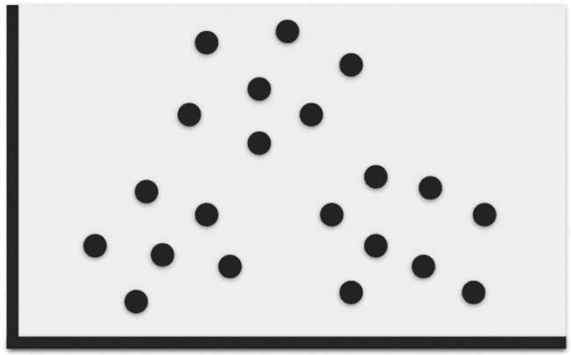
represent the training data with a number of data centers known as prototypes.

training data are reduced to a set of k representatives.

the test set point are compared to the prototypes.

Once these prototype centers are established, the test data are assigned to the class of the closest prototype.

1. Initialise random centroids
2. Until convergence:
   - Assign step
   - Update step
3. End



https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c

The k-means Clustering Classifier (cont)

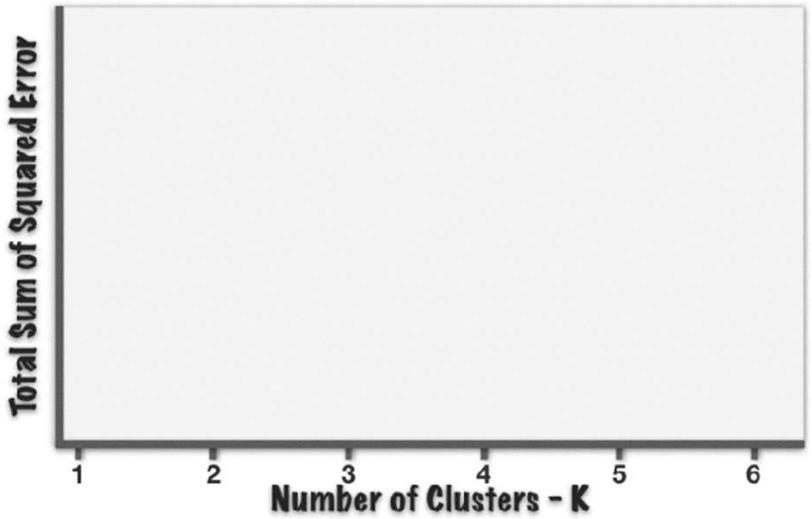
In this approach the k stands for the number of prototype centers.

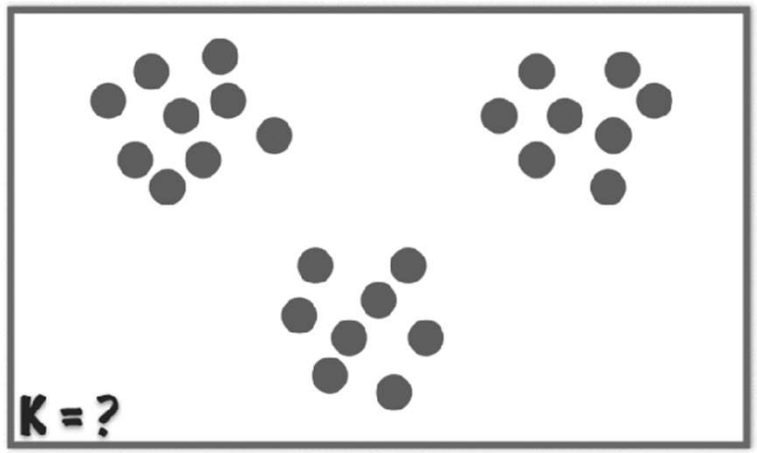
The position of the prototypes determines the boundary, and the number of prototypes chosen to represent each class determines the complexity of the boundary.

The larger the value of k, the more complicated the boundary. The value of k is directly related to machine capacity.

