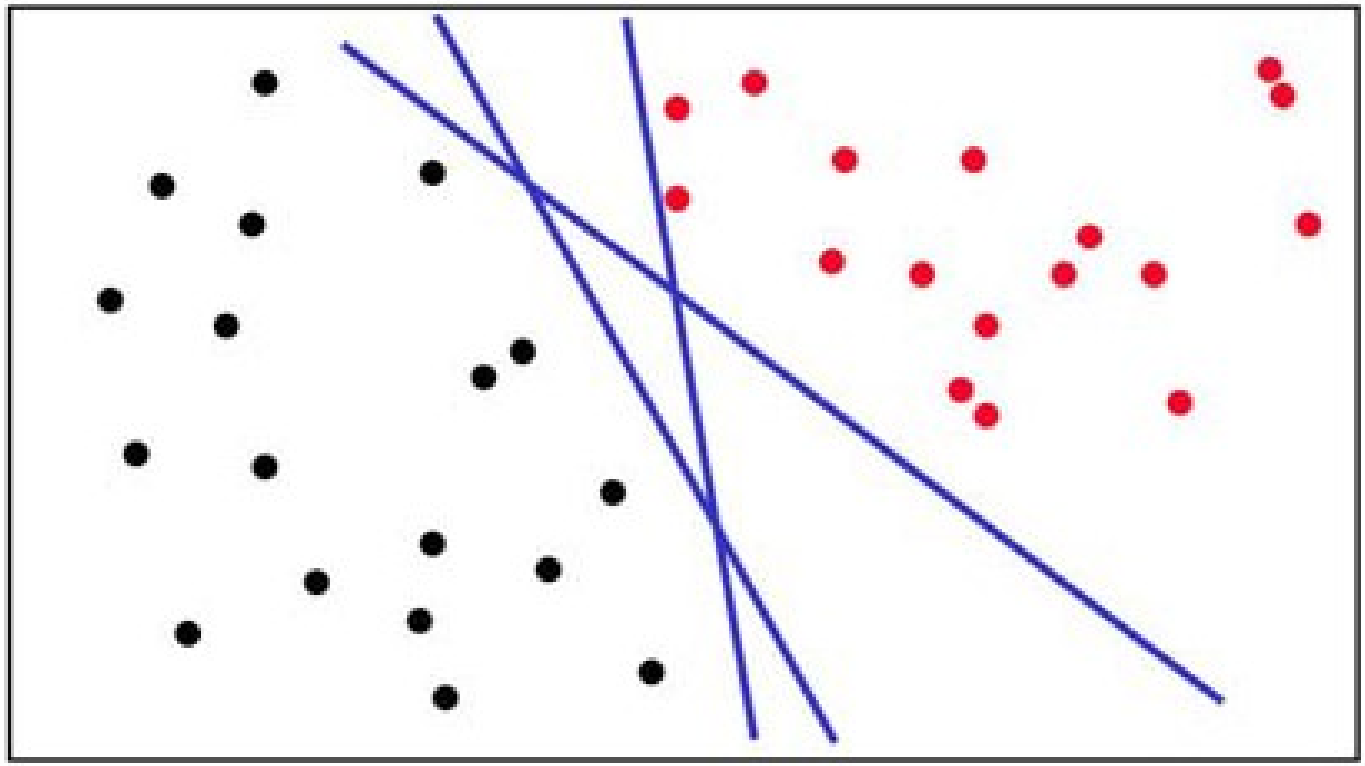
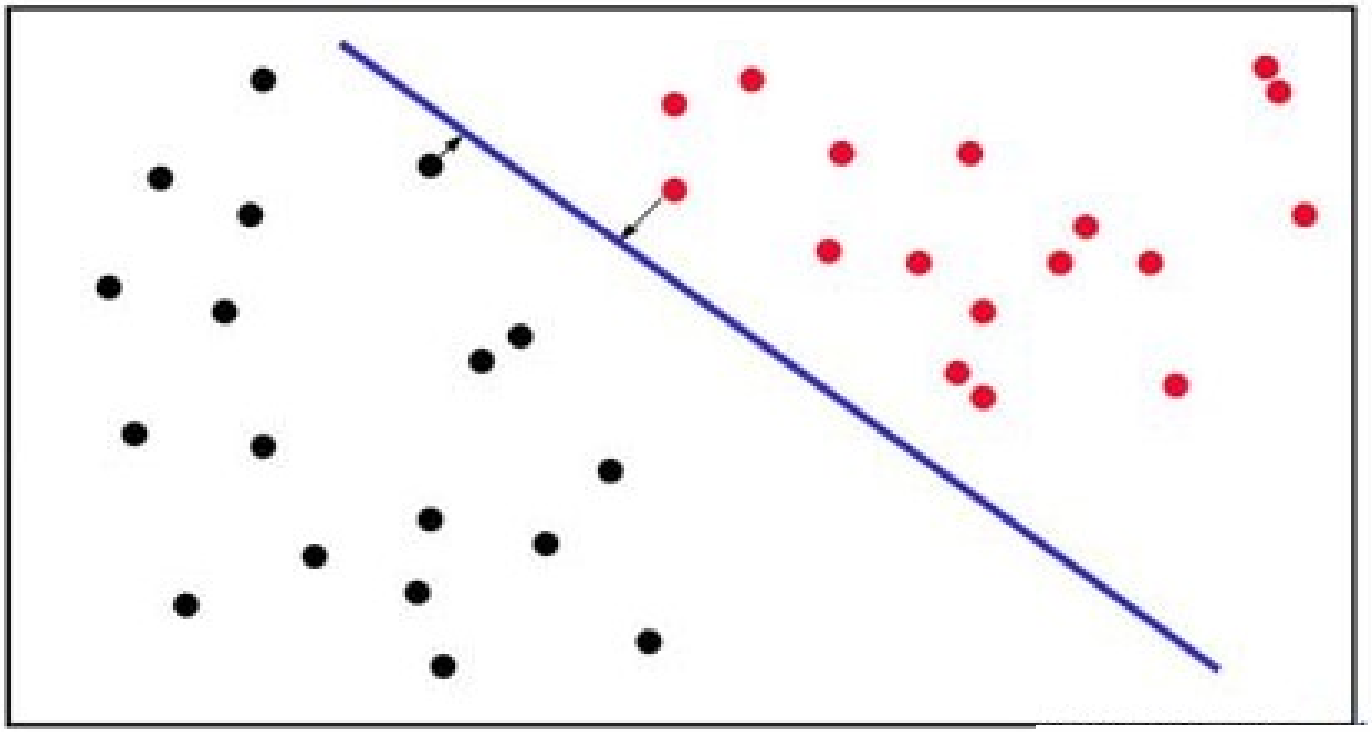
Separator and Margin

Which Separator?



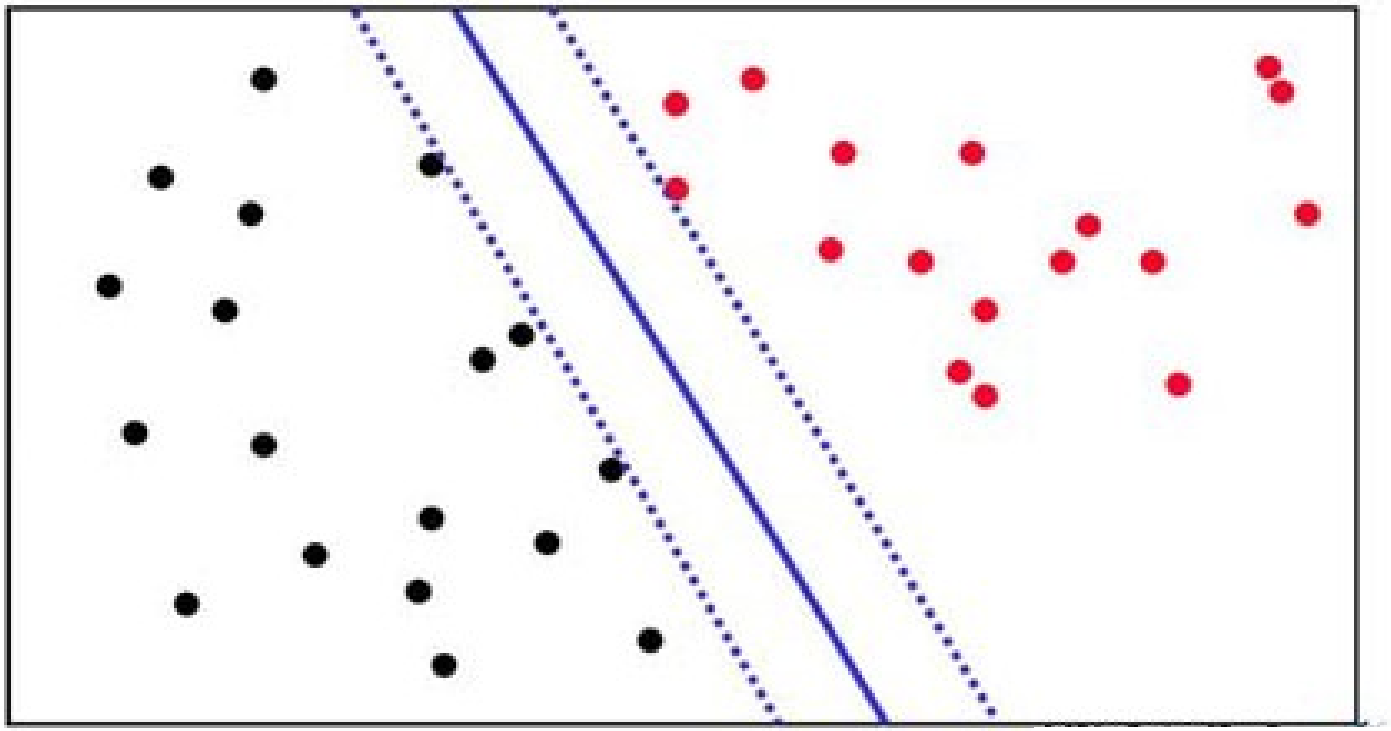
Which Separator?

Maximize the margin to closest points



Which Separator?

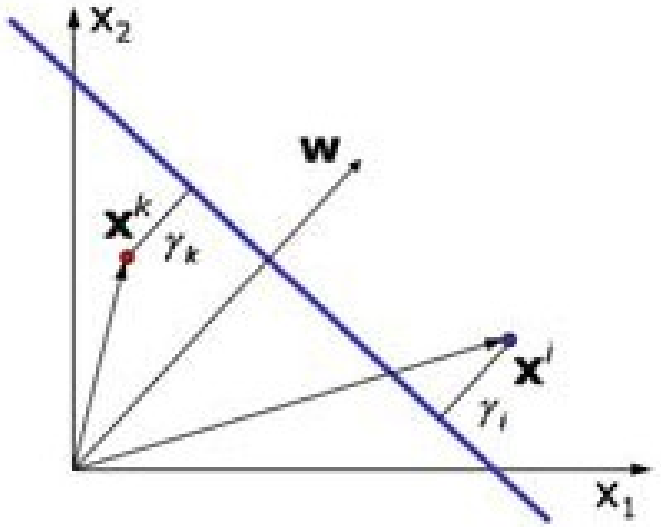
Maximize the margin to closest points



Margin of a point

$$\gamma^i \equiv y^i(\mathbf{w} \cdot \mathbf{x}^i + b)$$

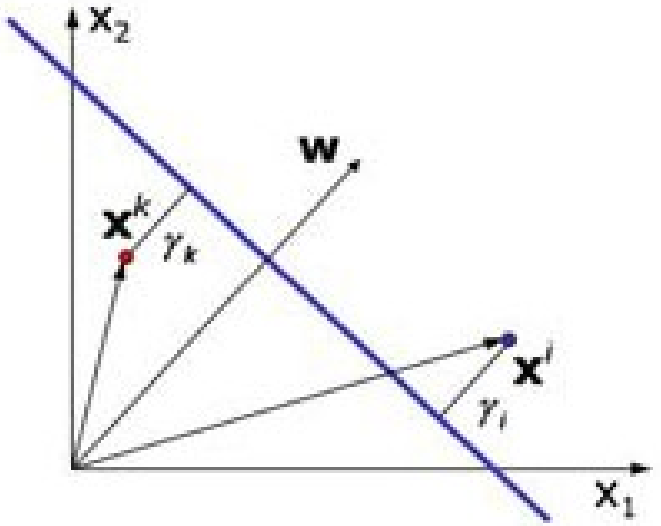
- proportional to perpendicular distance of point $\mathbf{x}^i$ to hyperplane



Margin of a point

$$\gamma^i \equiv y^i(\mathbf{w} \cdot \mathbf{x}^i + b)$$

- proportional to perpendicular distance of point $\mathbf{x}^i$ to hyperplane
- geometric margin is $\gamma^i / \|\mathbf{w}\|$

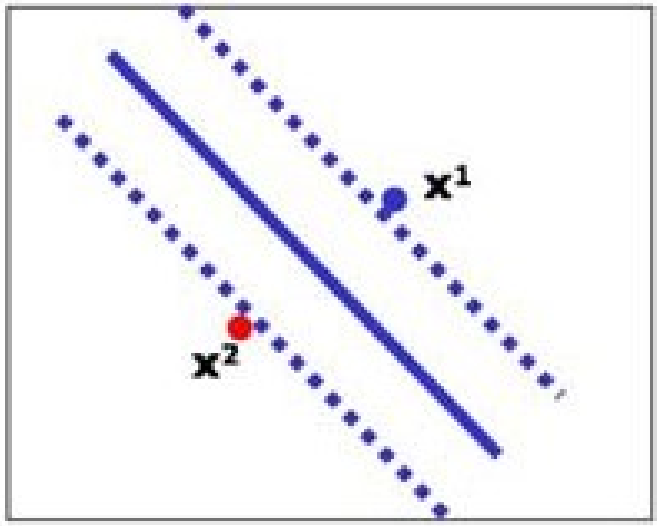


Margin

$$\gamma^i \equiv y^i(\mathbf{w} \cdot \mathbf{x}^i + b)$$

- Scaling **w** changes value of margin but not actual distances to separator (geometric margin)
- Pick the margin to closest positive and negative points to be 1

$$+1(\mathbf{w} \cdot \mathbf{x}^1 + b) = 1$$
$$-1(\mathbf{w} \cdot \mathbf{x}^2 + b) = 1$$



Margin

- Pick the margin to closest positive and negative points to be 1

$$+1(\mathbf{w} \cdot \mathbf{x}^1 + b) = 1$$
$$-1(\mathbf{w} \cdot \mathbf{x}^2 + b) = 1$$

- Combining these

$$\mathbf{w} \cdot (\mathbf{x}^1 - \mathbf{x}^2) = 2$$

- Dividing by length of **w** gives perpendicular distance between dashed lines (2 **x** geometric margin)

$$\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot (\mathbf{x}^1 - \mathbf{x}^2) = \frac{2}{\|\mathbf{w}\|}$$

Picking w to Maximize Margin

- Pick **w** to maximize geometric margin

$$\frac{2}{\|\mathbf{w}\|}$$

- or, equivalently, minimize

$$\|\mathbf{w}\| = \sqrt{\mathbf{w} \cdot \mathbf{w}}$$

- or, equivalently, minimize

$$\frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} = \frac{1}{2}\sum_j w_j^2$$

Picking w to Maximize Margin

- Pick **w** to maximize geometric margin

$$\frac{2}{\|\mathbf{w}\|}$$

- or, equivalently, minimize

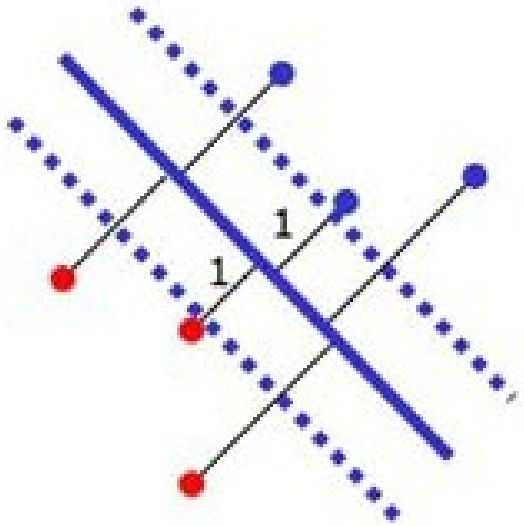
$$\frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} = \frac{1}{2}\sum_j w_j^2$$

- while classifying points correctly

$$y^i(\mathbf{w}\cdot\mathbf{x}^i + b) \geq 1$$

- or, equivalently,

$$y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1 \geq 0$$



Constrained Optimization

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1 \geq 0, \ \forall_i$$

Picking w to Maximize Margin

- Pick **w** to maximize geometric margin

$$\frac{2}{\|\mathbf{w}\|}$$

- or, equivalently, minimize

$$\|\mathbf{w}\| = \sqrt{\mathbf{w} \cdot \mathbf{w}}$$

- or, equivalently, minimize

$$\frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} = \frac{1}{2}\sum_j w_j^2$$

Constrained Optimization

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1 \geq 0, \ \forall_i$$

Convert to unconstrained optimization by incorporating the constraints as an additional term

$$\min_{\mathbf{w}}\left(\frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i\left[y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1\right]\right) \quad \alpha_i \geq 0, \forall_i$$

To minimize expression:

- minimize first (original) term, and
- maximize second (constraint) term
  - since $\alpha_i > 0$, encourages constraints to be satisfied
  - but we want least "distortion" of original term...

Constrained Optimization

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1 \geq 0, \ \forall_i$$

Convert to unconstrained optimization by incorporating the constraints as an additional term

$$\min_{\mathbf{w}}\left(\frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i\left[y^i(\mathbf{w}\cdot\mathbf{x}^i + b) - 1\right]\right) \quad \alpha_i \geq 0, \forall_i$$

Lagrange multipliers

To minimize expression:

- minimize first (original) term, and
- maximize second (constraint) term
  - since $\alpha_i > 0$, encourages constraints to be satisfied
  - but we want least "distortion" of original term...

Method of Lagrange multipliers

Maximizing the Margin

$$L(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \left[y^i (\mathbf{w} \cdot \mathbf{x}^i + b) - 1 \right]$$

Maximizing the Margin

$$L(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \left[y^i (\mathbf{w} \cdot \mathbf{x}^i + b) - 1 \right]$$

Minimized when: $\mathbf{w}^* = \sum_i \alpha_i y^i \mathbf{x}^i$ $\quad$ $\sum_i \alpha_i y^i = 0$

Maximizing the Margin

$$L(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \alpha_i \left[y^i(\mathbf{w} \cdot \mathbf{x}^i + b) - 1\right]$$

Minimized when: $\mathbf{w}^* = \sum_i \alpha_i y^i \mathbf{x}^i$ $\quad$ $\sum_i \alpha_i y^i = 0$

Substituting **w*** into L yields dual Lagrangian:

$$L(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{k=1}^{m} \alpha_i \alpha_k y_i y_k \mathbf{x}_i \mathbf{x}_k$$

Only dot products of the feature vectors appear

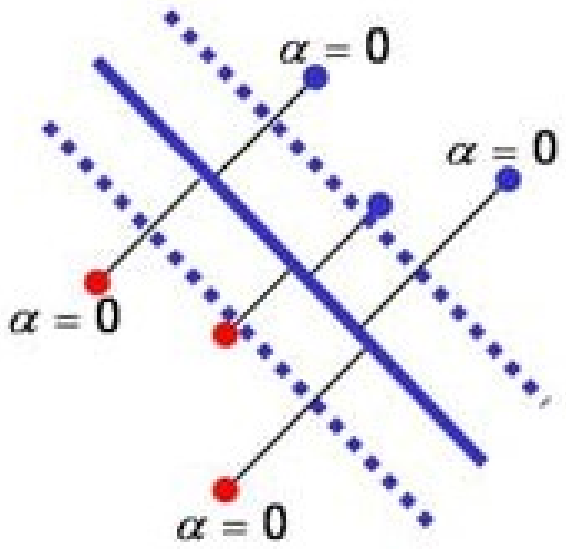
Dual Lagrangian

$$\max_{\alpha} L(\alpha) \quad \text{subject to} \quad \sum_{i} \alpha_i y^i = 0 \quad \text{and} \quad \alpha_i \geq 0, \forall i$$

Dual Lagrangian

$$\max_{\alpha} L(\alpha) \text{ subject to } \sum_{i} \alpha_i y^i = 0 \text{ and } \alpha_i \geq 0, \forall i$$

In general, since $\alpha_i >= 0$, either
$\alpha_i = 0$: constraint is satisfied with no distortion at optimum w
or
$\alpha_i > 0$: constraint is satisfied with equality (in this case $\mathbf{x}^i$ is known as a support vector)



Dual Lagrangian

$$\max_{\alpha} L(\alpha) \text{ subject to } \sum_i \alpha_i y^i = 0 \text{ and } \alpha_i \geq 0, \forall i$$