The number of prototypes is selected by the user and the prototype centers are positioned during a training period.

Finding the “Elbow”





https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c

Prototype Positioning - LVQ method

There are several different methods for training the prototypes. The method described here is the learning vector quantization

In the LVQ method

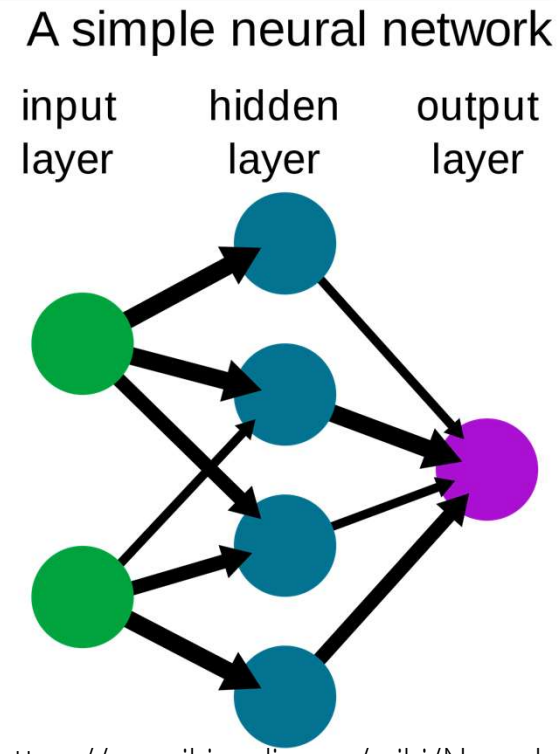
- initial prototypes are placed randomly within each class.
- A random training data point is selected and the closest prototype is found.
- If that prototype is of the same class as the training point, the prototype is moved toward the training point.
- If not, the prototype is moved away from the training point.
- The amount of movement is proportional to the distance and a learning rate constant.

The procedure begins with a smaller learning rate constant and continues until the learning rate constant is zero.

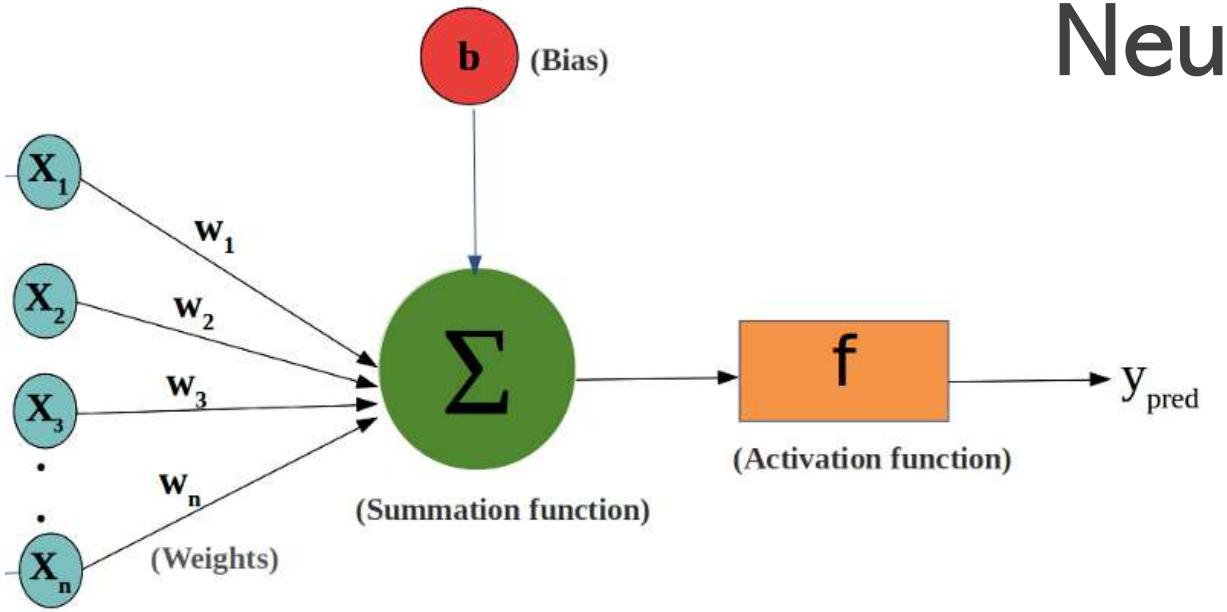
Deep Learning & Neural Networks

Neural Networks

- Artificial neural networks
  - Mimic the way neurons in the brain work
- Hidden layer contains weights, interconnected nodes, inhibitory and excitatory pathways
- Nodes are all connected to each other
- Every node is connected to next node
- Backbone for deep learning