In general, since $\alpha_i >= 0$, either

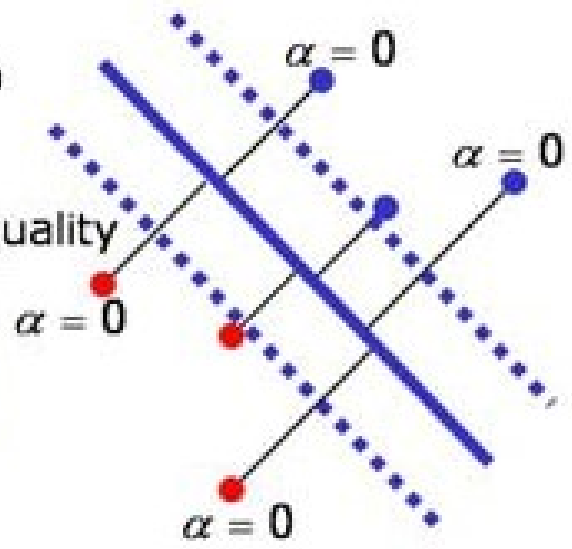
$\alpha_i = 0$: constraint is satisfied with no distortion at optimum w

or

$\alpha_i > 0$: constraint is satisfied with equality ($\mathbf{x}^i$ is known as a support vector)

$$\mathbf{w}^* = \sum_i \alpha_i y^i \mathbf{x}^i$$

$$b = 1/y^i - \mathbf{w}^* \mathbf{x}^i$$

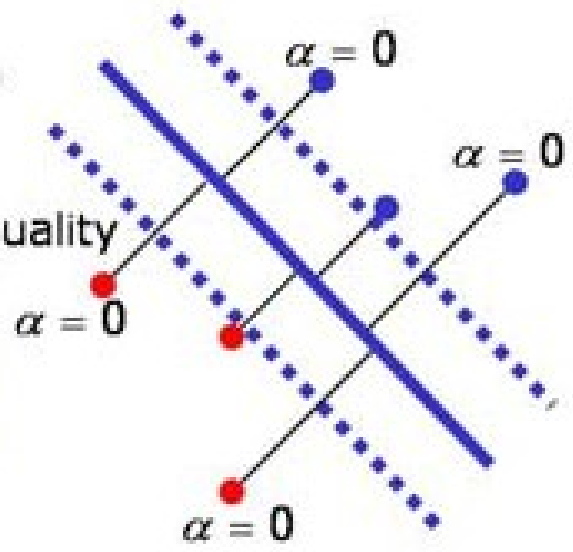


Dual Lagrangian

$$\max_{\alpha} L(\alpha) \quad \text{subject to} \quad \sum_{i} \alpha_i y^i = 0 \quad \text{and} \quad \alpha_i \geq 0, \forall i$$

In general, since $\alpha_i >= 0$, either
$\alpha_i = 0$: constraint is satisfied with no distortion at optimum w
or
$\alpha_i > 0$: constraint is satisfied with equality ($\mathbf{x}^i$ is known as a support vector)

$$\mathbf{w}^* = \sum_i \alpha_i y^i \mathbf{x}^i \qquad b = 1/y^i - \mathbf{w}^* \mathbf{x}^i$$



- Has a unique maximum vector
- Can be found using quadratic programming or gradient ascent

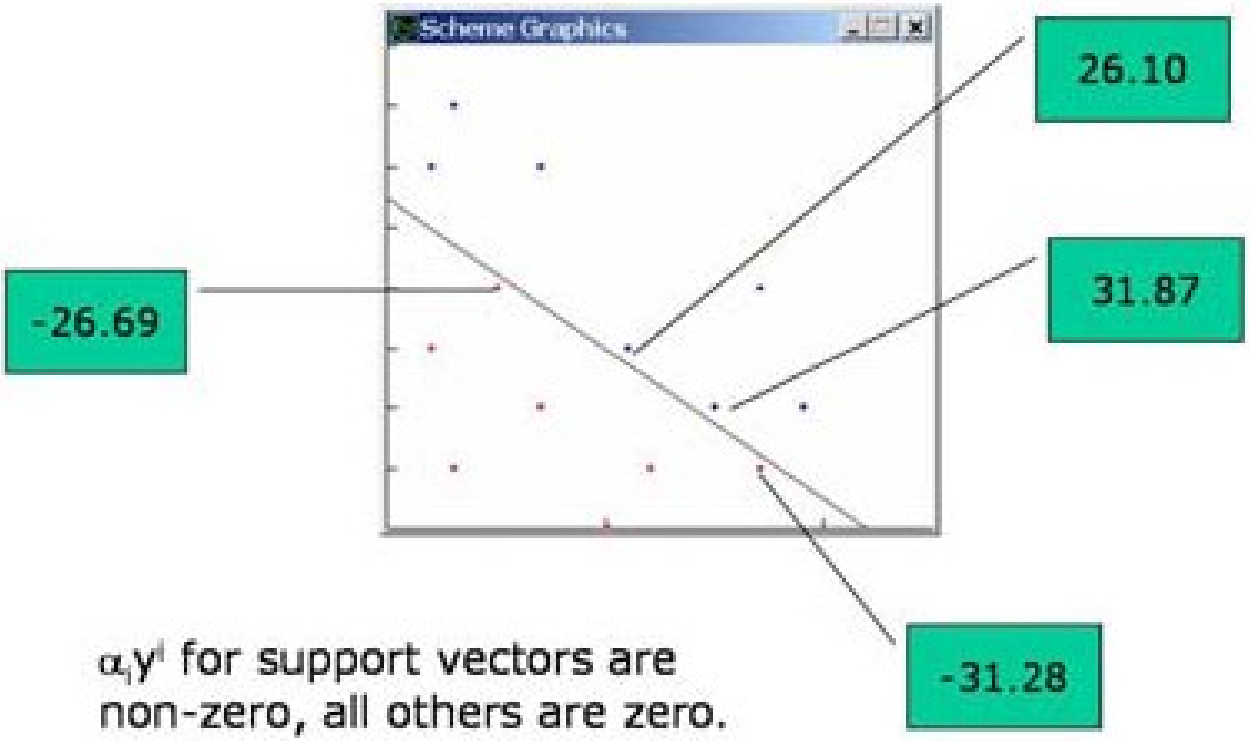
SVM Classifier

- Given unknown vector **u**, predict class (1 or -1) as follows:

$$h(\mathbf{u}) = sign\left(\sum_{i=1}^{k} \alpha_i y^i \mathbf{x}^i \cdot \mathbf{u} + b\right)$$

- The sum is over *k* support vectors

Bankruptcy Example



$\alpha_i y^i$ for support vectors are non-zero, all others are zero.

Key Points

- Learning depends only on dot products of sample pairs. Recognition depends only on dot products of unknown with samples.

SVM Classifier

- Given unknown vector **u**, predict class (1 or -1) as follows:

$$h(\mathbf{u}) = sign\left(\sum_{i=1}^{k} \alpha_i y^i \mathbf{x}^i \cdot \mathbf{u} + b\right)$$

- The sum is over *k* support vectors

Key Points

- Learning depends only on dot products of sample pairs. Recognition depends only on dot products of unknown with samples.
- Exclusive reliance on dot products enables approach to non-linearly-separable problems.
- The classifier depends only on the support vectors, not on all the training points.

Key Points

- Learning depends only on dot products of sample pairs. Recognition depends only on dot products of unknown with samples.
- Exclusive reliance on dot products enables approach to non-linearly-separable problems.
- The classifier depends only on the support vectors, not on all the training points.
- Max margin lowers hypothesis variance.

Key Points

- Learning depends only on dot products of sample pairs. Recognition depends only on dot products of unknown with samples.
- Exclusive reliance on dot products enables approach to non-linearly-separable problems.
- The classifier depends only on the support vectors, not on all the training points.
- Max margin lowers hypothesis variance.
- The optimal classifier is defined uniquely – there are no "local maxima" in the search space
- Polynomial in number of data points and dimensionality

BINARY LOGISTIC REGRESSION

in

Machine Learning

Logistic regression is a statistical model in which the response variable takes a discrete value and the explanatory variables can either be continuous or discrete. If the outcome variable takes only two values, then the model is called binary logistic regression model.

Assume that the outcomes are called positive (usually coded as *Y = 1) and negative (usually coded as Y = 0).*

Then the probability that a record belongs to a positive class, *P(Y = 1), using the binary logistic regression model is given by:*

$$P(Y=1)=\frac{e^{Z}}{1+e^{Z}} \qquad (1)$$

where

$$Z=\beta_0+\beta_1X_1+\beta_2X_2+\beta_3X_3+\cdots+\beta_mX_m$$

Here X1, X2, …, Xm are the independent variables or features.

The logistic regression has an S-shaped curve, and gives the class probability of an observation belonging to class labelled as 1, that is, *P(Y = 1)*.

The Equation (1) in the previous slide can be re-written as:

$$\ln\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = Z = \beta_0 + \beta_1 X_1 + \cdots + \beta_m X_m \qquad (2)$$

The left-hand side of this Equation (2) is a log natural of odds and is known as logit function; the right-had side is a linear function. Such models are called **generalized linear models (GLM)**. In GLM, the errors may not follow normal distribution and there exists a transformation function of the outcome variable that takes a linear functional form.

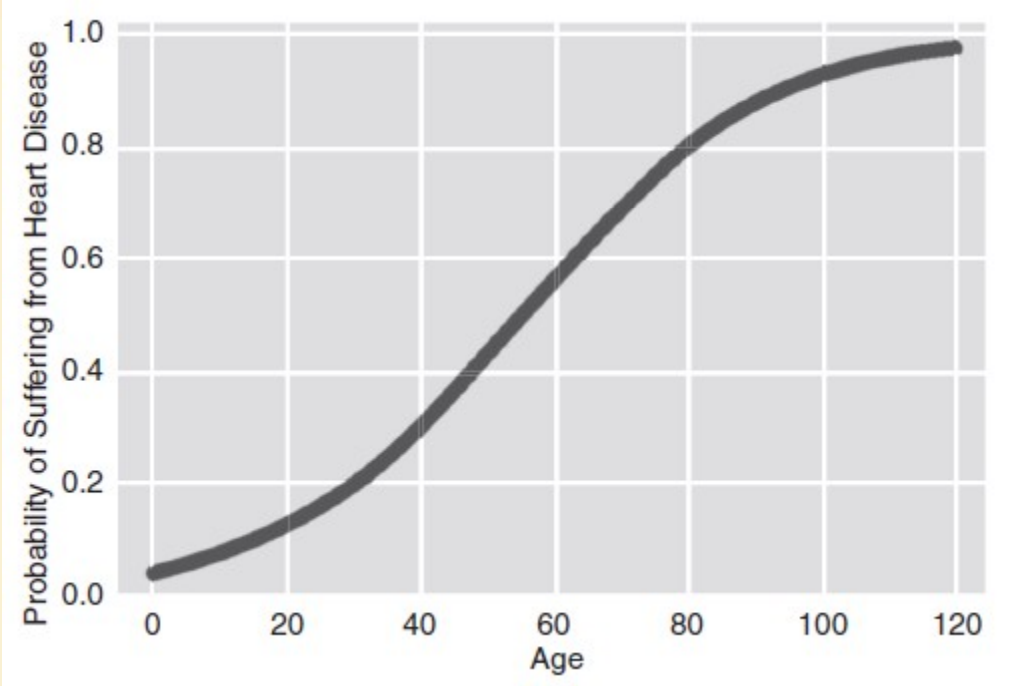
The basic logistic function given in the original equation (1) has an *S-shaped curve (thus also known as **Sigmoid function**).*

Consider an example of classifying a person **with heart disease and without heart disease [https://archive.ics.uci.edu/ml/datasets/heart+Disease]**.

That is, we are interested in finding the probability of a person suffering from heart disease as a function of age. The outcome variable is either someone suffers from heart disease or not and the independent variable (feature) is ***age***. Then a logistic or sigmoid function can be fit to explain the probability of suffering from heart disease with respect to age as shown in the Figure in the following slide.

The corresponding logistic function is given as:

$$P(\text{Heart Disease} = 1) = \frac{e^{\beta_0 + \beta_1 Age}}{1 + e^{\beta_0 + \beta_1 Age}}$$



Logistic regression model for predicting probability of suffering from Heart Disease.

SUPPORT VECTOR MACHINES

in

Machine Learning

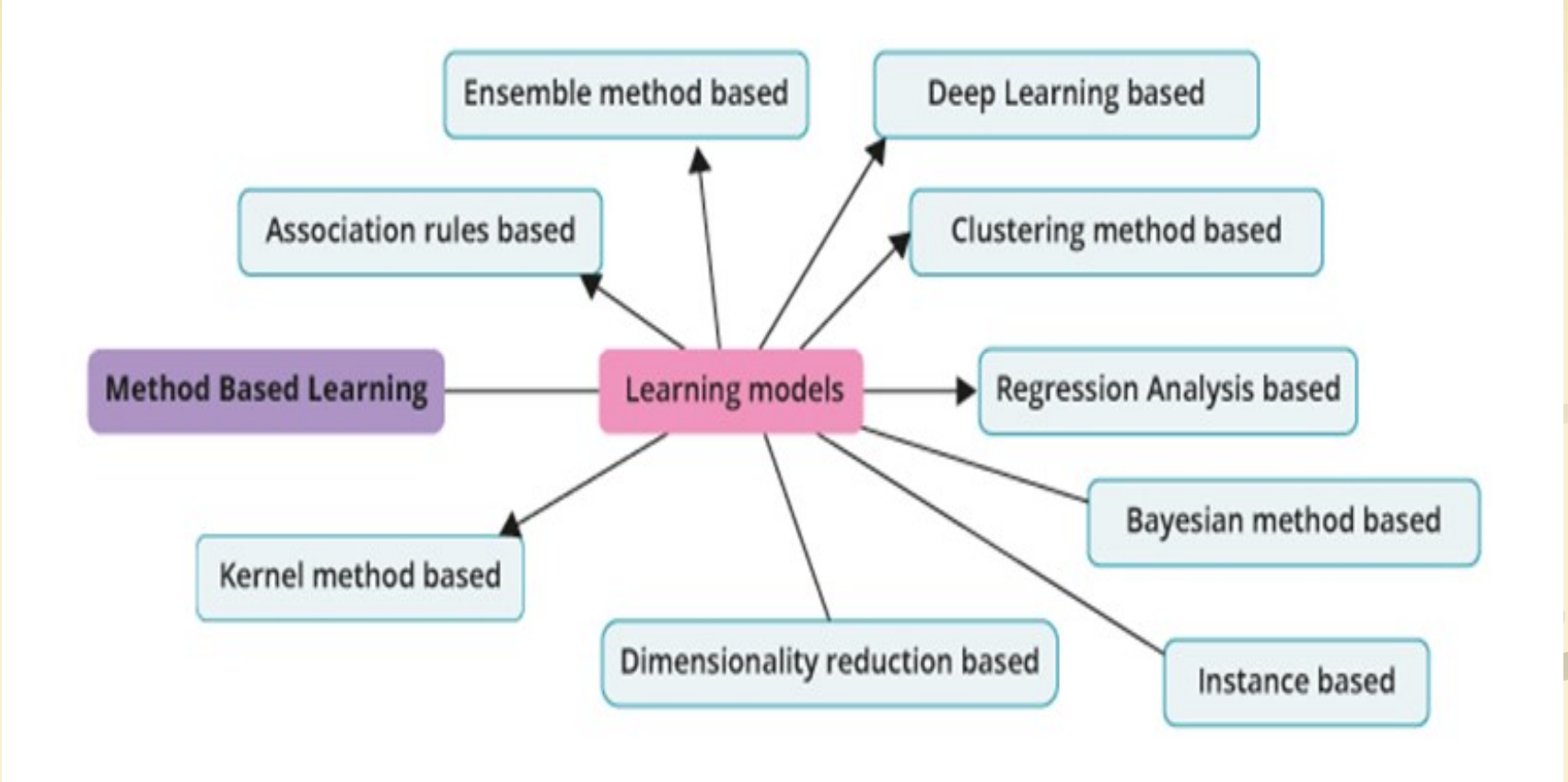
While many classifiers exist that can classify linearly separable data such as **Logistic Regression**, **Support Vector Machines** can handle highly non-linear problems using a kernel trick which implicitly maps the input vectors to higher-dimensional feature spaces.

The transformation rearranges the dataset in such a way that it is then linearly solvable. In this session we are going to look at how SVM works, learn about kernel functions, hyper parameters and pros and cons of SVM along with some of the real life applications of SVM.

Support Vector Machines (SVMs), also known as support vector networks, are a family of extremely powerful models which use method based learning and can be used in classification and regression problems.

They aim at finding decision boundaries that separate observations with differing class memberships. In other words, SVM is a discriminative classifier formally defined by a separating hyperplane.

Method Based Learning



Let us understand what **Kernel method** based learning is all about.

In simple terms, a kernel is a similarity function which is fed into a machine learning algorithm. It accepts two inputs and suggests the similarity. For example, suppose we want to classify images, the input data is a key-value pair (image, label). The image data is taken into consideration, features are computed, and a vector of features are fed into the Machine learning algorithm. But in the case of similarity functions, a kernel function can be defined which internally computes the similarity between images, and then feeds into the learning algorithm along with the images and label data. The outcome of this is a classifier.

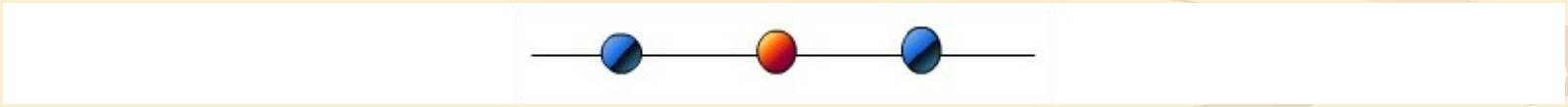
Perceptron frameworks or Support vector machines work with kernels and use vectors only. Here, the machine learning algorithms are expressed as dot products so that kernel functions can be used.

Feature vectors generally prefer kernels. Its ease of computing makes it one of the key reasons, also, feature vectors need more storage space in comparison to dot products. You can write Machine learning algorithms to use dot products and later map them to use kernels. This completely avoids the usage of feature vectors. This allows us to work with highly complex, efficient-to-compute, and yet high performing kernels effortlessly, without really developing multi-dimensional vectors.

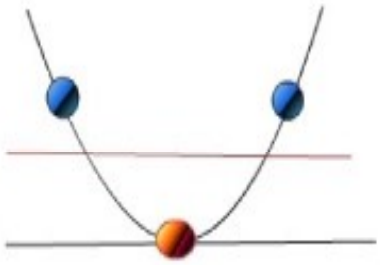
Kernel functions:

Let us understand what kernel functions are:

The figure shown below represents a 1D function using a simple 1-Dimensional example. Assume that given points are as follows, it will depict a vertical line and no other vertical lines will separate the dataset.



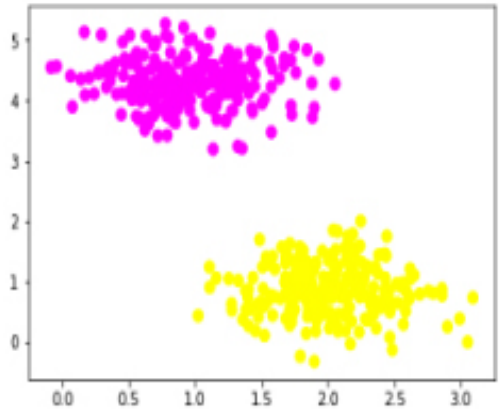
Now, if we consider a 2-Dimensional representation, as shown in the figure below, there is a hyperplane (an arbitrary line in 2-Dimensions) which separates red and blue points, which can be separated using Support Vector Machines.



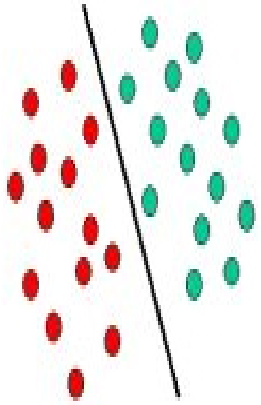
As we keep increasing dimensional space, the need to be able to separate data will eventually decrease. This mapping, x→(x1, x2), is called the kernel function. In case of growing dimensional space, the computations become more complex and kernel trick needs to be applied to address these computations cheaply.

What is Support Vector Machine?

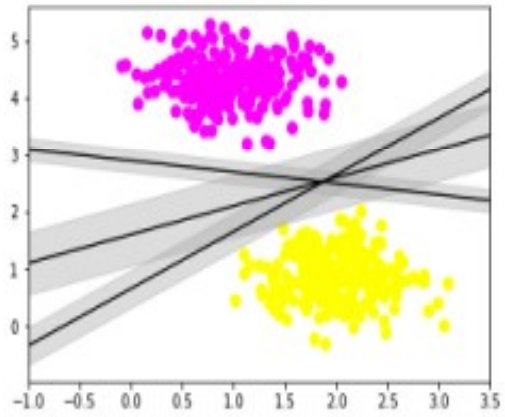
Support Vector Machine (SVM) is a **supervised machine learning algorithm** which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, each data is plotted in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. After that, we perform classification by locating the hyperplane which differentiates both the classes.



Support vector machine is based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects with different class memberships. For example, in the figure mentioned below, there are objects which belong to either class Green or Red. The separating line defines a boundary on the right side of which all objects are Green and to the left of which all objects are Red. Any new object (white circle) falling to the right is labeled, i.e., classified, as Green (or classified as Red should it fall to the left of the separating line).

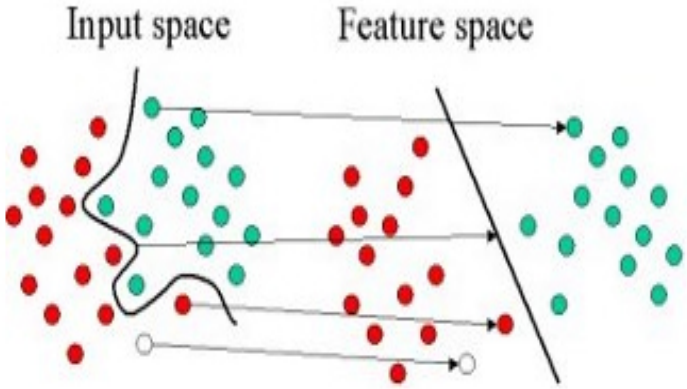


Support vector machines not only draw a line between two classes, but consider a region about the line of some given width.



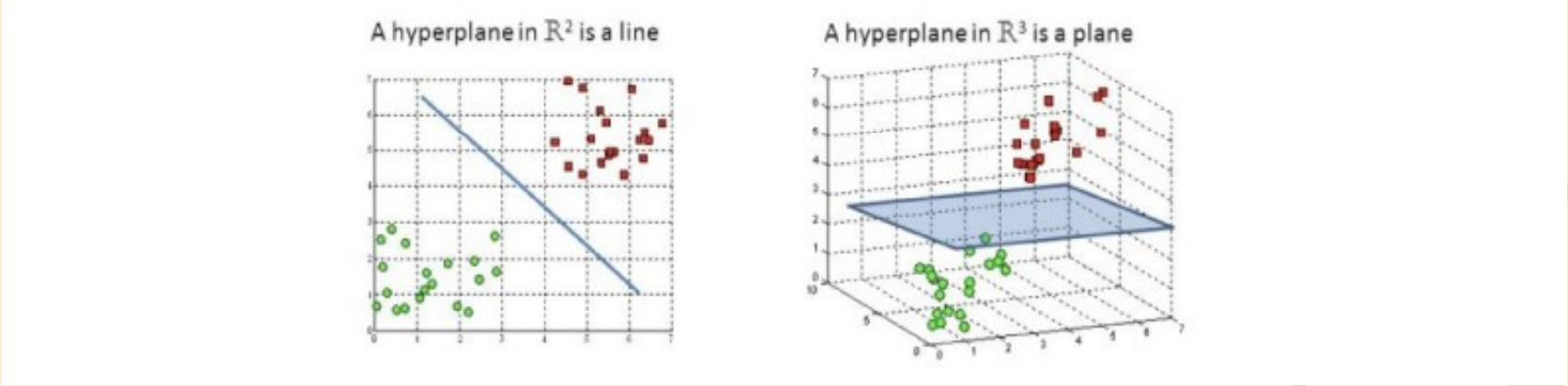
Another scenario, where it is clear that a full separation of the Green and Red objects would require a curve (which is more complex than a line). Classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyperplane classifiers. Support Vector Machines are particularly suited to handle such tasks.

The figure below shows the basic idea behind Support Vector Machines. Here you will see that the original objects (left side of the schematic) mapped, are rearranged using a set of mathematical functions called kernels. This process of rearranging objects is known as mapping or transformation. You will notice that the right side of the schematic is linearly separable. All we can do is find an optimal line that will separate red and green objects.



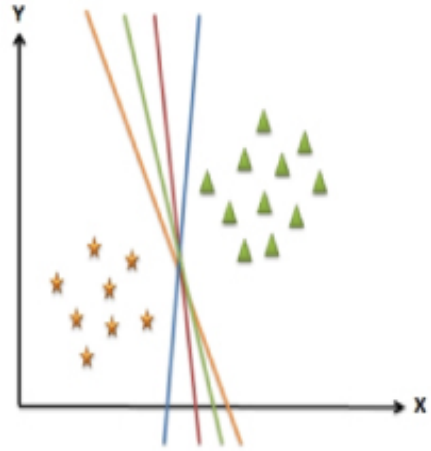
What is a hyperplane?

The goal of Support Vector Machine is to find the hyperplane which separates these two objects or classes.

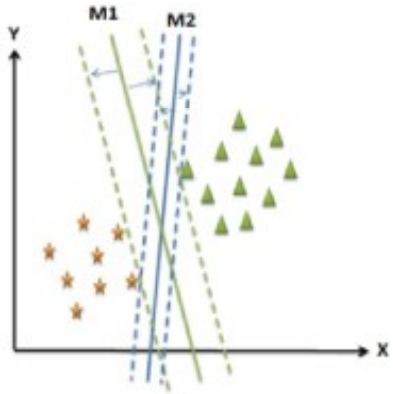


Let us consider another figure which shows some of the possible hyperplanes which can help in separating or dividing the dataset. It is the choice of the best hyperplane which is also the goal.

The best hyperplane is defined by the extent to which a maximum margin is left for both classes. The margin is the distance between the hyperplane and the closest point in the classification.



Let us consider two hyperplanes among all and then check the margins represented by M1 and M2. You will notice that margin M1 > M2, so the choice of the hyperplane which separates the best one is the new plane between the green and blue planes.



How do we find the right hyperplane?

Now, let us represent the new plane by a linear equation as:
$f(x) = ax + b$

Let us consider that this equation delivers all values ≥ 1 from the green triangle class and ≤ -1 for the gold star class. The distance of this plane from the closest points in both the classes is at least one; the modulus is one.
$f(x) \geq 1$ for triangles and $f(x) \leq 1$ or $|f(x)| = 1$ for star

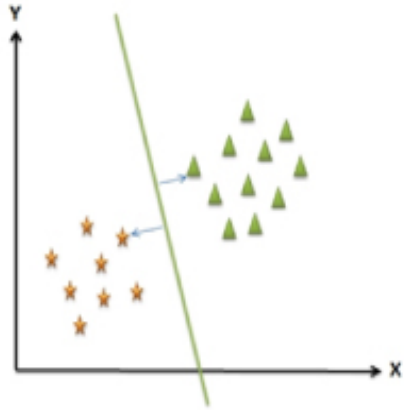
The distance between the hyperplane and the point can be computed using the following equation.
$M1 = |f(x)| / ||a|| = 1 / ||a||$

The total margin is $1 / ||a|| + 1 / ||a|| = 2 / ||a||$.

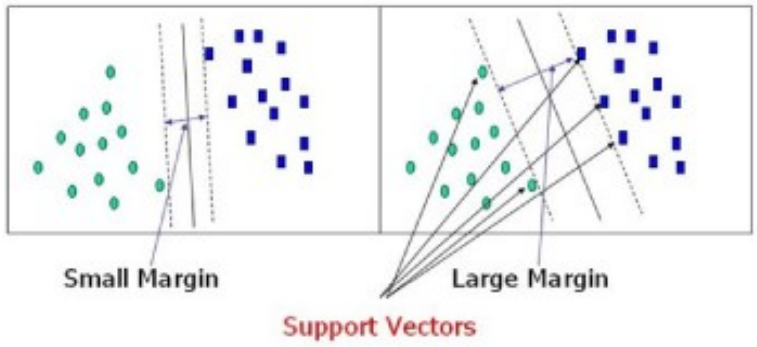
In order to maximize the separability, we will have to maximize the ||a|| value. This particular value is known as a weight vector. We can minimize the weight value which is a non-linear optimization task. One of the methods is to use the Karush-Kuhn-Tucker (KKT) condition, using the Lagrange multiplier λi.

$$\mathbf{a} = \sum_{i=0}^{N} \lambda_i y_i \vec{x}_i$$

$$\sum_{i=0}^{N} \lambda_i y_i = 0$$

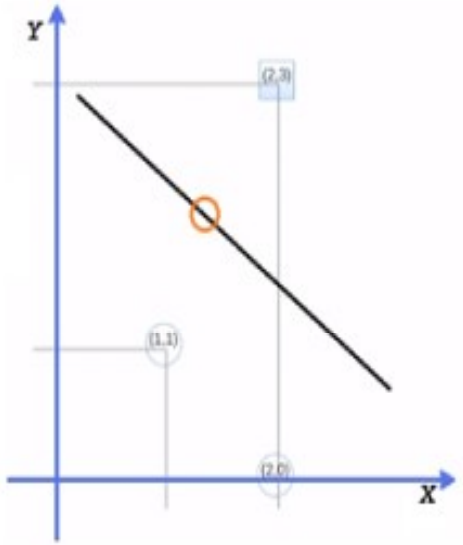


What is a support vector in SVM?



Let's take an example of two points between the two attributes X and Y. We need to find a point between these two points that has a maximum distance between these points.

This requirement is represented in the graph depicted here.
The optimal point is depicted using the red circle.



The maximum margin weight vector is parallel to the line from (1, 1) to (2, 3). The weight vector is at (1,2), and this becomes a decision boundary that is halfway between and in perpendicular, that passes through (1.5, 2).
So, $y = x1 + 2x2 - 5.5$ and the geometric margin is computed as $\sqrt{5}$.

Following are the steps to compute SVMs:

With w = (a, 2a) for the functions of the points (1,1) and (2,3) can be represented as shown here:

$a + 2a + \omega_0 = -1$ for the point (1,1)

$2a + 6a + \omega_0 = 1$ for the point (2,3)

The weights can be computed as follows:

$$\omega_0 = 1 - 8a \quad 3a + 1 - 8a = -1$$

$$\vdots \quad 5a = 2$$

$$a = \frac{2}{5}$$

$$\omega_0 = 1 - 8\frac{2}{5} = \frac{5-16}{5}$$

$$\omega_0 = -\frac{11}{5}$$

These are the support vectors:

$$\vec{w} = \left(\frac{2}{5}, \frac{4}{5} \right)$$

Lastly, the final equation is as follows:

$$g(\vec{x}) = \frac{2}{5} x_1 + \frac{4}{5} x_2 - \frac{11}{5}$$

$$g(\vec{x}) = x_1 + 2x_2 - 5.5$$

Large Margin Intuition:

In logistic regression, the output of linear function is taken and the value is squashed within the range of [0,1] using the sigmoid function. If the value is greater than a threshold value, say 0.5, label 1 is assigned else label 0.

In case of support vector machines, the linear function is taken and if the output is greater than 1 and we identify it with one class and if the output is -1, it is identified with another class. Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values([-1,1]) which acts as margin.

Cost Function and Gradient Updates:

In the SVM algorithm, we maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is called the hinge loss.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases} \qquad c(x, y, f(x)) = (1 - y * f(x))_+$$

Hinge loss function (function on the left can be represented as a function on the right). If the predicted value and the actual value are of the same sign, the cost is 0 . If not, we calculate the loss value.

We also add a regularization parameter the cost function. The objective of the regularization parameter is to balance the margin maximization and loss.

After adding the regularization parameter, the cost functions looks as below:

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

Loss function for SVM

Now that we have the loss function, we take partial derivatives with respect to the weights to find the gradients. Using gradients, we can update our weights.

$$\frac{\delta}{\delta w_k}\lambda \parallel w \parallel^2 = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k}\left(1 - y_i\langle x_i, w\rangle\right)_+ = \begin{cases} 0, & \text{if } y_i\langle x_i, w\rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

Gradients

When there is no misclassification, i.e our model correctly predicts the class of our data point, we only have to update the gradient from the regularization parameter.

$$w = w - \alpha \cdot (2\lambda w)$$

Gradient Update — No misclassification

When there is a misclassification, i.e. our model makes a mistake on the prediction of the class of our data point, we include the loss along with the regularization parameter to perform gradient update.

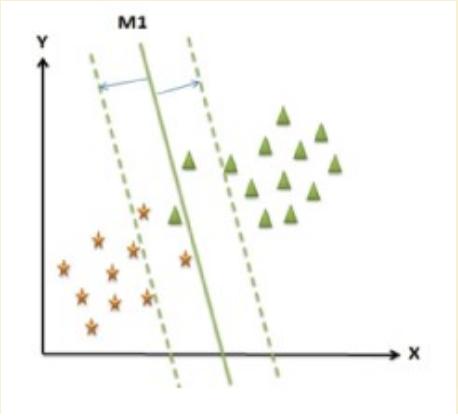
$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

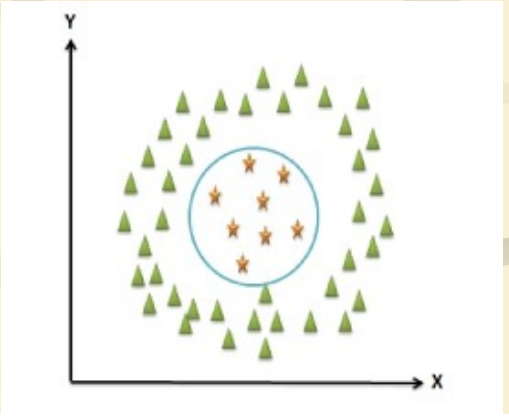
Gradient Update — Misclassification

But what happens when there is no clear hyperplane?

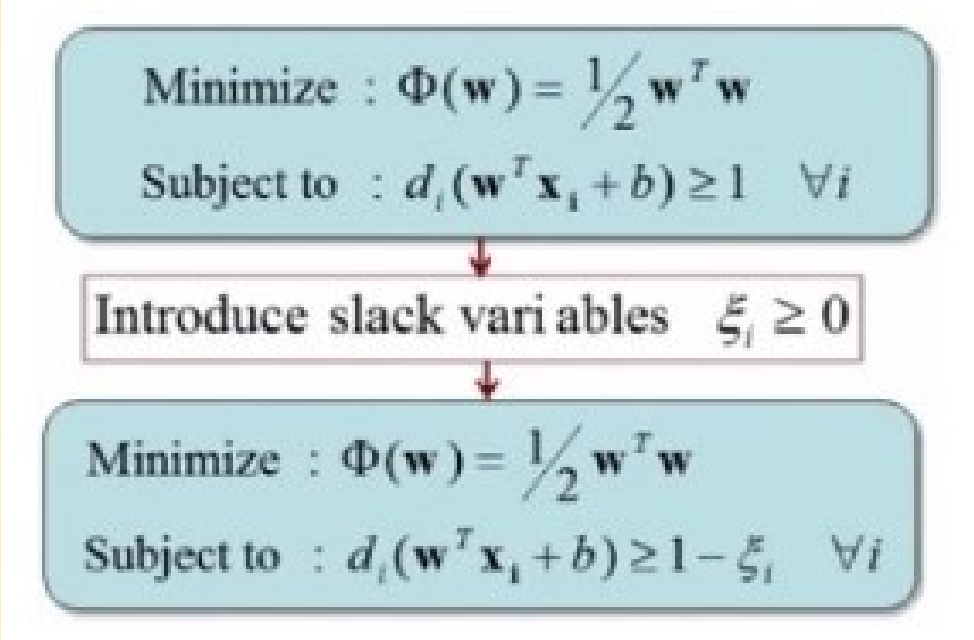
Support Vector Machines can probably help you to find a separating hyperplane but only if it exists. There are certain cases when it is not possible to define a hyperplane, this happens due to noise in the data. Another possible reason can be a non-linear boundary.

There might be cases where there is no possibility to define a hyperplane, which can happen due to noise in the data. In fact, another reason can be a non-linear boundary as well. The first graph on the left depicts noise and the next one on the right shows a non-linear boundary.





For such problems which arise due to noise in the data, the best way is to reduce the margin itself and introduce slack.



The non-linear boundary problem can be solved if we introduce a kernel. Some of the kernel functions that can be introduced are mentioned below:

Polynomial: $K_p(\mathbf{X}, \mathbf{Y}) = (1 + \mathbf{X} \bullet \mathbf{Y})^p$

Radial Basis Function (RBF) or Gaussian: $K_r(\mathbf{X}, \mathbf{Y}) = e^{-\frac{1}{2\sigma^2}\|\mathbf{X}-\mathbf{Y}\|_2^2}$

Hyperbolic Tangent: $K_s(\mathbf{X}, \mathbf{Y}) = \tanh(\beta_0 \mathbf{X} \bullet \mathbf{Y} + \beta_1)$

A radial basis function is a real-valued function whose value is dependent on the distance between the input and some fixed point. In machine learning, the radial basis function kernel, or RBF kernel, is a popular kernel function used in various kernelized learning algorithms.

The RBF kernel on two samples x and x', represented as feature vectors in some input space, is defined as:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

How to tune Parameters of SVM (in *Scikit-learn*)?

Kernel: Kernel in support vector machine is responsible for the transformation of the input data into the required format. Some of the kernels used in support vector machines are linear, polynomial and radial basis function (RBF). In order to create a non-linear hyperplane, we use RBF and Polynomial function, and for complex applications, you should use more advanced kernels to separate classes that are nonlinear in nature. With this transformation, you can obtain accurate classifiers.

Regularization: Using the *Scikit-learn's* C parameters and adjusting we can maintain regularization. C denotes a penalty parameter representing an error or any form of misclassification. This misclassification allows you to understand how much of the error is actually bearable. This helps you nullify the compensation between the misclassified term and the decision boundary. With a smaller C value, you obtain hyperplane of small margin and with a larger C value, hyperplane of larger value is obtained.

Gamma: Lower value of Gamma creates a loose fit of the training dataset. On the other hand, a high value of gamma allows the model to get fit more appropriately. A low value of gamma will only provide consideration to the nearby points for the calculation of a separate plane. However, the high value of gamma will consider all the data-points to calculate the final separation line.

You do not need to tune parameter in all cases.

Tuning Hyperparameters (The 'C' and 'gamma'):

C is the parameter for the soft margin cost function, which controls the influence of each individual support vector. This process involves trading error penalty for stability. Small C tends to emphasize the margin while ignoring the outliers in the training data(Soft Margin), while large C may tend to overfit the training data(Hard Margin). Thus for a very large values we can cause overfitting of the model and for a very small value of C we can cause underfitting. Thus the value of C must be chosen in such a manner that it generalizes the unseen data well.

The **gamma** parameter is the inverse of the standard deviation of the RBF kernel (Gaussian function), which is used as a similarity measure between two points. A small gamma value define a Gaussian function with a large variance. In this case, two points can be considered similar even if are far from each other. On the other hand, a large gamma value define a Gaussian function with a small variance and in this case, two points are considered similar just if they are close to each other.

Advantages of SVM:

1. SVM Classifiers offer good accuracy and perform faster prediction compared to Naïve Bayes algorithm.
2. SVM guarantees optimality due to the nature of Convex Optimization, the solution will always be global minimum not a local minimum.
3. SVMcan be access it conveniently, be it from Python or Matlab.
4. SVM can be used for both linearly separable as well as non-linearly separable data. Linearly separable data is the hard margin however, non-linearly separable data poses a soft margin.
5. SVM provides compliance to the semi-supervised learning models as well. It can be implemented in both labelled and unlabelled data. The only thing it requires is a condition to the minimization problem which is known as the Transductive SVM.
6. Feature Mapping used to be complex with respect to computation of the overall training performance of the model. With the help of Kernel Trick, SVM can carry out the feature mapping using simple dot product.
7. SVM works well with a clear margin of separation and with high dimensional space.

Disadvantages of SVM:

1. SVM is not at all capable of handling text structures. It leads to bad performance as it results in the loss of sequential information.
2. SVM is not suitable for large datasets because of its high training time and it also takes more time in training compared to Naïve Bayes.
3. SVM works poorly with overlapping classes and is also sensitive to the type of kernel used.
4. In cases where the number of features for each data point exceeds the number of training data samples , the SVM under performs.

Applications of SVM in Real World:

Support vector machines depend on supervised learning algorithms. The main goal of using SVM is to classify unseen data correctly. SVMs can be used to solve various real-world problems:

1. **Face detection** – SVM can be used to classify parts of the image as a face and non-face and create a square boundary around the face.
2. **Text and hypertext categorization** – SVM allows text and hypertext categorization for both inductive and transductive models. It uses training data for classification of documents into different categories. It categorizes based on the score generated and then compares with the threshold value.
3. **Classification of images** – SVMs enhances search accuracy for image classification. In comparison to the traditional query-based searching techniques, SVM provides better accuracy.

contd…

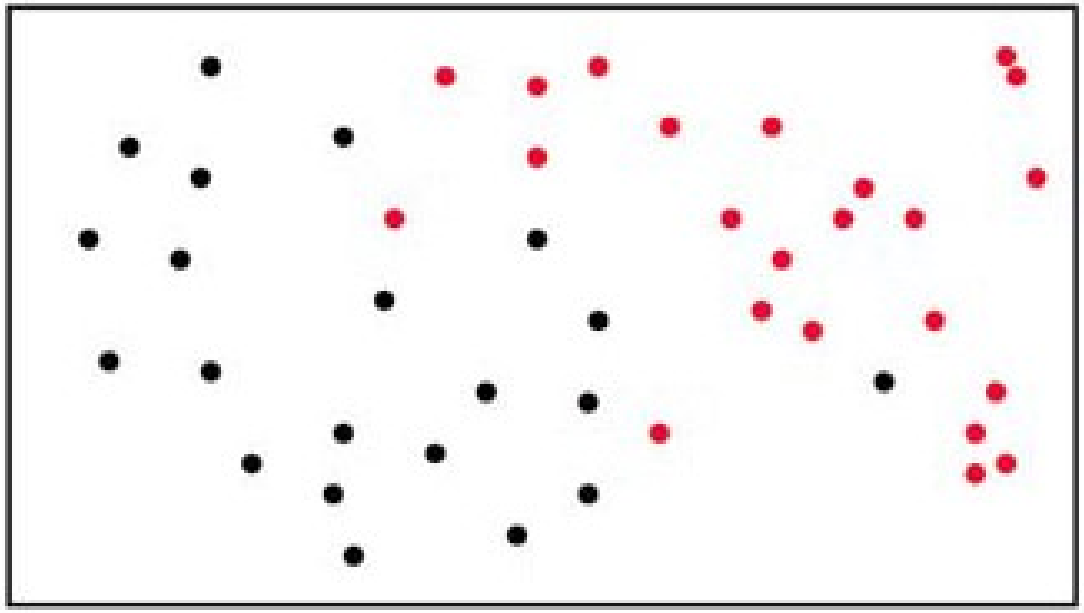
Applications of SVM in Real World…contd.:

4. **Bioinformatics** – It includes classification of proteins and classification of cancer. SVM is used for identifying the classification of genes, patients on the basis of genes and other biological problems.
5. **Protein fold and remote homology detection** – SVM algorithms are applied for protein remote homology detection.
6. **Handwriting recognition** – SVMs are used widely to recognize handwritten characters.
7. **Generalized predictive control (GPC)** – You can use SVM based GPC in order to control chaotic dynamics with useful parameters.