https://en.wikipedia.org/wiki/Neural_network

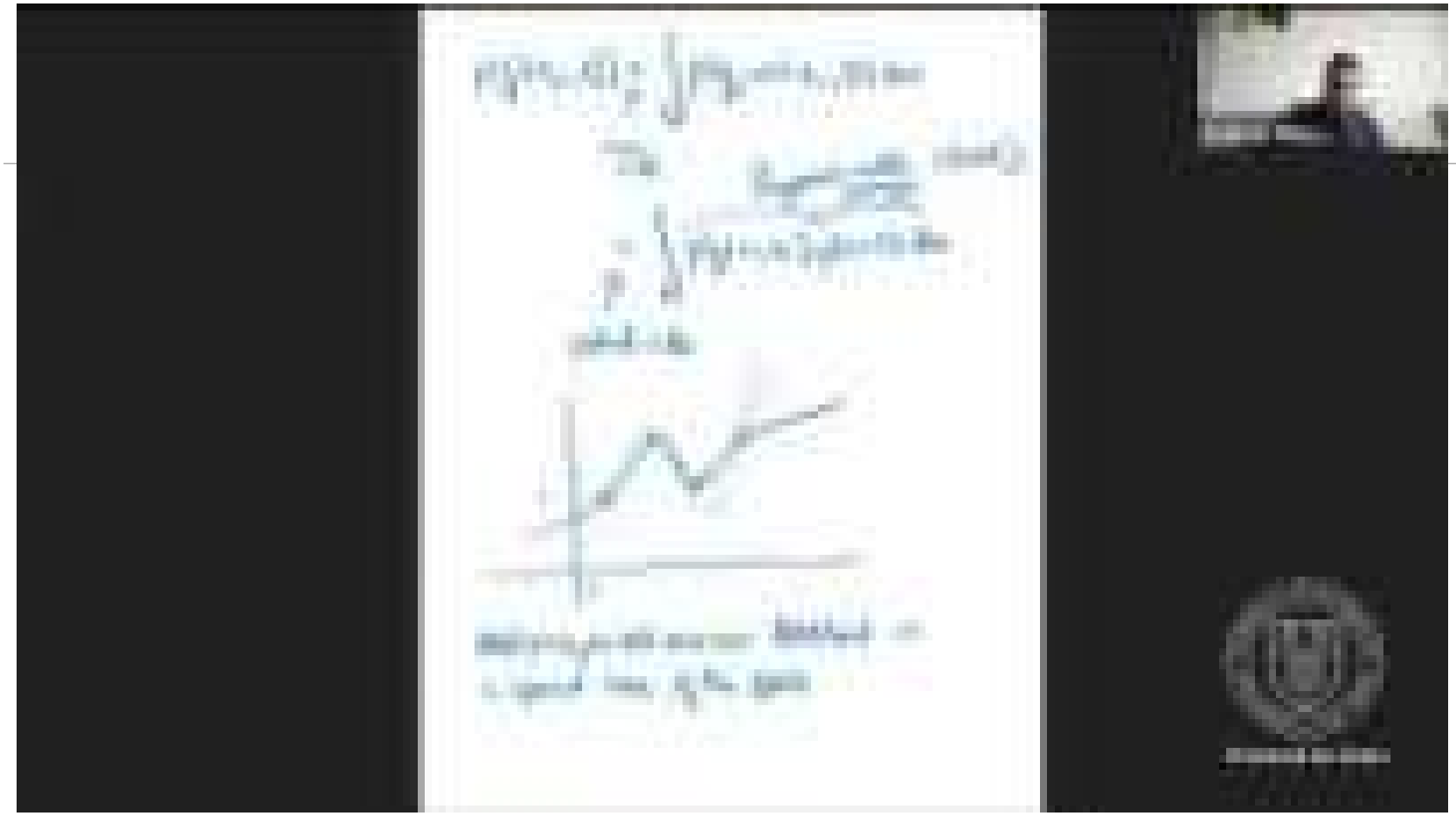


Neural Networks

- Connections all have weights and biases w, b
- Weight and bias terms are what the model optimizes
- Each level has a non-linear activation function f or σ
  - f(X*w +b)
- Forces nonlinearity into the system

Allows for complicated input output relationships

Example

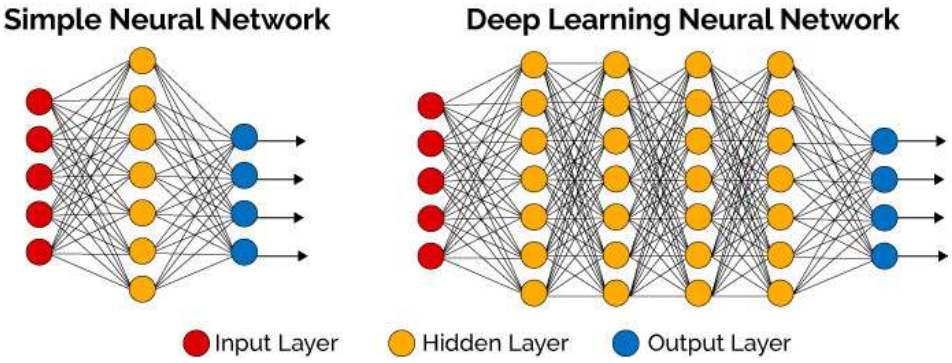


https://youtu.be/AissM0v__5s?t=3123

Neural Networks

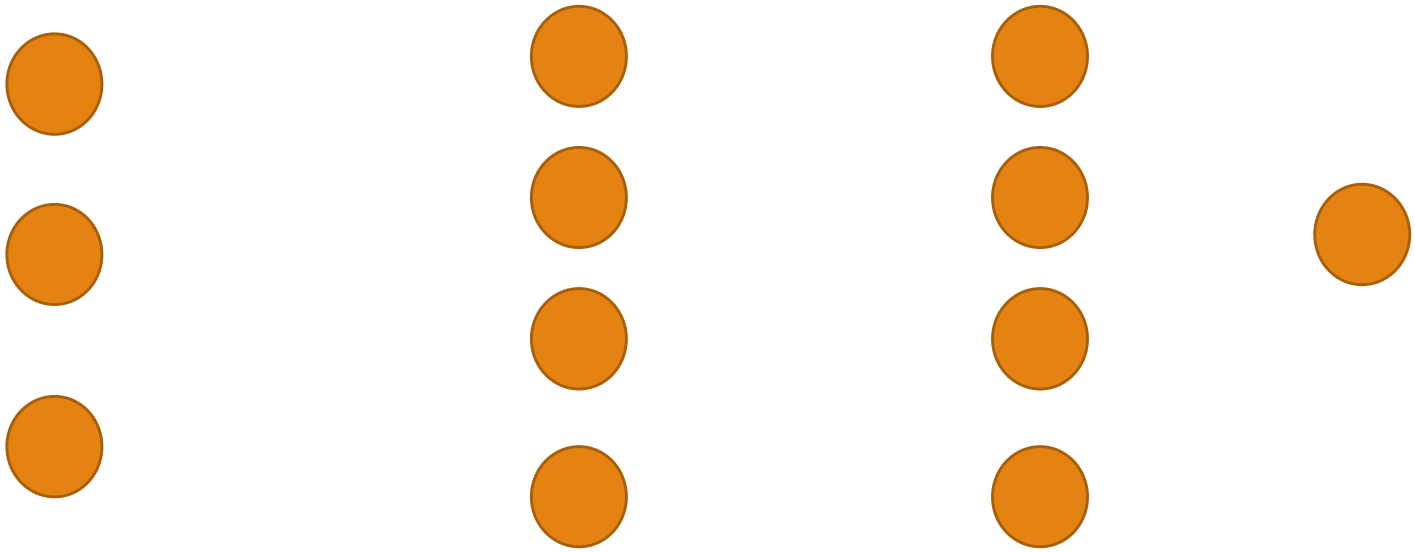
- Can do anything with just one hidden layer as long as there are enough nodes
  - Like a piecewise function to approximate a continuous function
  - if you have enough pieces you can make anything
- Node in layer uses all nodes in previous layer as input
- If we increase the number of nodes, we increase the computation time drastically…
- What if we add layers instead of adding nodes?

Deep Learning



- Can be used in unsupervised or supervised settings
- Good for large data sets
- Layers cascade
  - $\sigma_1(X*w_1 + b_1) \rightarrow \sigma_2(\sigma_1(X*w_1 + b_1)\ w_2 + b_2)$
- Can also be looked at as matrix transformations occurring at each layer
  - $\phi = W^t x \rightarrow \phi' = \phi U^t$
- More hidden layers allow for reusing functions which can simplify the algorithm
- Each weight vector (node) learns differently

Example



Example

Create an algorithm to sort photos of leaves vs faces

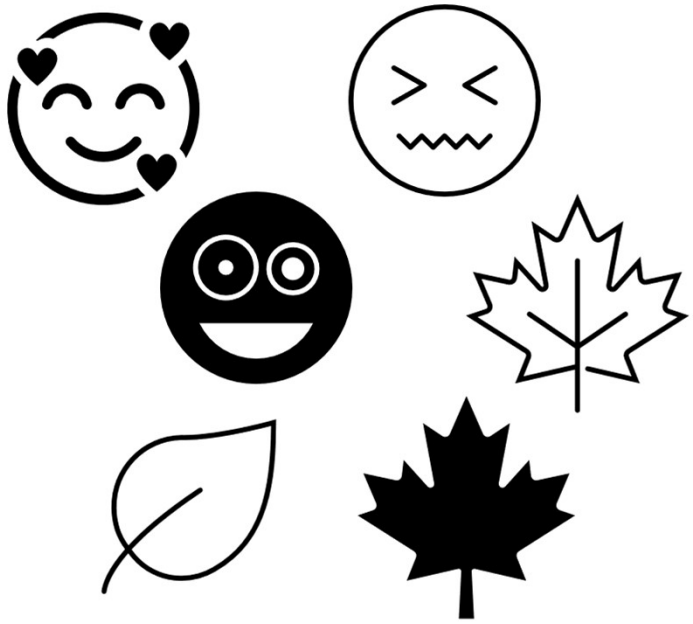
Normal supervised learning
- give it factors
- we have to know what factors are useful to start with

What factors do we give it?
- fractal dimension
- colour
- histogram

What if we have 1000 classes with thousands of images in each…

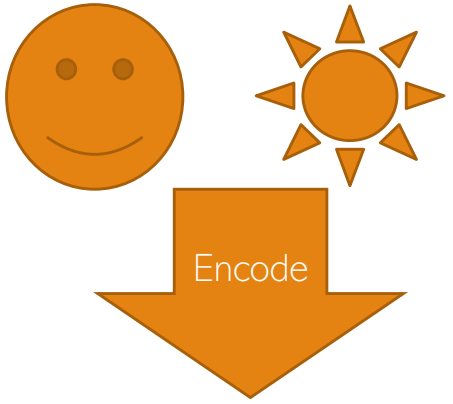
Deep learning creates a nonlinear model that can generalize the data well