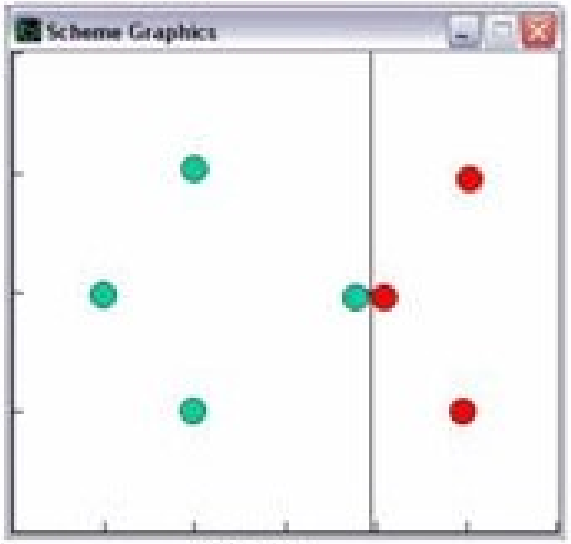
Kernel Function

Not Linearly Separable?

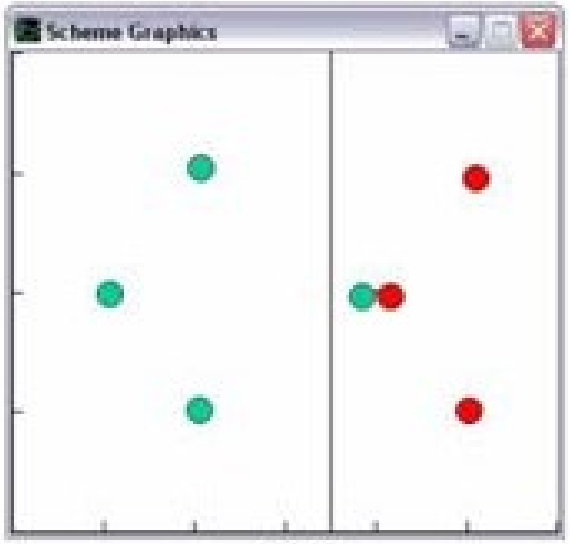
- Require $0 \leq \alpha_i \leq C$
- C specified by user; controls tradeoff between size of margin and classification errors
- $C = \infty$ for separable case



C Change



C=10



C=1

Example: Linearly Separable

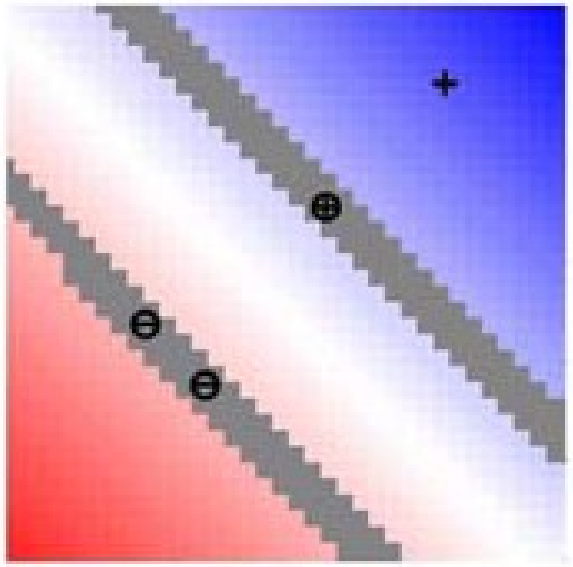


Image by Patrick Winston

Another example: Not linearly separable

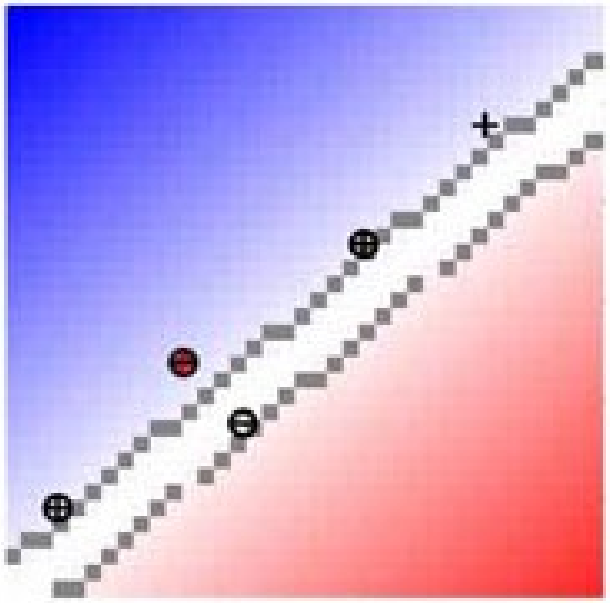
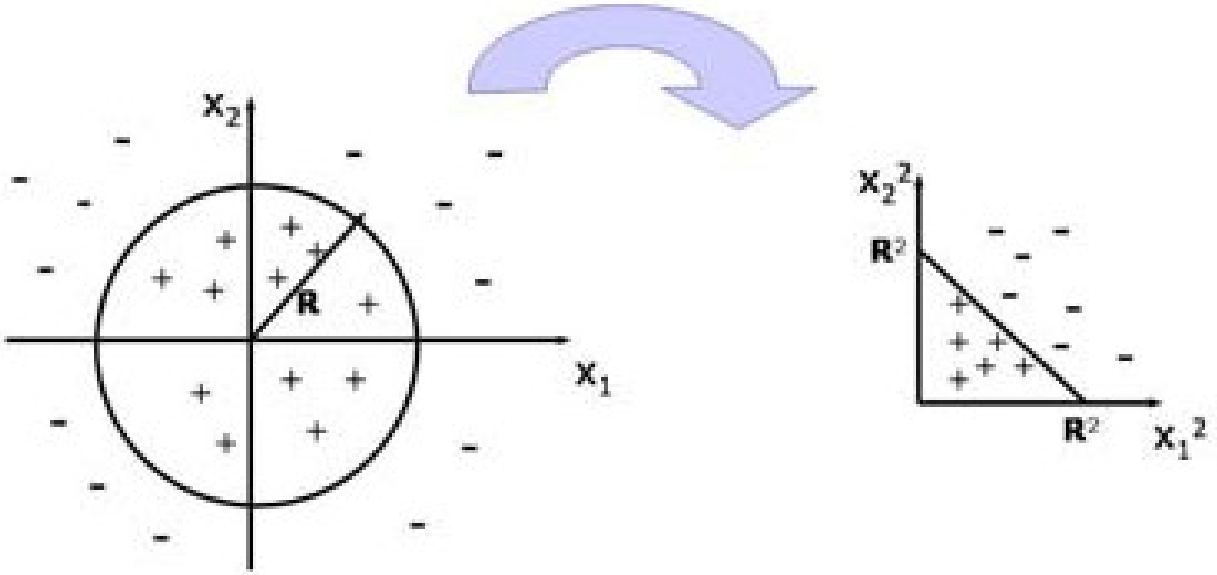


Image by Patrick Winston

Isn't a linear classifier very limiting?

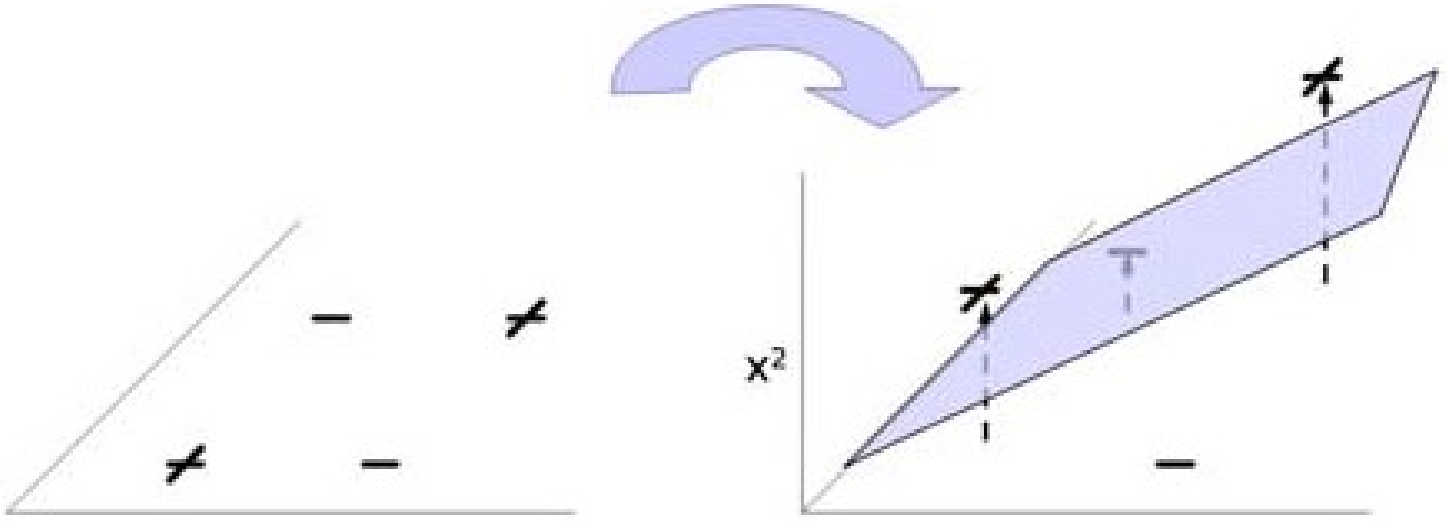


not linearly separable

linearly separable using squared value of features.

Important: Linear separator in transformed feature space maps into non-linear separator in original feature space

Not separable?
Try a higher dimensional space!



Not separable with 2D line

Separable with 3D plane

What you need

- To get into the new feature space, you use $\Phi(\mathbf{x}^i)$
- The transformation can be to a higher-dimensional feature space and may be non-linear in the feature values.

What you need

- To get into the new feature space, you use $\Phi(\mathbf{x}^i)$
- The transformation can be to a higher-dimensional feature space and may be non-linear in the feature values.
- Recall that SVM's only use dot products of the data, so
- To optimize classifier, you need $\Phi(\mathbf{x}^i) \cdot \Phi(\mathbf{x}^k)$
- To run classifier, you need $\Phi(\mathbf{x}^i) \cdot \Phi(\mathbf{u})$
- So, all you need is a way to compute dot products in transformed space as a function of vectors in original space!

The "Kernel Trick"

- If dot products can be efficiently computed by
  $\Phi(\mathbf{x}^i) \cdot \Phi(\mathbf{x}^k) = K(\mathbf{x}^i, \mathbf{x}^k)$
- Then, all you need is a function on low-dim inputs
  $K(\mathbf{x}^i, \mathbf{x}^k)$
- You don't need ever to construct high-dimensional
  $\Phi(\mathbf{x}^i)$

Standard Choices For Kernels

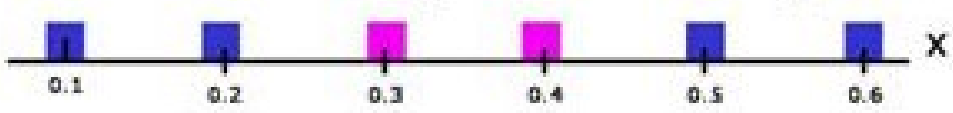
- No change (linear kernel)

$$\Phi(\mathbf{x}^i) \cdot \Phi(\mathbf{x}^k) = K(\mathbf{x}^i, \mathbf{x}^k) = \mathbf{x}^i \cdot \mathbf{x}^k$$

What you need

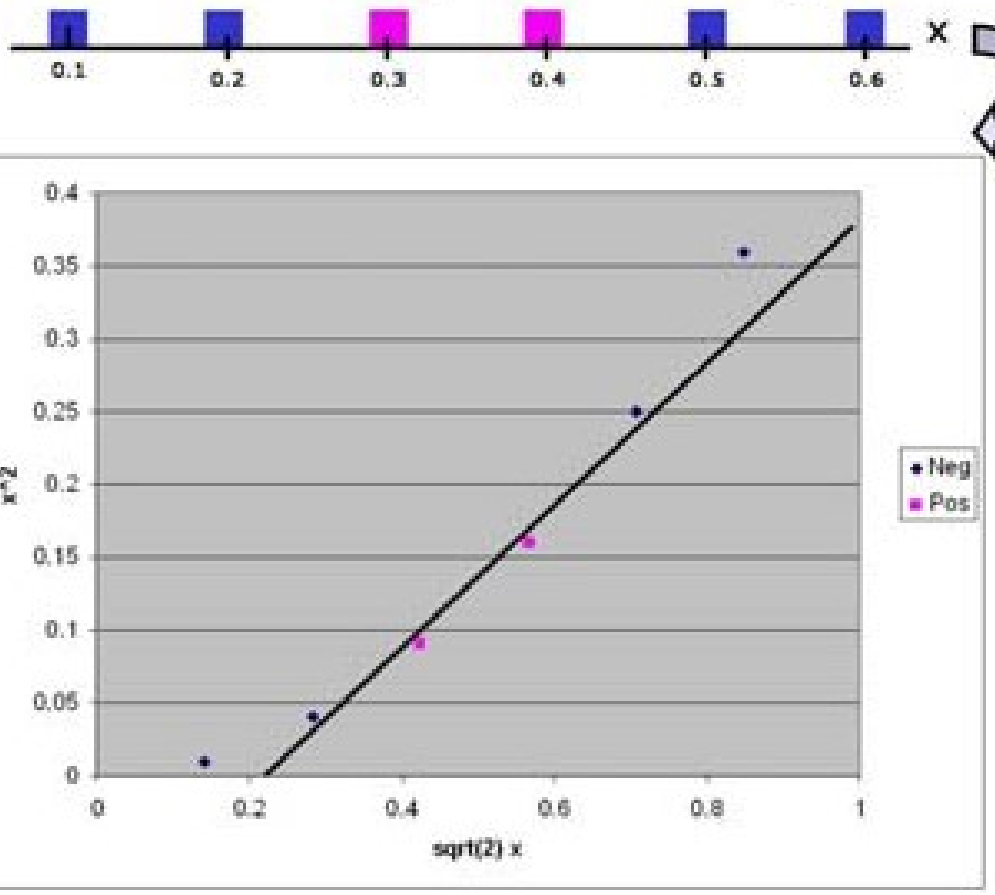
- To get into the new feature space, you use $\Phi(\mathbf{x}^i)$
- The transformation can be to a higher-dimensional feature space and may be non-linear in the feature values.
- Recall that SVM's only use dot products of the data, so
- To optimize classifier, you need $\Phi(\mathbf{x}^i) \cdot \Phi(\mathbf{x}^k)$
- To run classifier, you need $\Phi(\mathbf{x}^i) \cdot \Phi(\mathbf{u})$
- So, all you need is a way to compute dot products in transformed space as a function of vectors in original space!

Polynomial Kernel Example
(one feature)



Not separable

Polynomial Kernel Example
(one feature)



Not separable

$\Phi(x) = (x^2, \sqrt{2}x, 1)$

Separable

$$\Phi(x) \cdot \Phi(z) = x^2 z^2 + 2xz + 1 = (1 + xz)^2$$

Polynomial Kernel

- Polynomial kernel for n=2 and features $\mathbf{x}=[x_1\ x_2]$

$$K(\mathbf{x},\mathbf{z}) = (1+\mathbf{x}\cdot\mathbf{z})^2$$

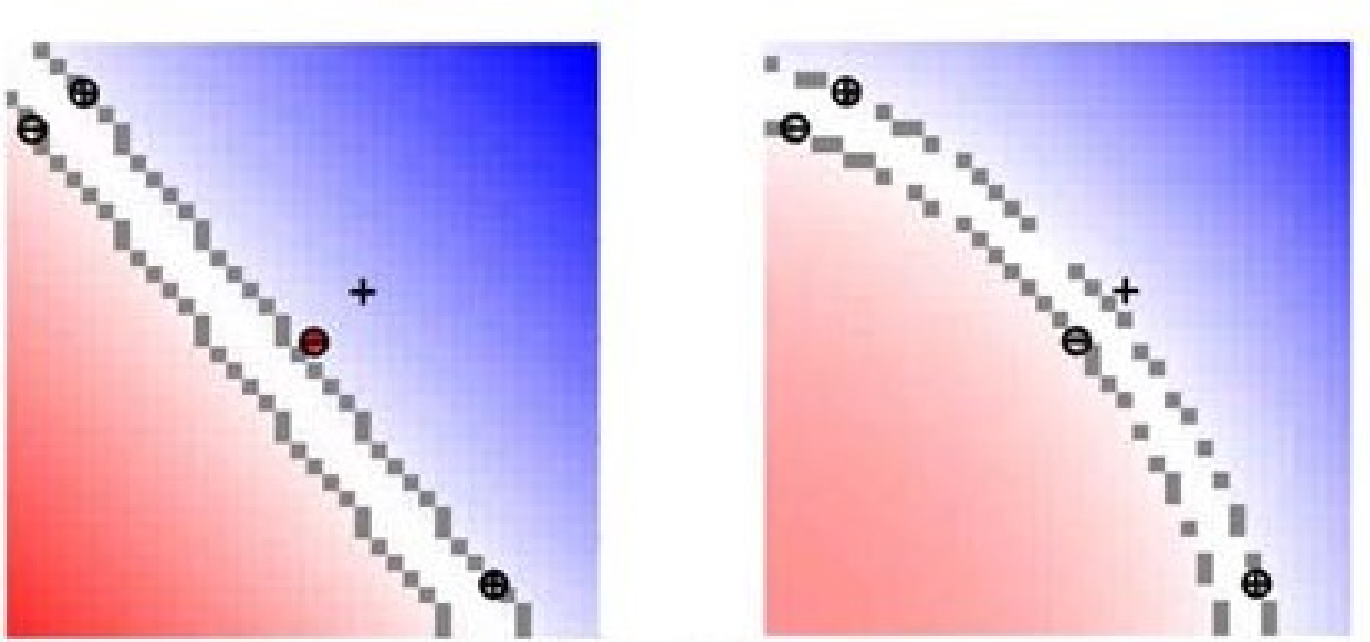
is equivalent to the following feature mapping:

$$\Phi(\mathbf{x}) = [x_1^2\ x_2^2\ \sqrt{2}x_1x_2\ \sqrt{2}x_1\ \sqrt{2}x_2\ 1]$$

- We can verify that:

$$\begin{aligned}\Phi(\mathbf{x})\cdot\Phi(\mathbf{z}) &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 + 2x_1z_1 + 2x_2z_2 + 1\\ &= (1+x_1z_1+x_2z_2)^2\\ &= (1+\mathbf{x}\cdot\mathbf{z})^2\\ &= K(\mathbf{x},\mathbf{z})\end{aligned}$$

Polynomial Kernel



Images by Patrick Winston

Standard Choices For Kernels

- No change (linear kernel)

$$\Phi(\mathbf{x}^i) \cdot \Phi(\mathbf{x}^k) = K(\mathbf{x}^i, \mathbf{x}^k) = \mathbf{x}^i \cdot \mathbf{x}^k$$

- Polynomial kernel (n^{th} order)

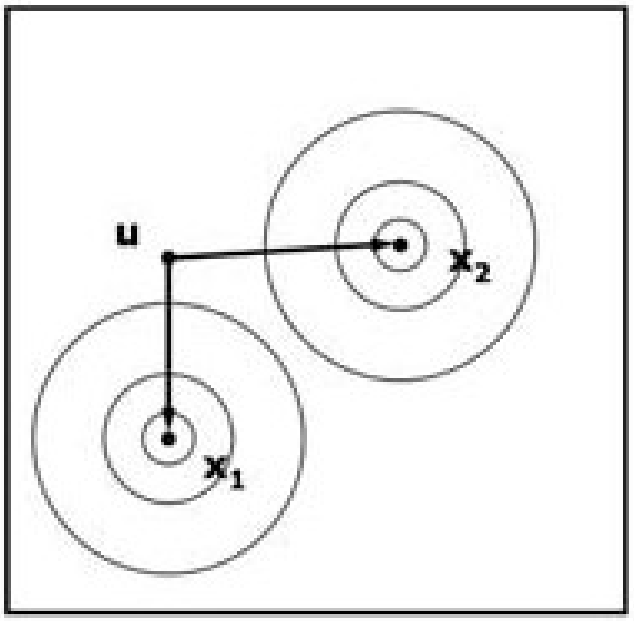
$$K(\mathbf{x}^i, \mathbf{x}^k) = (1 + \mathbf{x}^i \cdot \mathbf{x}^k)^n$$

- Radial basis kernel (σ is standard deviation)

$$K(\mathbf{x}^i, \mathbf{x}^k) = e^{\frac{-\left|\mathbf{x}^i - \mathbf{x}^k\right|^2}{2\sigma^2}} = e^{\frac{-(\mathbf{x}^i - \mathbf{x}^k)\cdot(\mathbf{x}^i - \mathbf{x}^k)}{2\sigma^2}}$$

Radial-basis kernel

- Classifier based on sum of Gaussian bumps with standard deviation σ, centered on support vectors.



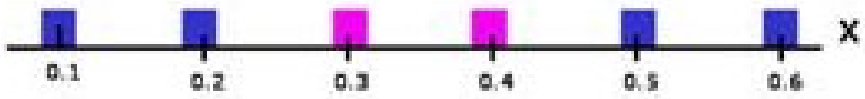
$$h(\mathbf{u}) = sign[h'(\mathbf{u})]$$

$$h'(\mathbf{u}) = \sum_{i=1}^{k} \alpha_i y^i K(\mathbf{x}^i, \mathbf{u}) + b$$

$$K(\mathbf{x}^i, \mathbf{u}) = e^{\frac{-\|\mathbf{x}^i - \mathbf{u}\|^2}{2\sigma^2}}$$

Radial-basis kernel

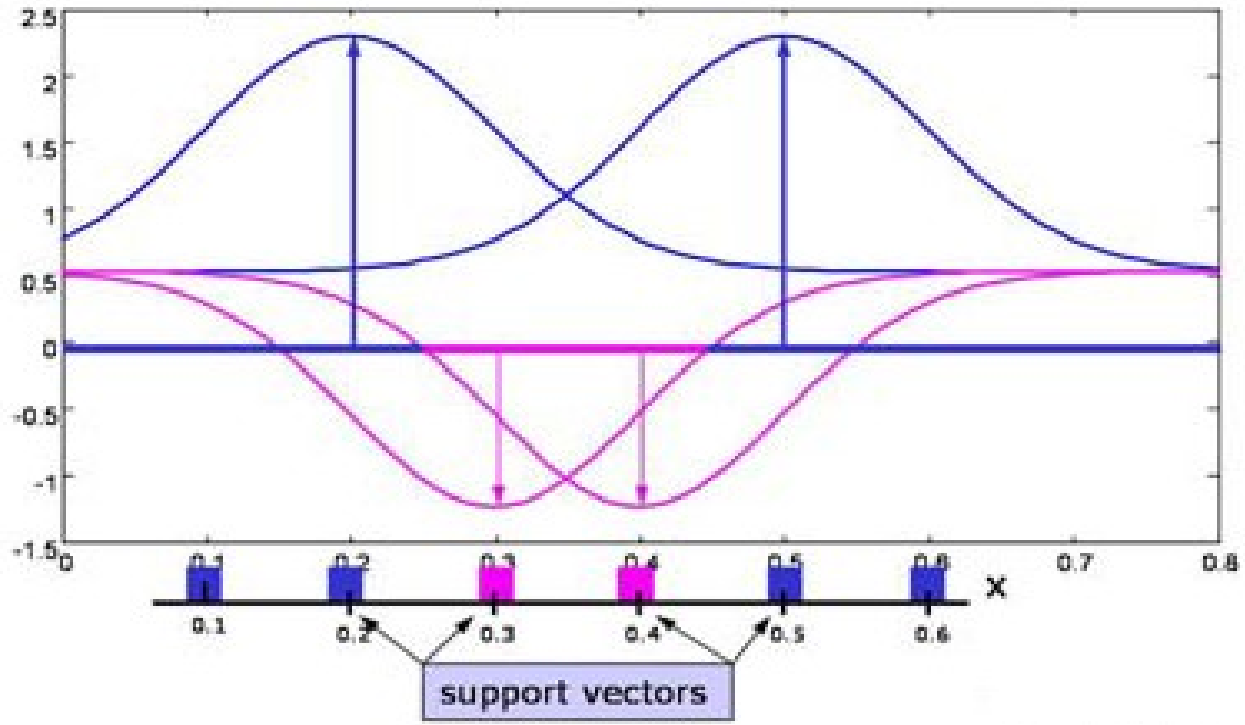
$\sigma = 0.1$



Radial-basis kernel

$\alpha_1 = 1.76$ $\alpha_2 = -1.76$ $b = 0.525$ $\sigma = 0.1$

$\alpha_3 = 1.76$ $\alpha_4 = -1.76$



Radial-basis kernel

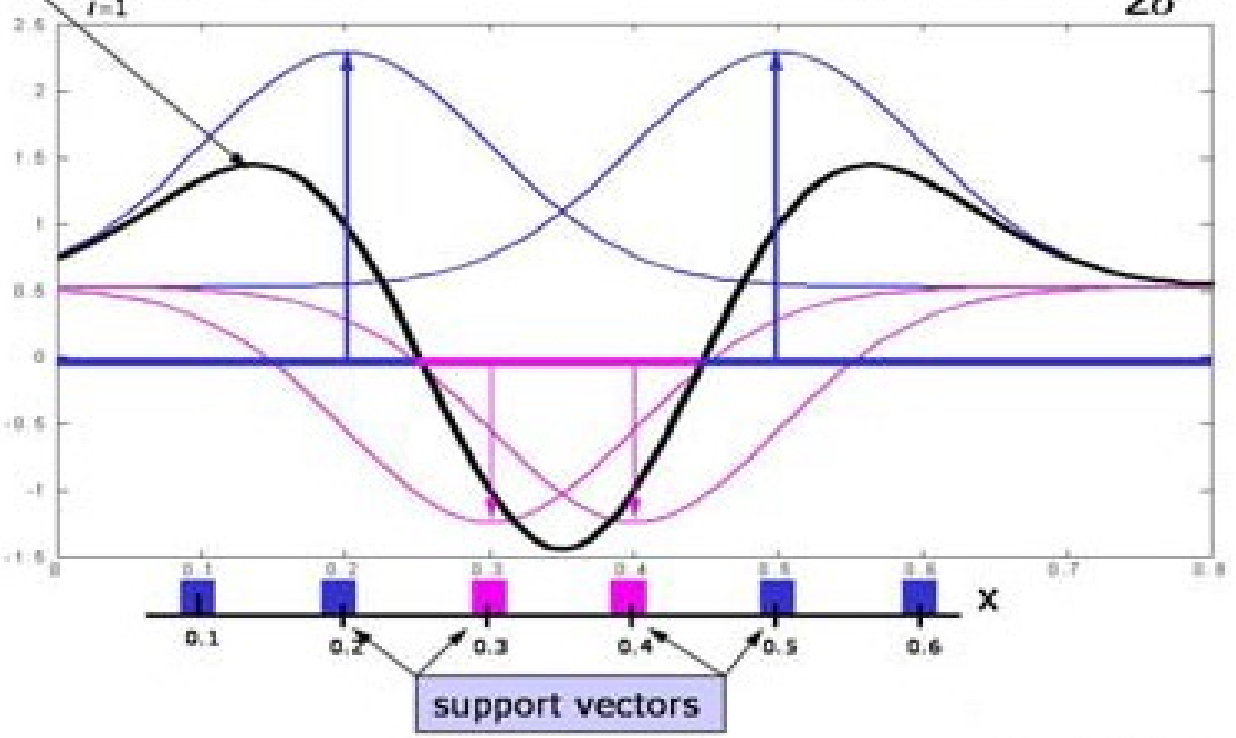
$\alpha_1 = 1.76 \quad \alpha_2 = -1.76$
$\alpha_3 = 1.76 \quad \alpha_4 = -1.76$

$b = 0.525$

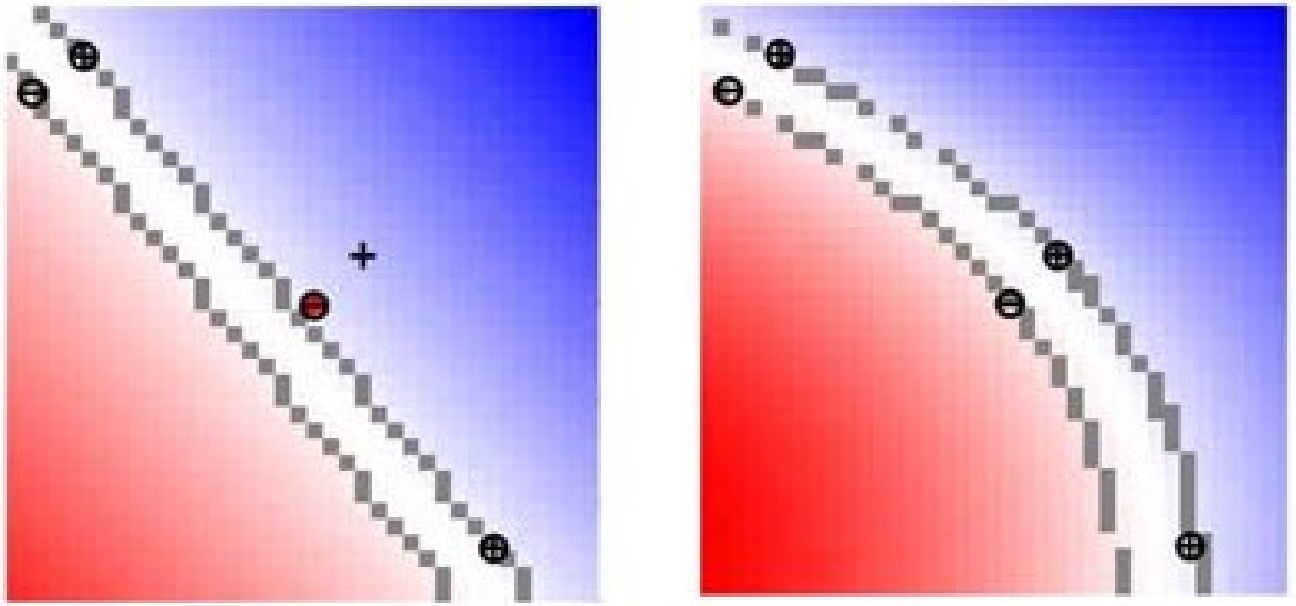
$\sigma = 0.1$

$$h'(\mathbf{u}) = \sum_{i=1}^{4} \alpha_i y^i K(\mathbf{x}^i, \mathbf{u}) + b$$

$$K(\mathbf{x}^i, \mathbf{u}) = e^{\frac{-|\mathbf{x}^i - \mathbf{u}|^2}{2\sigma^2}}$$



Radial-basis kernel (large σ)



Images by Patrick Winston

Another radial-basis example (small σ)

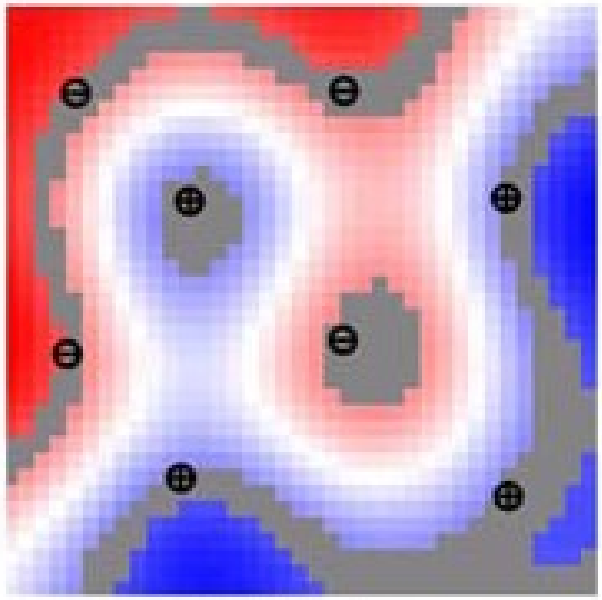


Image by Patrick Winston

Cross-Validation Error

- Does mapping to a very high-dimensional space lead to over-fitting?
- Generally, no, thanks to the fact that only the support vectors determine the decision surface.

Cross-Validation Error

- Does mapping to a very high-dimensional space lead to over-fitting?
- Generally, no, thanks to the fact that only the support vectors determine the decision surface.
- The expected leave-one-out cross-validation error depends on number of support vectors, not dimensionality of feature space.

$$\text{Expected CV error} \leq \frac{\text{Expected \# support vectors}}{\text{\# training samples}}$$

- If most data points are support vectors, a sign of possible overfitting, independent of the dimensionality of feature space.

Summary

- A single global optimum
  - Quadratic programming or gradient descent

Summary

- A single global maximum
  - Quadratic programming or gradient descent
- Fewer parameters
  - C and kernel parameters (n for polynomial, σ for radial basis kernel)

Summary

- A single global maximum
  - Quadratic programming or gradient descent
- Fewer parameters
  - C and kernel parameters (n for polynomial, σ for radial basis kernel)
- Kernel
  - Quadratic minimization depends only on dot products of sample vectors
  - Recognition depends only on dot products of unknown vector with sample vectors
  - Reliance on only dot products enables efficient feature mapping to higher-dimensional spaces where linear separation is more effective.

Real Data

- Wisconsin Breast Cancer Data
  - 9 features
  - C=1
  - 37 support vectors are used from 512 training data points
  - 12 prediction errors on training set (98% accuracy)
  - 96% accuracy on 171 held out points
  - Essentially same performance as nearest neighbors and decision trees
- Don't expect such good performance on every data set.

Success Stories

- Gene microarray data
  - outperformed all other classifiers
  - specially designed kernel
- Text categorization
  - linear kernel in >10,000 D input space
  - best prediction performance
  - 35 times faster to train than next best classifier (decision trees)
- Many others:
  http://www.clopinet.com/isabelle/Projects/SVM/applist.html

Feature Selection and Clustering

Feature Selection

- In many machine learning applications, there are huge numbers of features
  - text classification (# words)
  - gene arrays (5,000 – 50,000)
  - images (512 x 512 pixels)

Feature Selection

- In many machine learning applications, there are huge numbers of features
  - text classification (# words)
  - gene arrays (5,000 – 50,000)
  - images (512 x 512 pixels)
- Too many features
  - make algorithms run slowly
  - risk overfitting

Feature Selection

- In many machine learning applications, there are huge numbers of features
  - text classification (# words)
  - gene arrays (5,000 – 50,000)
  - images (512 x 512 pixels)
- Too many features
  - make algorithms run slowly
  - risk overfitting
- Find a smaller feature space
  - subset of existing features
  - new features constructed from old ones

Feature Ranking

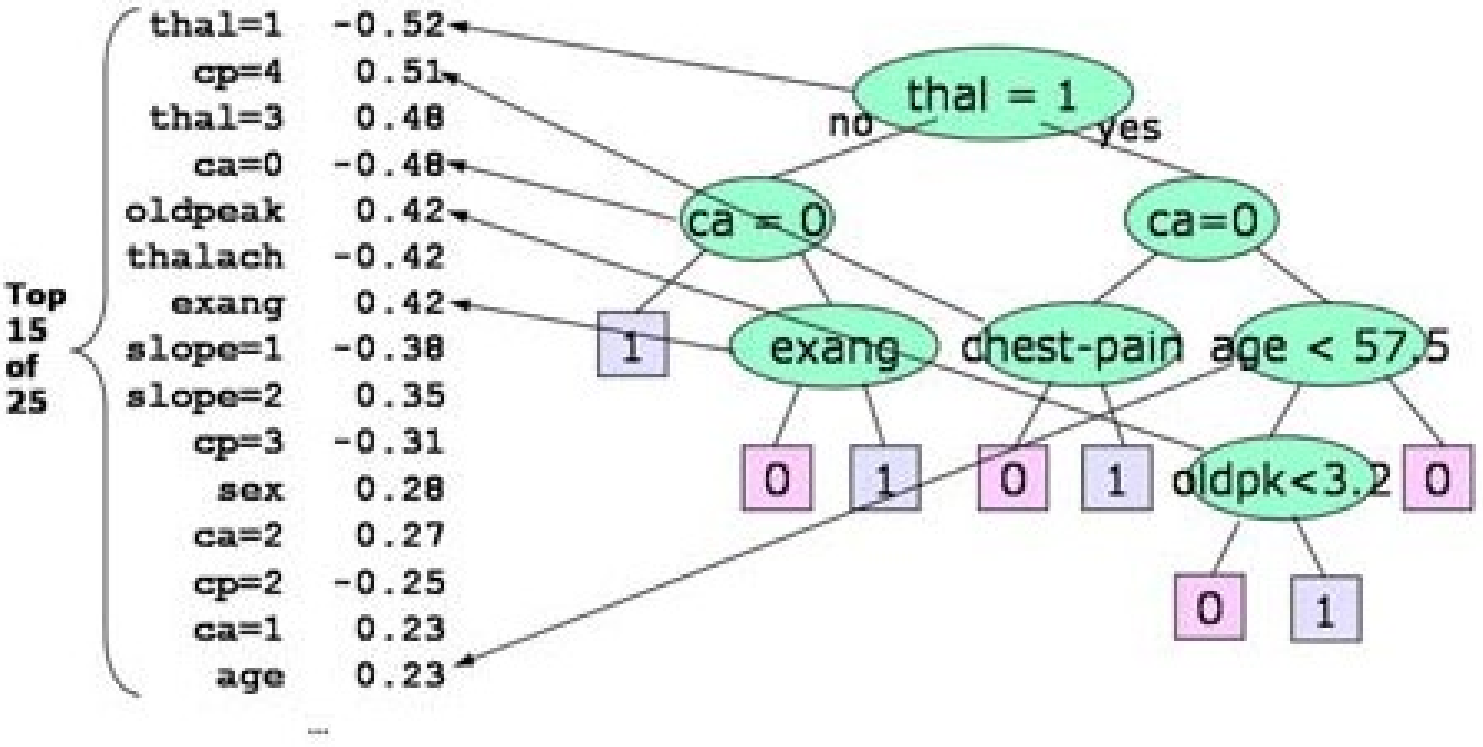
- For each feature, compute a measure of its relevance to the output
- Choose the k features with the highest rankings
- Correlation between feature j and output

$$R(j) = \frac{\sum_i (x_j^i - \bar{x}_j)(y^i - \bar{y})}{\sqrt{\sum_i (x_j^i - \bar{x}_j)^2 \sum_i (y^i - \bar{y})^2}}$$

$$\bar{x}_j = \frac{1}{n}\sum_i x_j^i \qquad \bar{y} = \frac{1}{n}\sum_i y^i$$

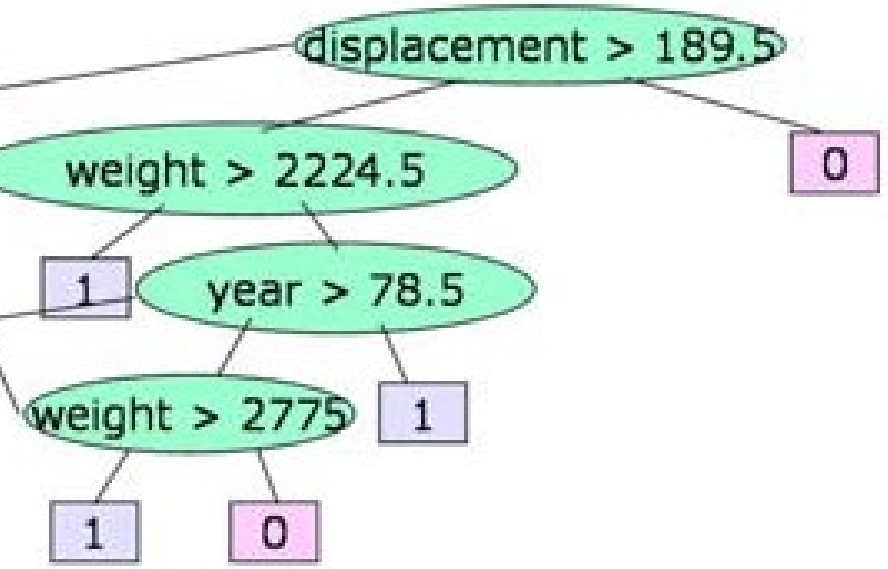
- Correlation measures how much x tends to deviate from its mean on the same examples on which y deviates from its mean

Correlations in Heart Data



Correlations in MPG > 22 data

cyl=4	0.82
displacement	-0.77
weight	-0.77
horsepower	-0.67
cyl=8	-0.58
origin=1	-0.54
model-year	0.44
origin=3	0.40
cyl=6	-0.37
acceleration	0.35
origin=2	0.26



XOR Bites Back

- As usual, functions with XOR in them will cause us trouble
  - Each feature will, individually, have a correlation of 0 (it occurs positively as much as negatively for positive outputs)
  - To solve XOR, we need to look at groups of features together