Small cost – just need the labelled set, or a labelled subset



Example: Auto Encoder

- High dimensional encoder →Compression latency representation→ reconstruct using decoder
- Objective: Algorithm optimizes itself to reconstruct data from the reduced dimensionality PCA data
- Self-training and can be used for unsupervised anomaly detection
- Take data down to latent small dimension representation
- Normal vs anomaly may be more obvious at this level
  - Faces may fall between -1 and 1
  - Sun may be ~-4



PCA representation: -1 -4

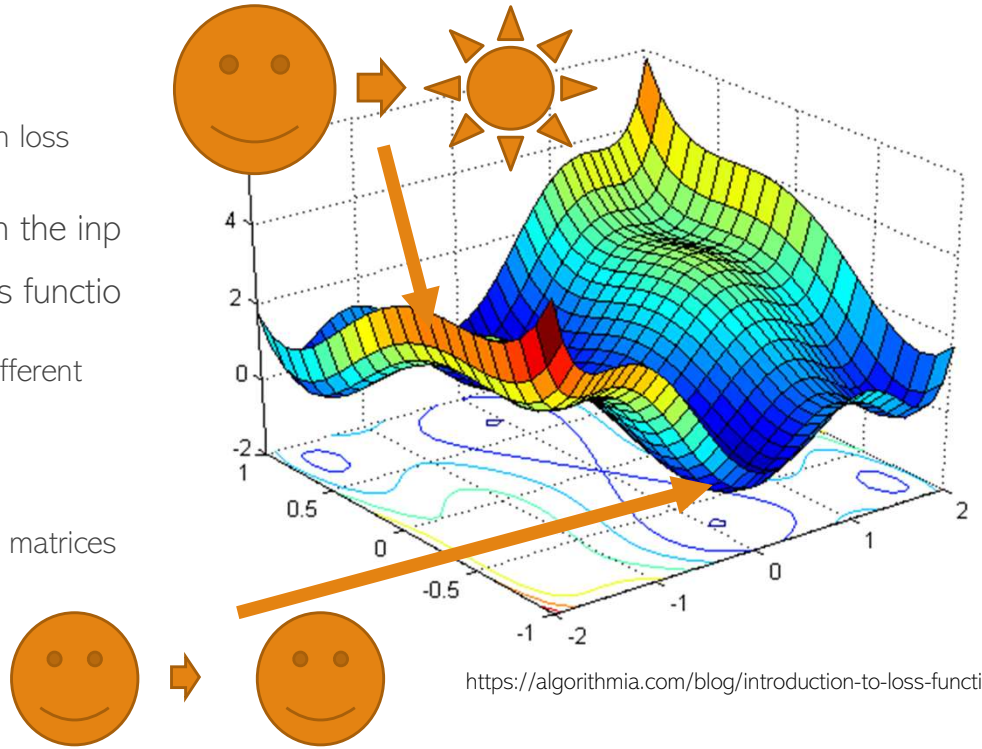
Loss Function

Mathematical Goal?