Subset Selection

- Consider subsets of variables
  - too hard to consider all possible subsets
  - wrapper methods: use training set or cross-validation error to measure the goodness of using different feature subsets with your classifier
  - greedily construct a good subset by adding or subtracting features one by one

Forward Selection

Given a particular classifier you want to use

```
F = {}
    For each f_j
        Train classifier with inputs F + {f_j}
    Add f_j that results in lowest-error classifier
      to F
Continue until F is the right size, or error has
  quit decreasing
```

Forward Selection

Given a particular classifier you want to use

```
F = {}
    For each f_j
        Train classifier with inputs F + {f_j}
    Add f_j that results in lowest-error classifier
      to F
  Continue until F is the right size, or error has
    quit decreasing
```

- Decision trees, by themselves, do something similar to this

Forward Selection

Given a particular classifier you want to use

```
F = {}
    For each f_j
        Train classifier with inputs F + {f_j}
    Add f_j that results in lowest-error classifier
      to F
  Continue until F is the right size, or error has
    quit decreasing
```

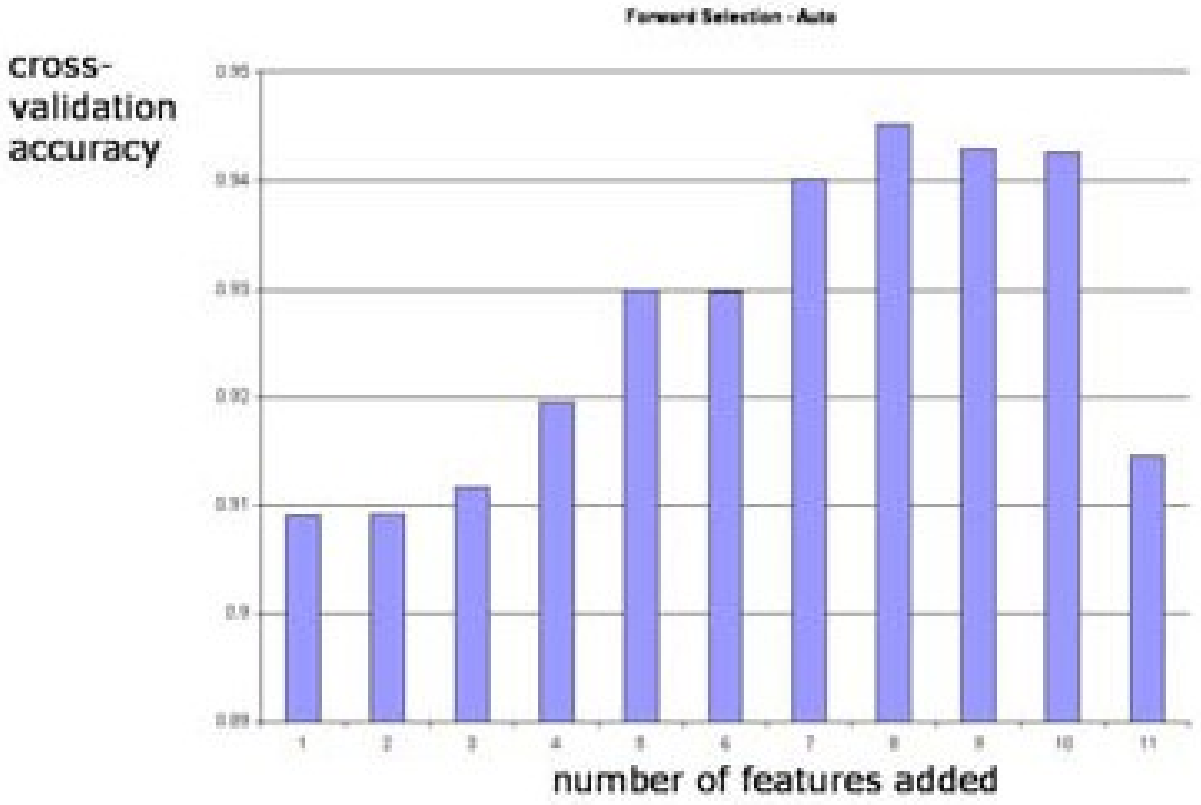
- Decision trees, by themselves, do something similar to this
- Trouble with XOR

Backward Elimination

Given a particular classifier you want to use

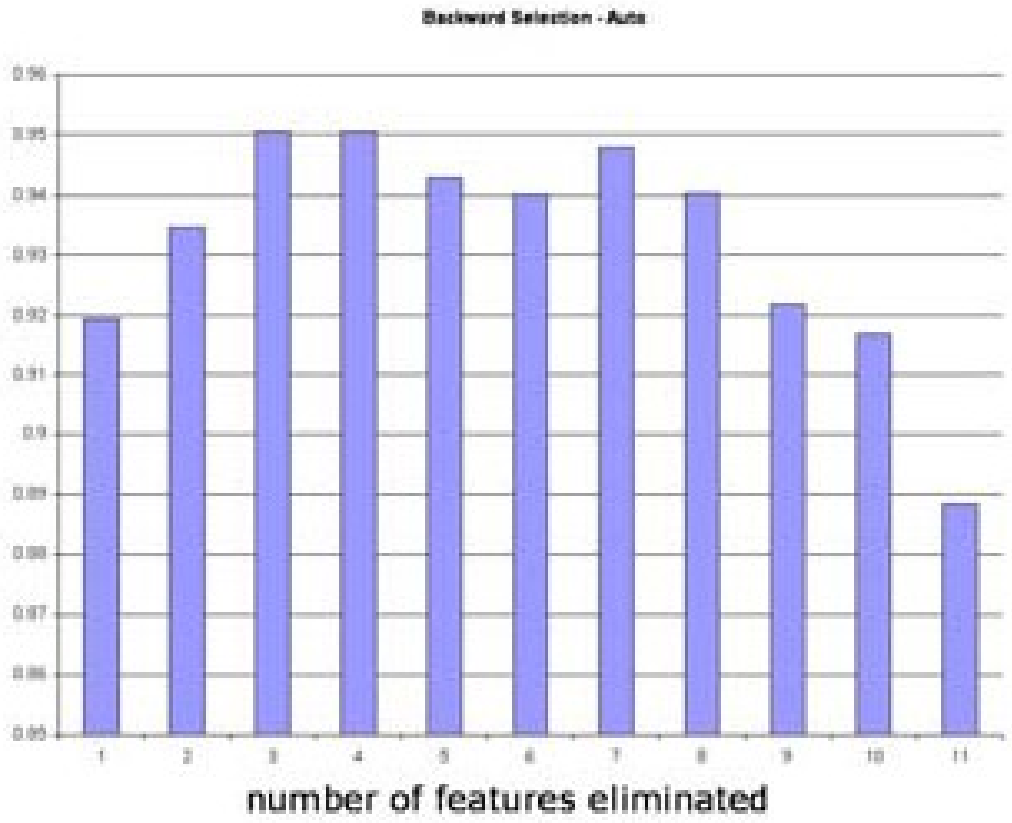
```
F = all features
    For each f_j
        Train classifier with inputs F - {f_j}
    Remove f_j that results in lowest-error
      classifier from F
  Continue until F is the right size, or error
    increases too much
```

Forward Selection on Auto Data

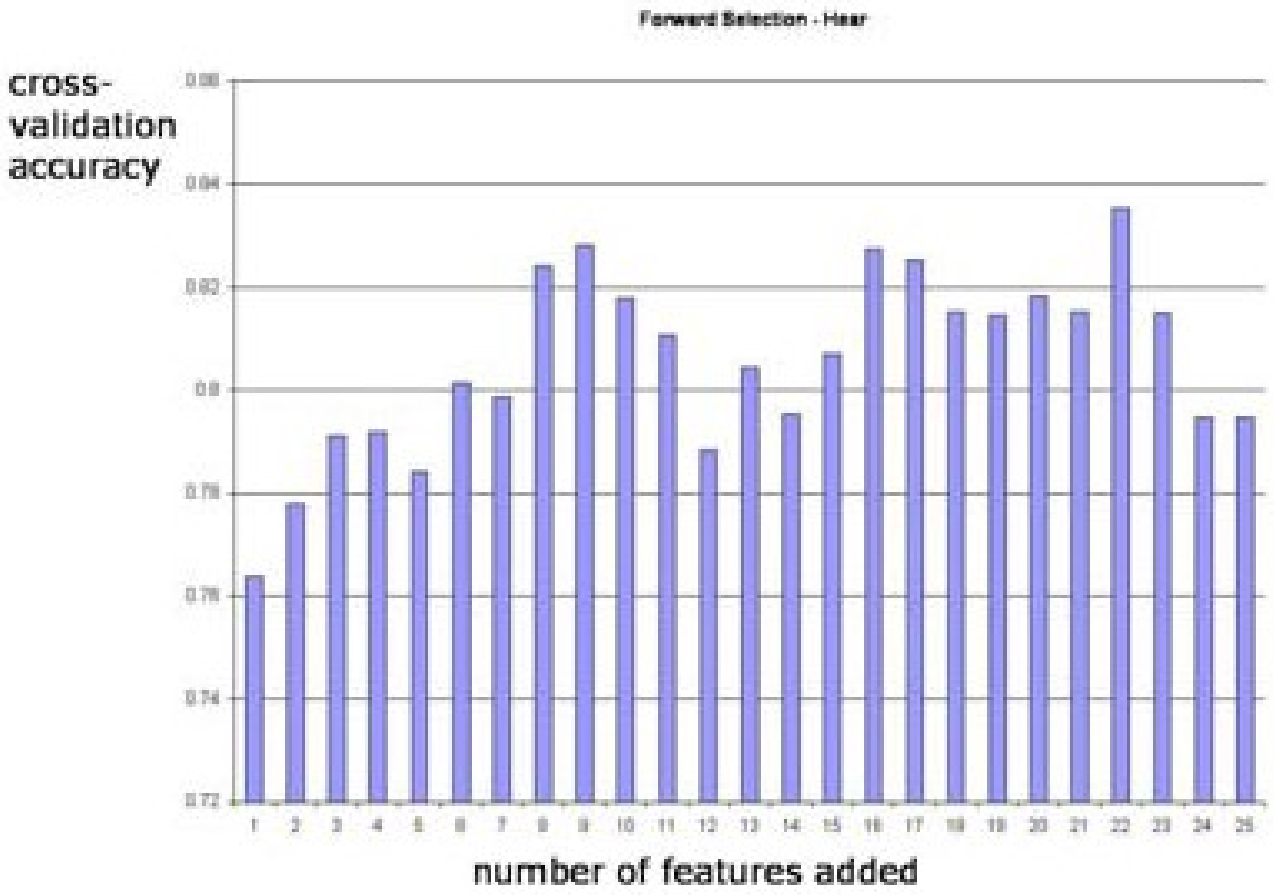


Backward Elimination on Auto Data

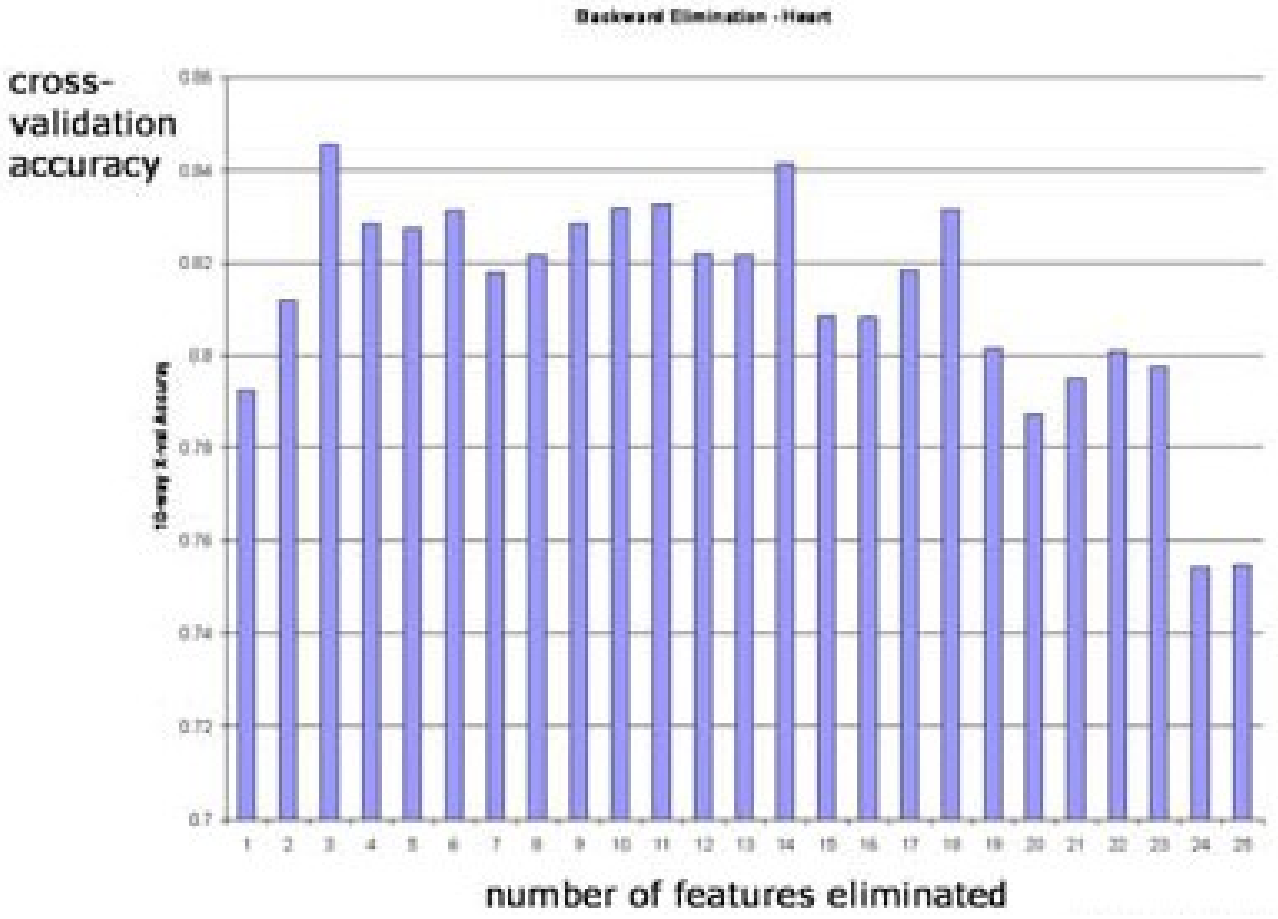
cross-validation accuracy



Forward Selection on Heart Data



Backward Elimination on Heart Data



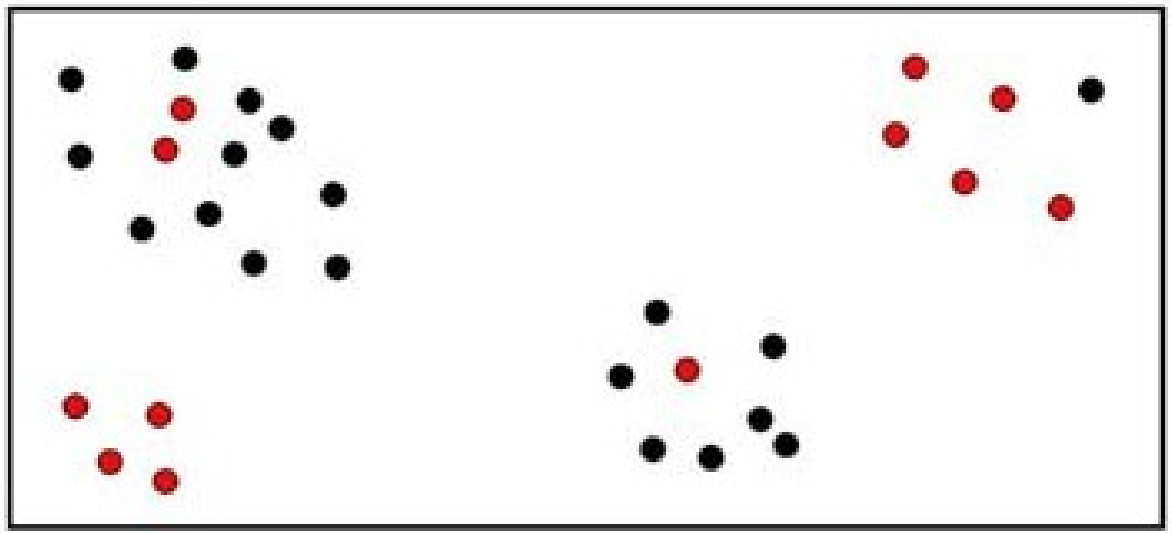
Recursive Feature Elimination

```
Train a linear SVM or neural network
Remove the feature with the smallest weight
Repeat
```

- More efficient than regular backward elimination
- Requires only one training phase per feature

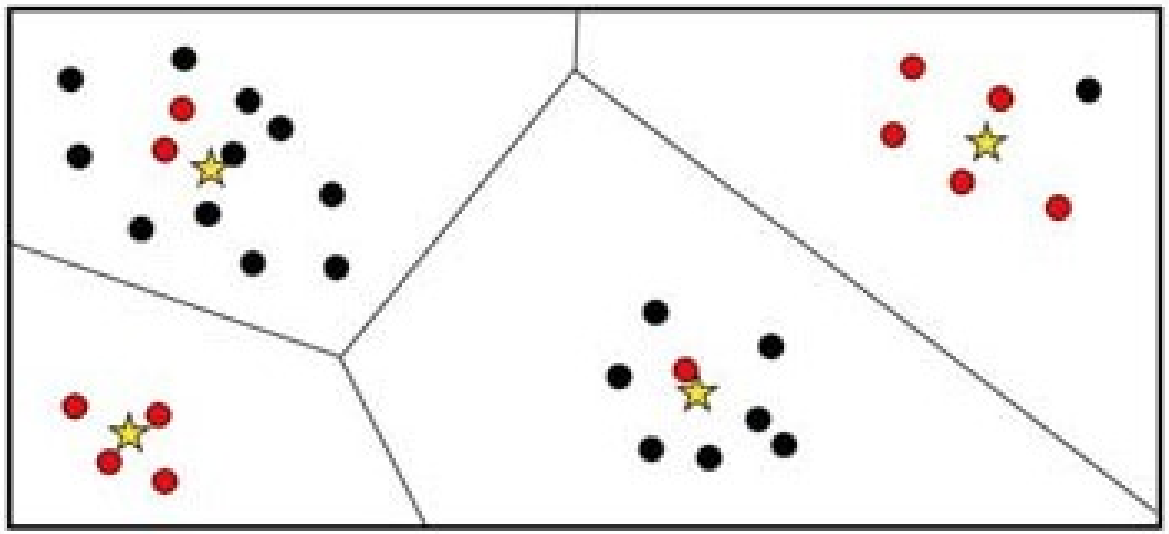
Clustering

- Form clusters of inputs
- Map the clusters into outputs
- Given a new example, find its cluster, and generate the associated output



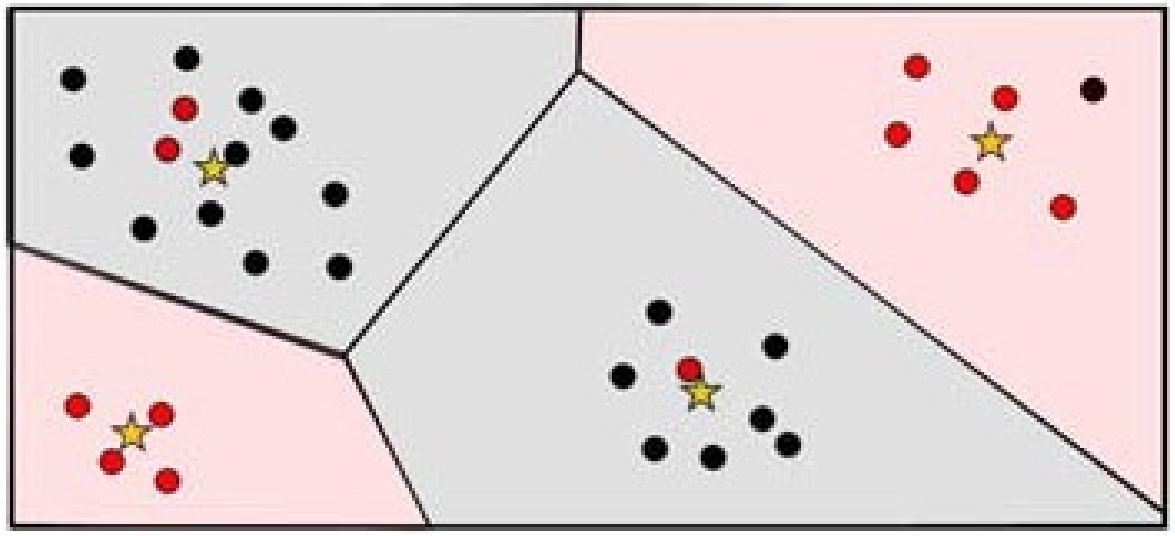
Clustering

- Form clusters of inputs
- Map the clusters into outputs
- Given a new example, find its cluster, and generate the associated output



Clustering

- Form clusters of inputs
- Map the clusters into outputs
- Given a new example, find its cluster, and generate the associated output



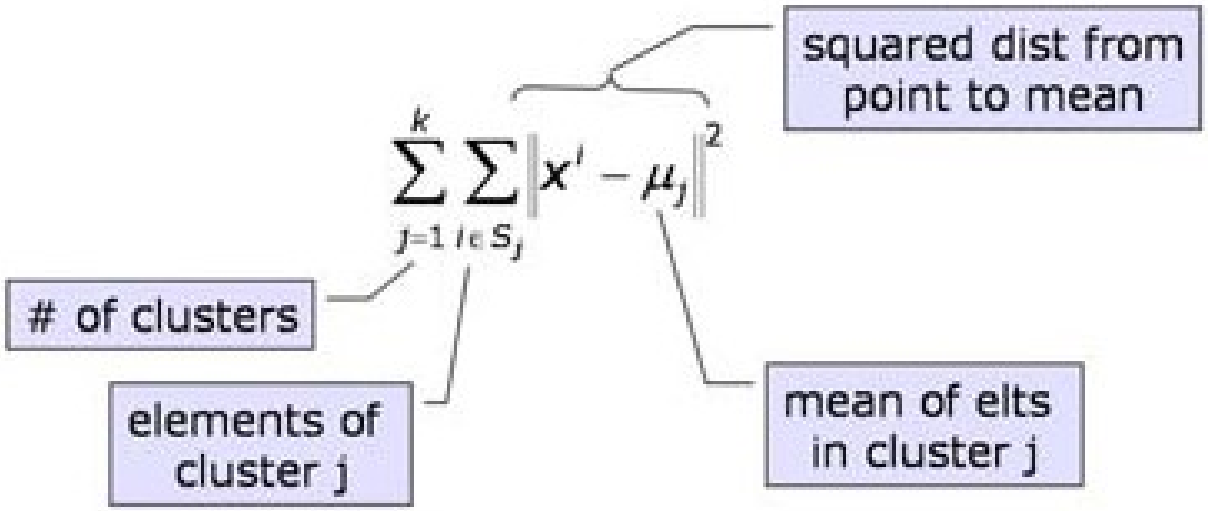
Clustering Criteria

- small distances between points within a cluster
- large distances between clusters

- Need a distance measure, as in nearest neighbor

K-Means Clustering

- Tries to minimize



- Only gets, greedily, to a local optimum

K-means Algorithm

Choose k

Randomly choose k points Cj to be cluster centers

K-means Algorithm

Choose k

Randomly choose k points Cj to be cluster centers

Loop

Partition the data into k classes Sj according to which of the Cj they're closest to

For each Sj, compute the mean of its elements and let that be the new cluster center

K-means Algorithm

```
Choose k
Randomly choose k points Cj to be cluster centers
Loop
    Partition the data into k classes Sj according
      to which of the Cj they're closest to
    For each Sj, compute the mean of its elements
      and let that be the new cluster center
Stop when centers quit moving
```

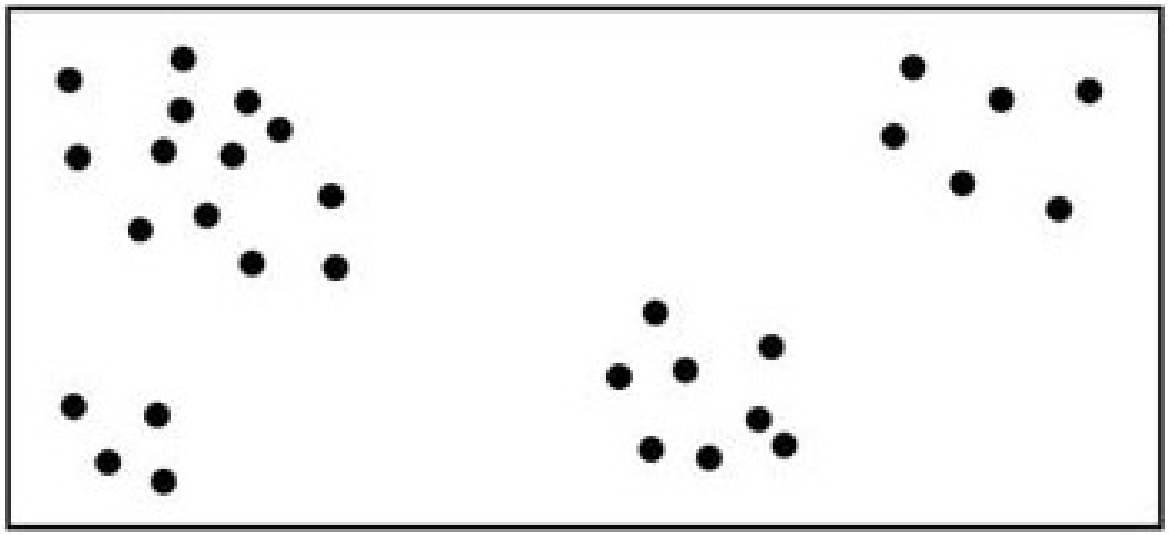
- Guaranteed to terminate

K-means Algorithm

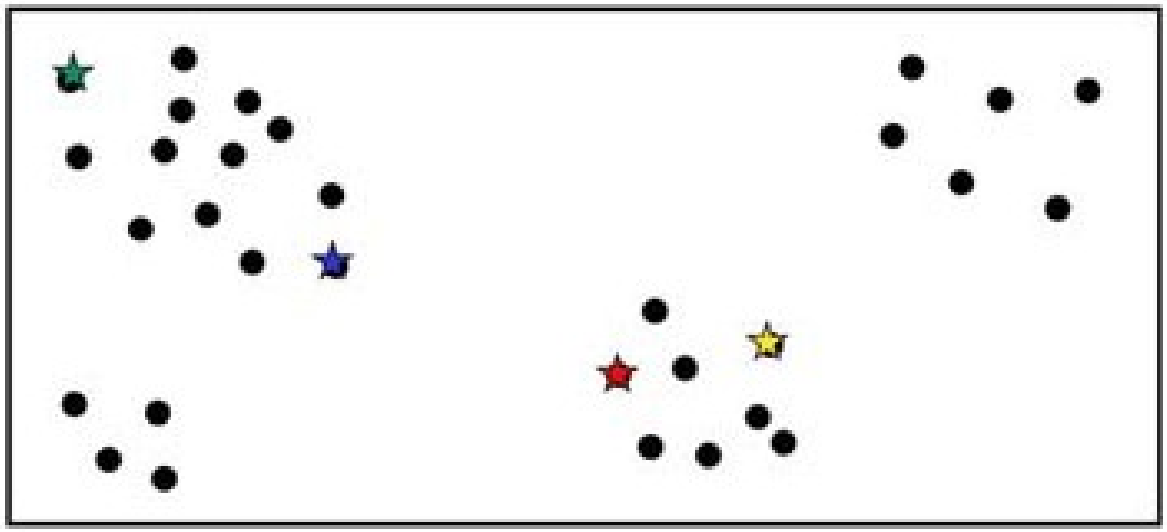
```
Choose k
Randomly choose k points Cj to be cluster centers
Loop
    Partition the data into k classes Sj according
      to which of the Cj they're closest to
    For each Sj, compute the mean of its elements
      and let that be the new cluster center
Stop when centers quit moving
```