- Minimize the mean squares error
- i.e. minimize the loss function
- NN keeps trying to optimize to step closer to local minimum in loss function

Loss function is a function of weights and is dependent on the inp

- In general, deep learning algorithms have complicated loss functio
  - many high and low points
  - Want to find deepest points which can be done using many different random inputs
- Stochastic gradient descent
  - Iterative model
  - Start with random values and slowly tailor your transformation matrices you go
  - Move along loss function towards local minima
  - Uses calculus to do this



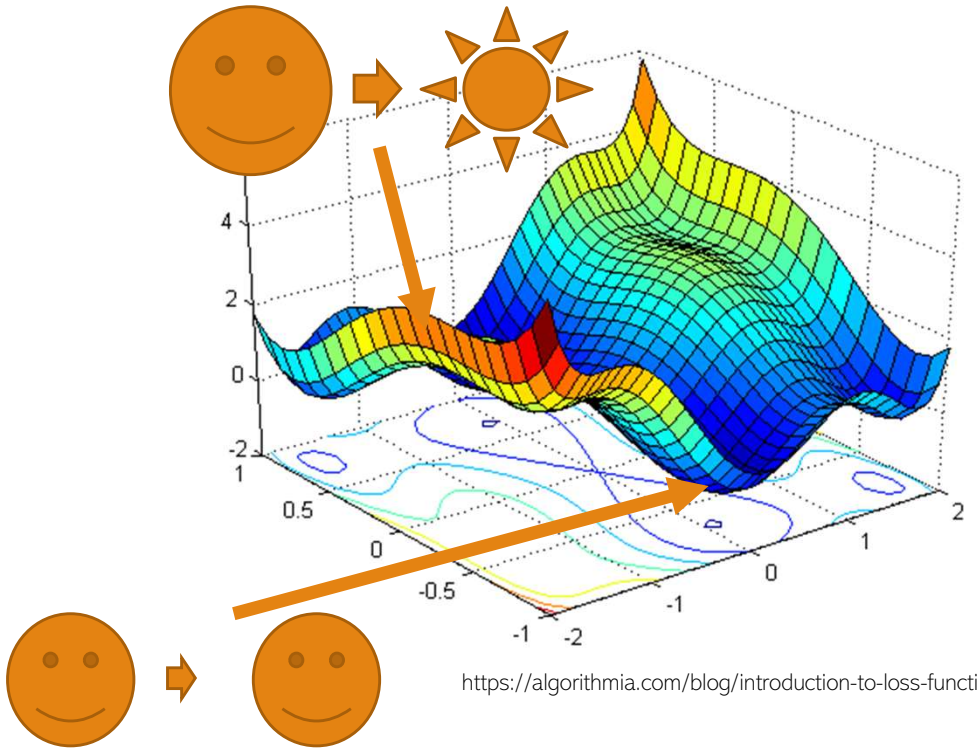
https://algorithmia.com/blog/introduction-to-loss-functions

Loss Function

Mathematical Goal?

- Minimize the mean squares error
- i.e. minimize the loss function
- NN keeps trying to optimize to step closer to local minimum in loss function

Loss function is a function of weights and is dependent on the input



https://algorithmia.com/blog/introduction-to-loss-functions

Picking a ML method

Have to understand your model to know if it is really working for your application

Need ML model to be generalizable

Model should work for data from many different sets and sources

Different models work best for different sizes of data sets

Is the cost of labelling worth it?

Is your data homogenous?

Using Supervised Learning to Predict Heart Attacks

Cardiologist wants to predict whether someone will have a heart attack within a year.

1) They have data on previous patients, including age, weight, height, and blood pressure, 6 lead ECG and HRV measures

2) They know whether those previous patients had heart attacks within a year.

3) So the problem is combining the existing data into a model that can predict whether a new person will have a heart attack within a year.

Modelling steps

1) Do we have patient consent to use their data?
2) Do we have a homogeneous data set?
3) CRD? RCBD?
4) Do we need all this data?
   - ECG → ICA
   - HRV→ just use Average N N intervals?
   - PCA→ Maybe blood pressure has limited variance between patients
5) Do we want more data?
   - STFT or wavelet transform ECG to look for changes to frequency
   - Cosinor to look for changes in BP throughout the day?
   - Is there fractal self similarity to the HRV throughout the period?

Modelling steps

6) Model
   - Linear regression?
   - Logistic regression?
   - Machine learning?

7) Last but not least…
   - Statistics