- Guaranteed to terminate
- If a cluster becomes empty, re-initialize the center

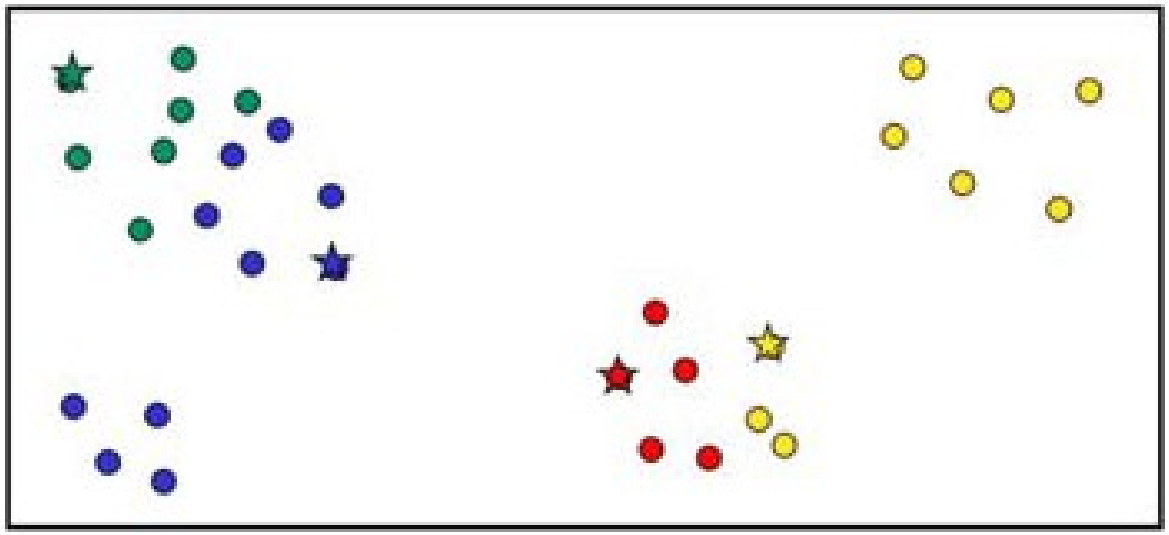
K-Means Example



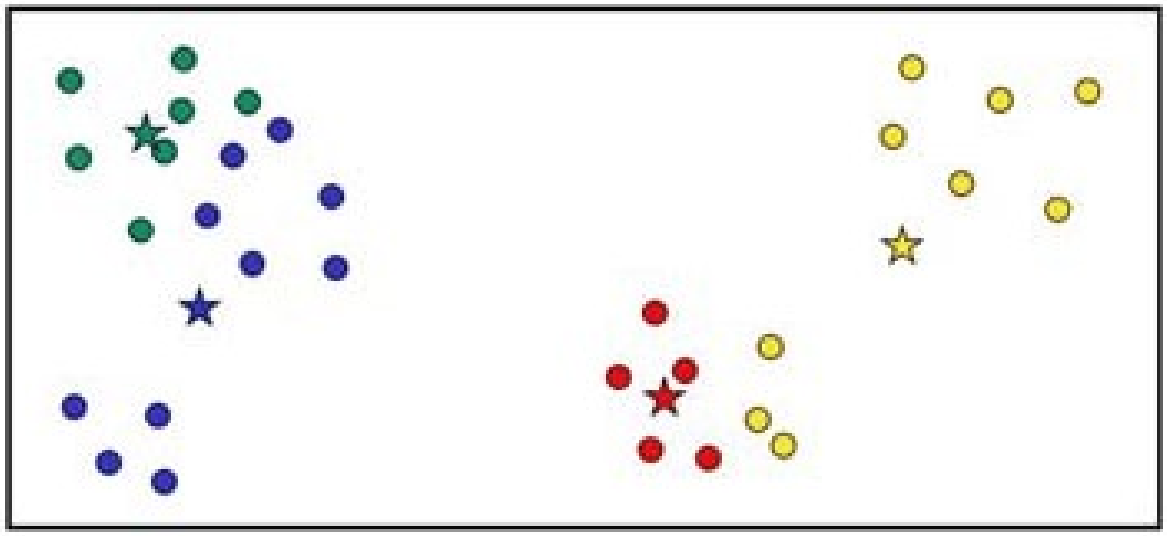
K-Means Example



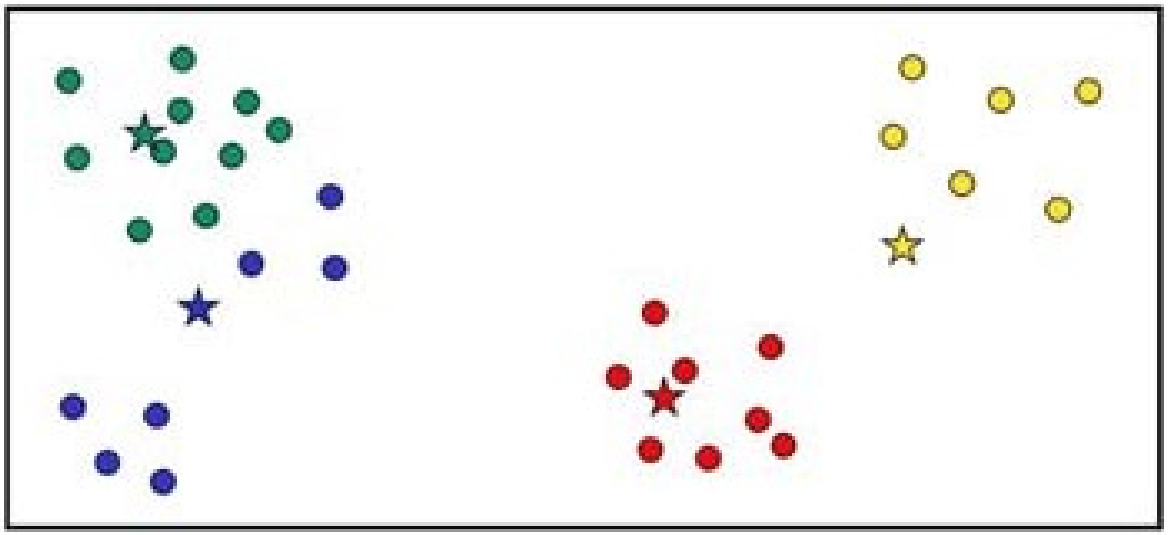
K-Means Example



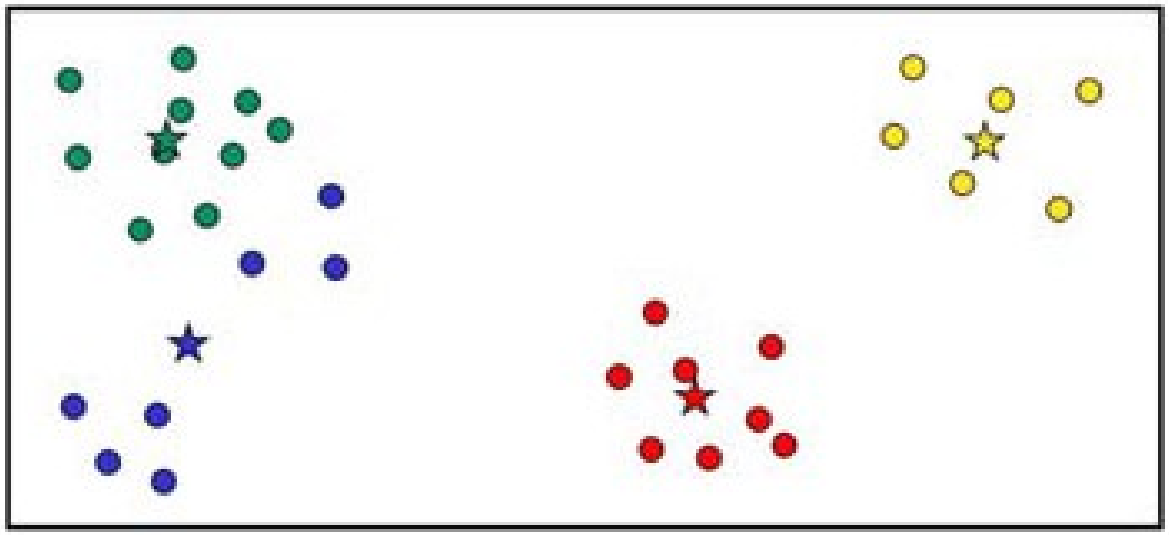
K-Means Example



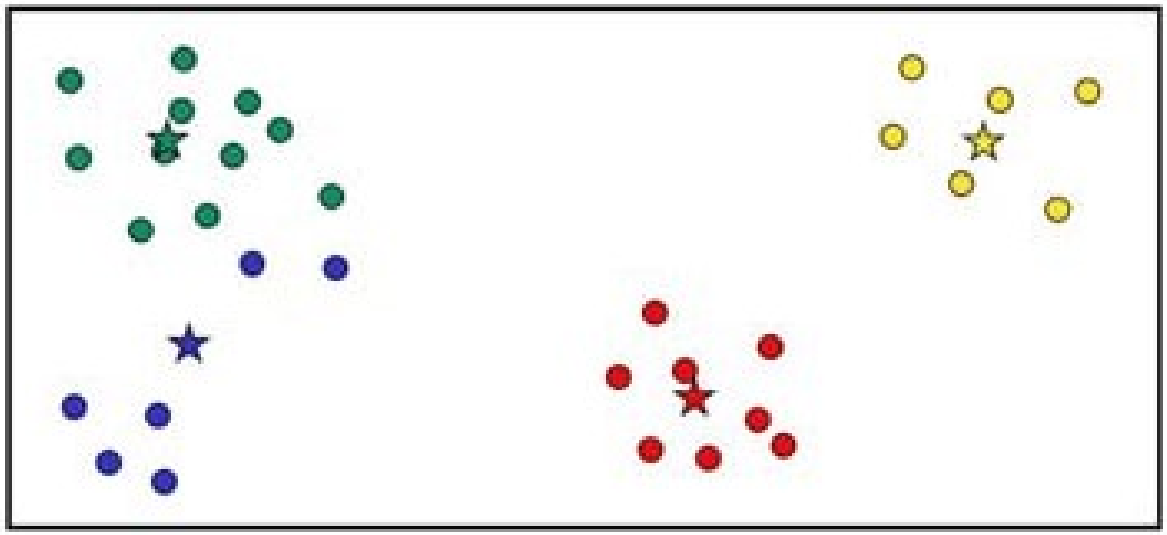
K-Means Example



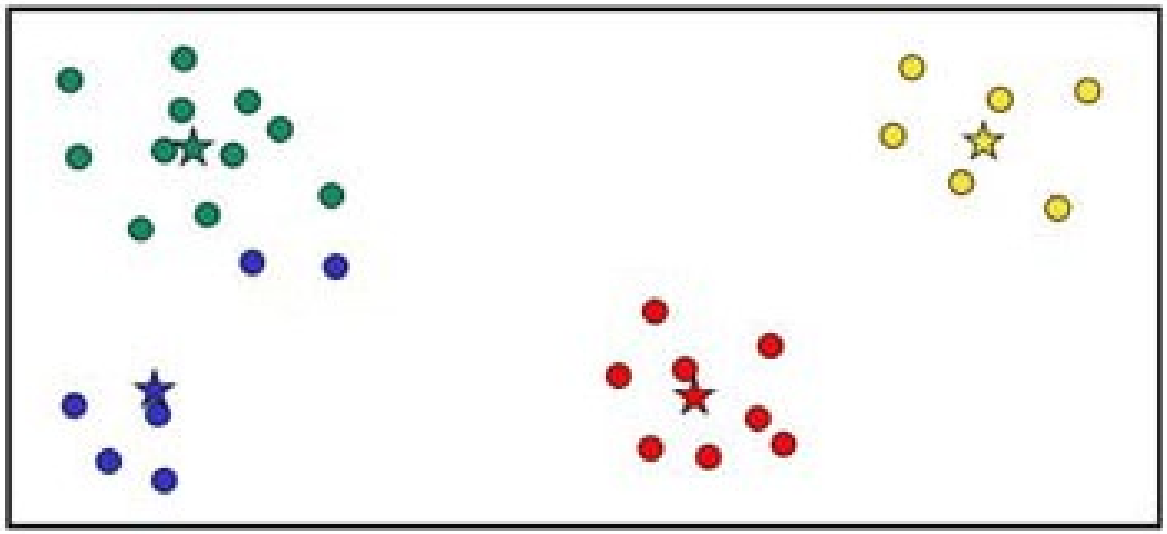
K-Means Example



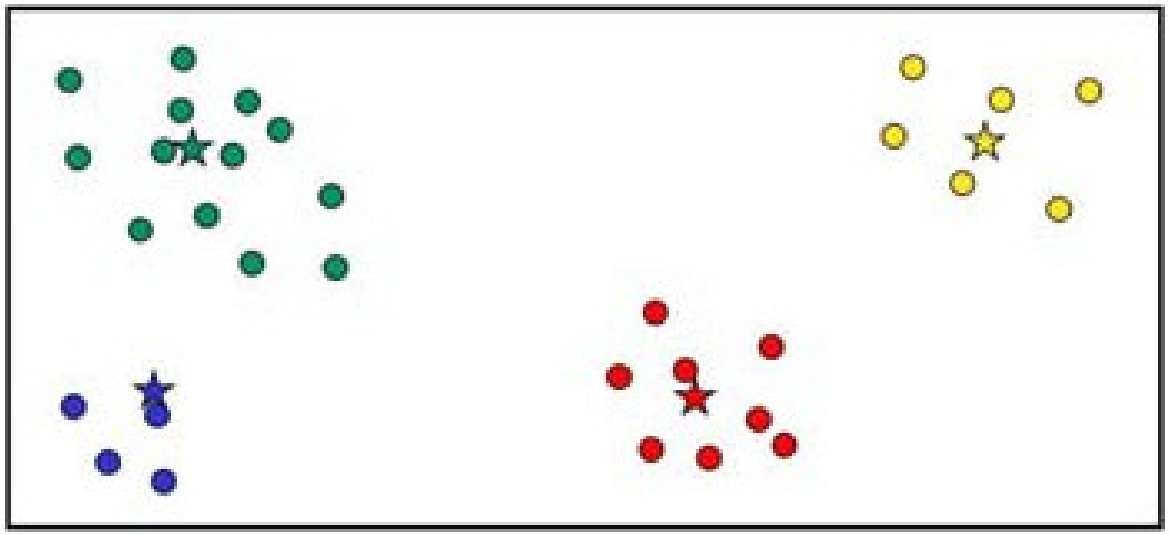
K-Means Example



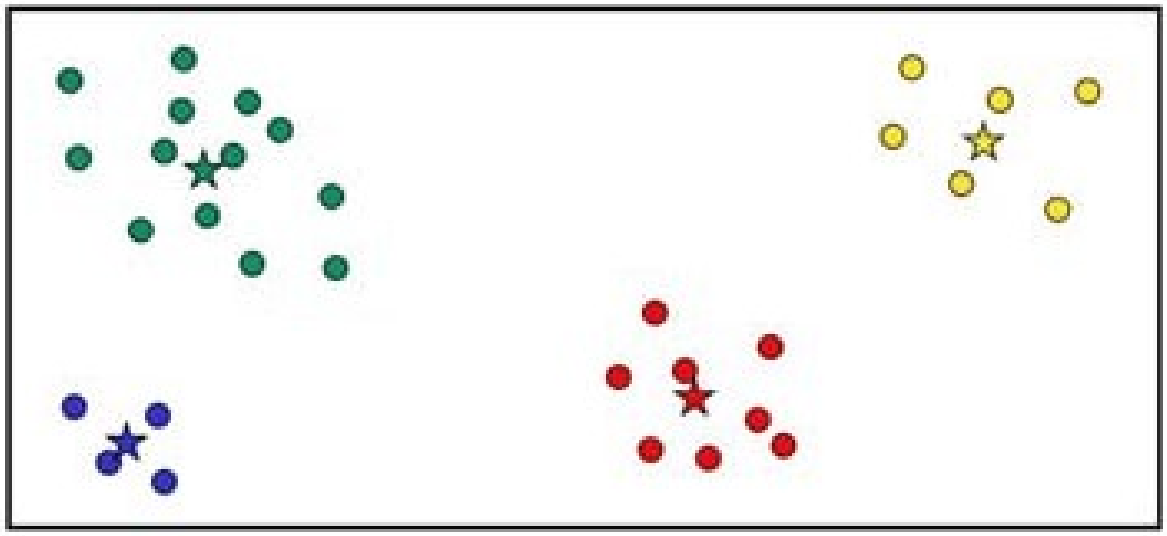
K-Means Example



K-Means Example



K-Means Example



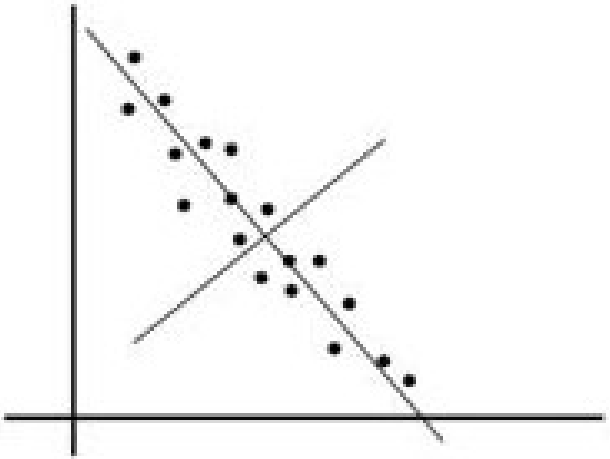
PCA
and
Performance evaluation using ROC Curves

Principal Components Analysis

- Given an n-dimensional real-valued space, data are often nearly restricted to a lower-dimensional subspace
- PCA helps us find such a subspace whose coordinates are linear functions of the originals

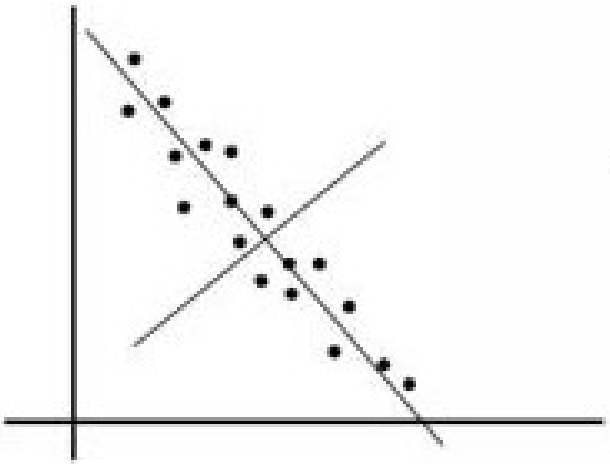
Principal Components Analysis

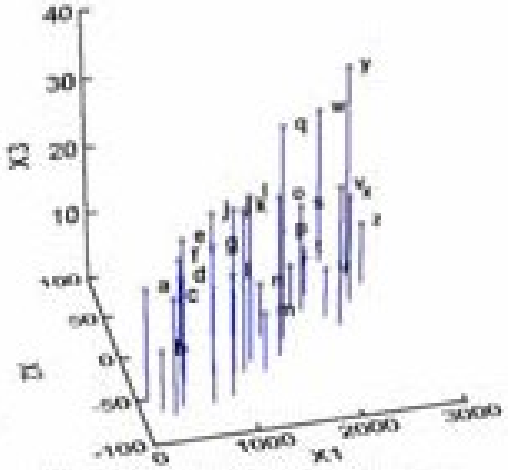
- Given an n-dimensional real-valued space, data are often nearly restricted to a lower-dimensional subspace
- PCA helps us find such a subspace whose coordinates are linear functions of the originals



Principal Components Analysis

- Given an n-dimensional real-valued space, data are often nearly restricted to a lower-dimensional subspace
- PCA helps us find such a subspace whose coordinates are linear functions of the originals





http://www.okstate.edu/artsci/ botany/ordinate/PCA.htm

Cartoon of algorithm

- Normalize the data (subtract mean, divide by stdev)

Cartoon of algorithm

- Normalize the data (subtract mean, divide by stdev)
- Find the line along which the data has the most variability: that's the first principal component

Cartoon of algorithm

- Normalize the data (subtract mean, divide by stdev)
- Find the line along which the data has the most variability: that's the first principal component
- Project the data into the n-1 dimensional space orthogonal to the line
- Repeat

Cartoon of algorithm

- Normalize the data (subtract mean, divide by stdev)
- Find the line along which the data has the most variability: that's the first principal component
- Project the data into the n-1 dimensional space orthogonal to the line
- Repeat

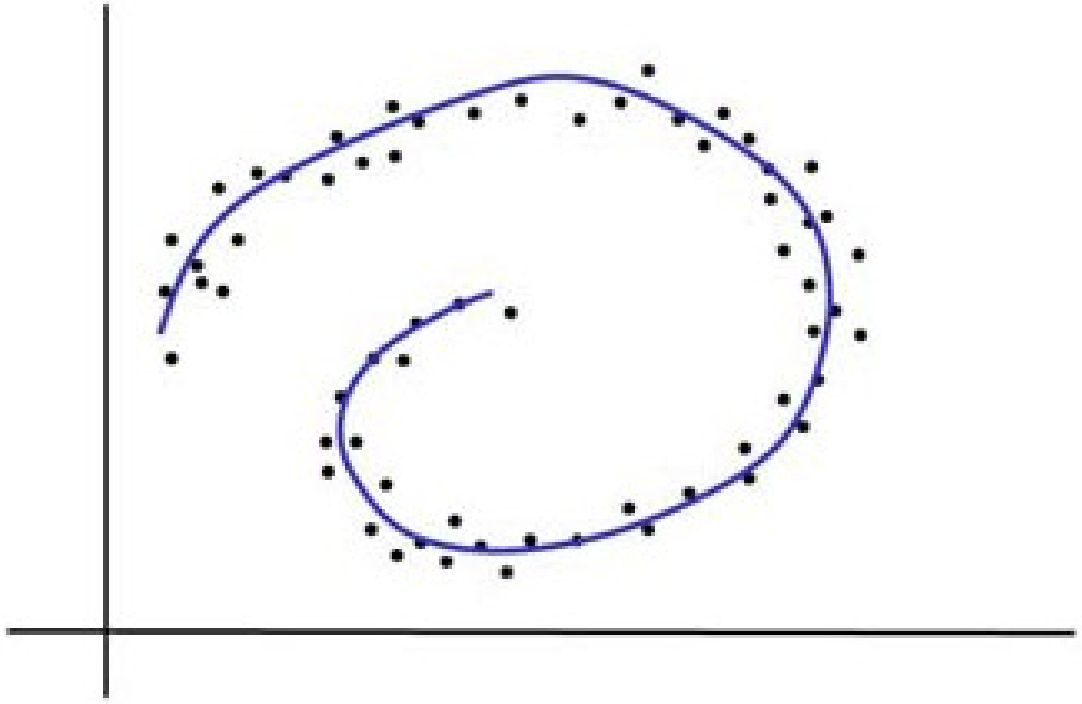
- Result is a new orthogonal set of axes
- First k give a lower-D space that represents the variability of the data as well as possible

Cartoon of algorithm

- Normalize the data (subtract mean, divide by stdev)
- Find the line along which the data has the most variability: that's the first principal component
- Project the data into the n-1 dimensional space orthogonal to the line
- Repeat

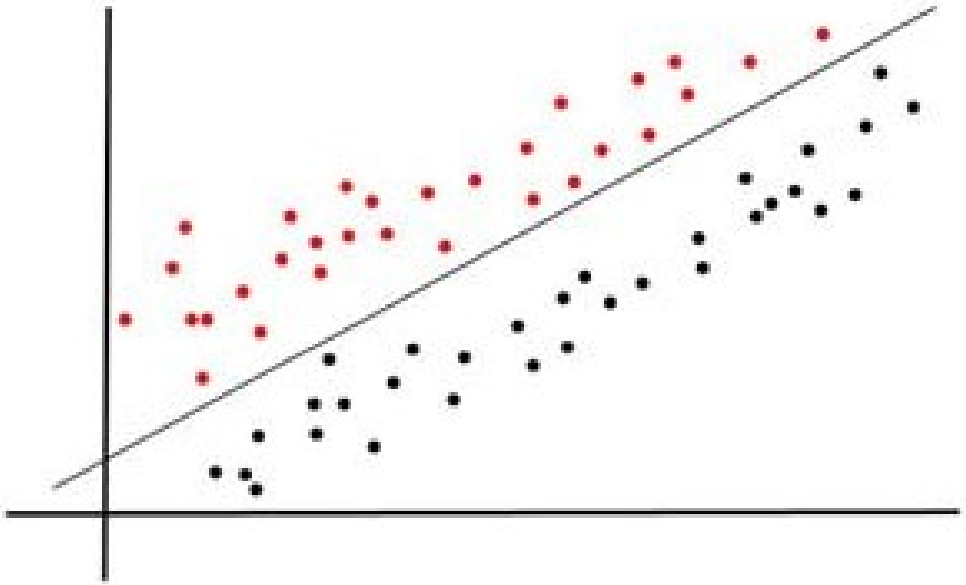
- Result is a new orthogonal set of axes
- First k give a lower-D space that represents the variability of the data as well as possible
- Really: find the eigenvectors of the covariance matrix with the k largest eigenvalues

Linear Transformations Only

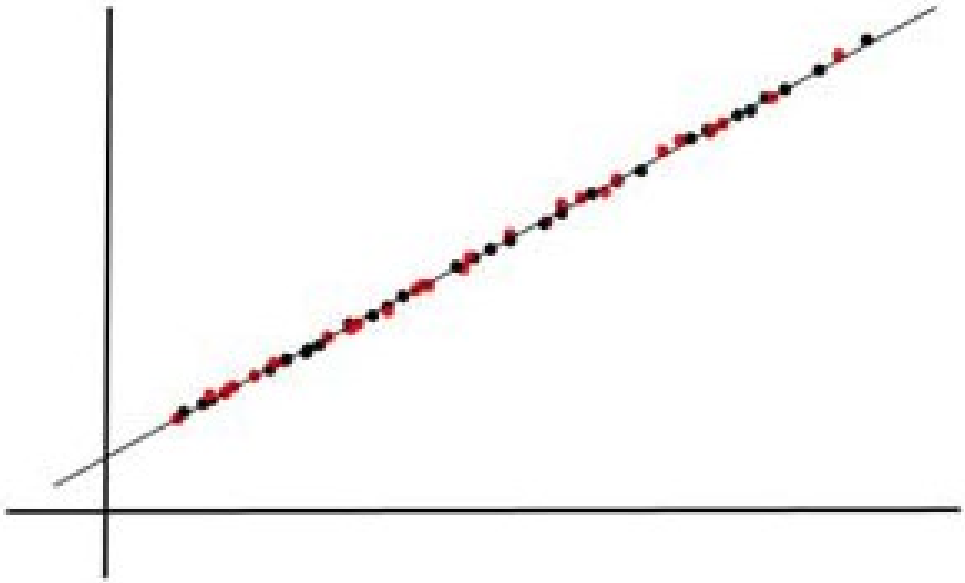


There are fancier methods that can find this structure

Insensitive to Classification Task



Insensitive to Classification Task

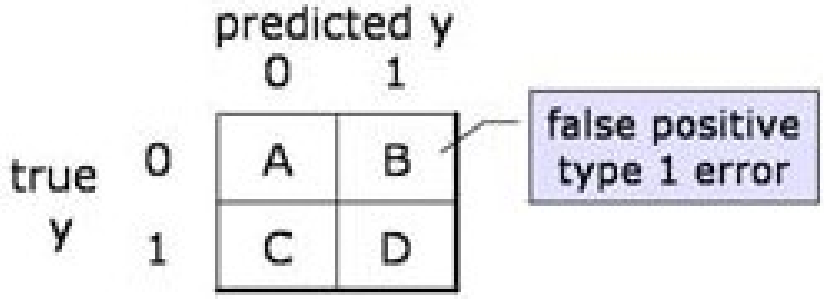


There are fancier methods that can take class into account

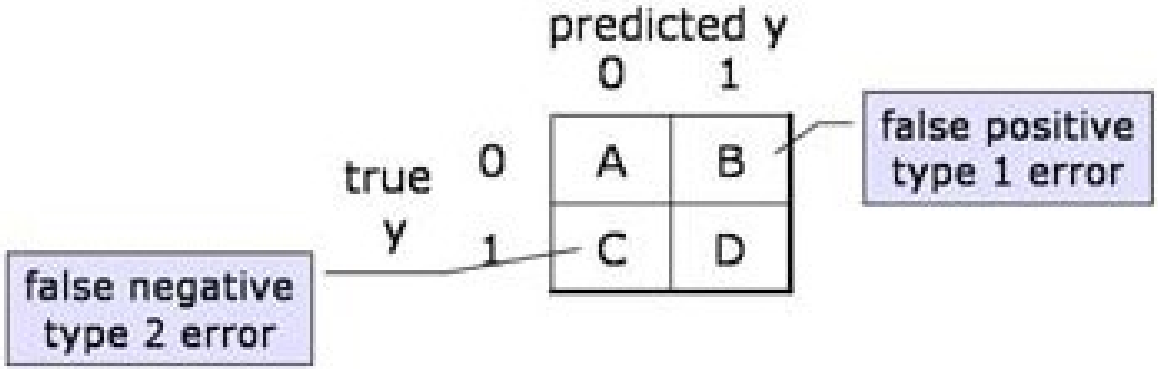
Validating a Classifier

		predicted y 0	1
true y	0	A	B
	1	C	D

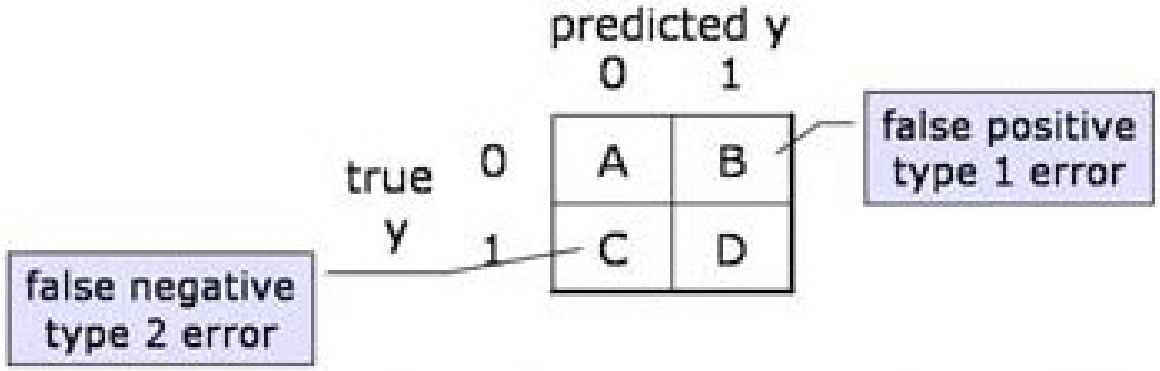
Validating a Classifier



Validating a Classifier

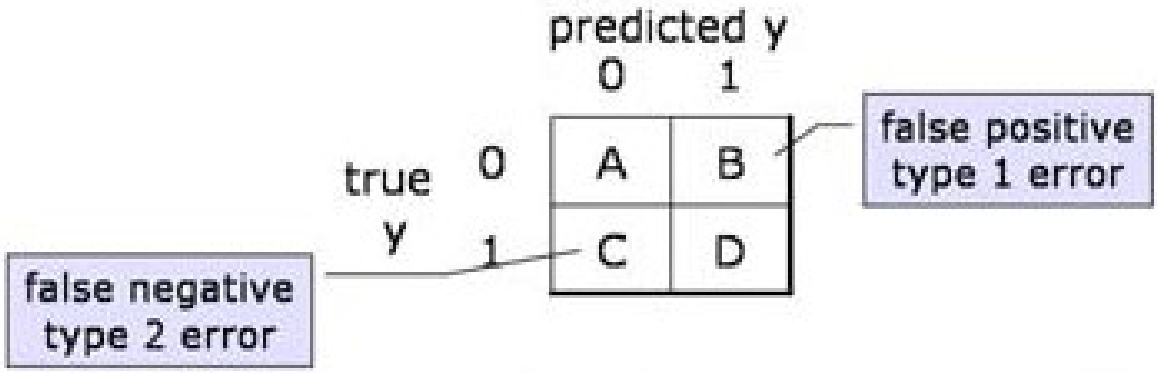


Validating a Classifier



- sensitivity: P(predict 1 | actual 1) = D/(C+D)
  - "true positive rate" (TP)

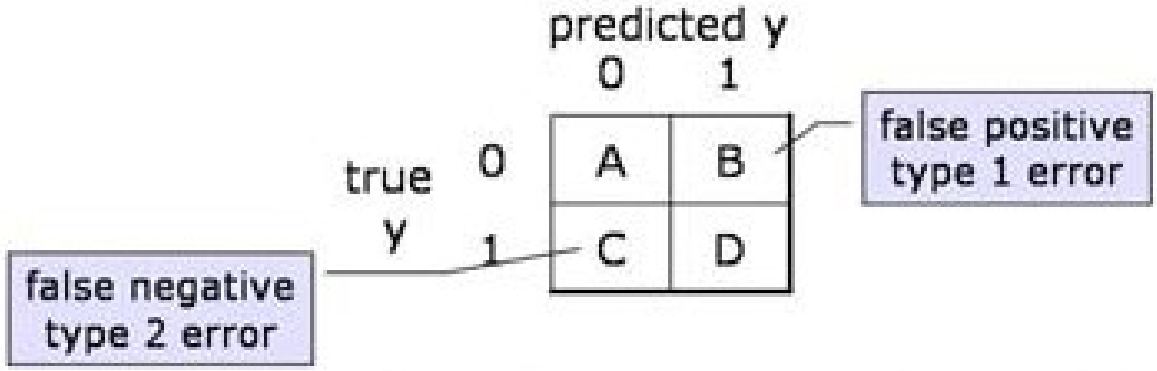
Validating a Classifier



- sensitivity: P(predict 1 | actual 1) = D/(C+D)
  - "true positive rate" (TP)

- specificity: P(predict 0 | actual 0) = A/(A+B)

Validating a Classifier



- sensitivity: P(predict 1 | actual 1) = D/(C+D)
  - "true positive rate" (TP)
- specificity: P(predict 0 | actual 0) = A/(A+B)
- false-alarm rate: P(predict 1 | actual 0) = B/(A+B)
  - "false positive rate" (FP)

Cost Sensitivity

- Predict whether a patient has pseuditis based on blood tests
  - Disease is often fatal if left untreated
  - Treatment is cheap and side-effect free

Cost Sensitivity

- Predict whether a patient has pseuditis based on blood tests
  - Disease is often fatal if left untreated
  - Treatment is cheap and side-effect free

- Which classifier to use?
  - Classifier 1: TP = 0.9, FP = 0.4

Cost Sensitivity

- Predict whether a patient has pseuditis based on blood tests
  - Disease is often fatal if left untreated
  - Treatment is cheap and side-effect free

- Which classifier to use?
  - Classifier 1: TP = 0.9, FP = 0.4
  - Classifier 2: TP = 0.7, FP = 0.1

Build Costs into Classifier

- Assess costs of both types of error
  - use a different splitting criterion for decision trees
  - make error function for neural nets asymmetric; different costs for each kind of error
  - use different values of C for SVMs depending on kind of error

Tunable Classifiers

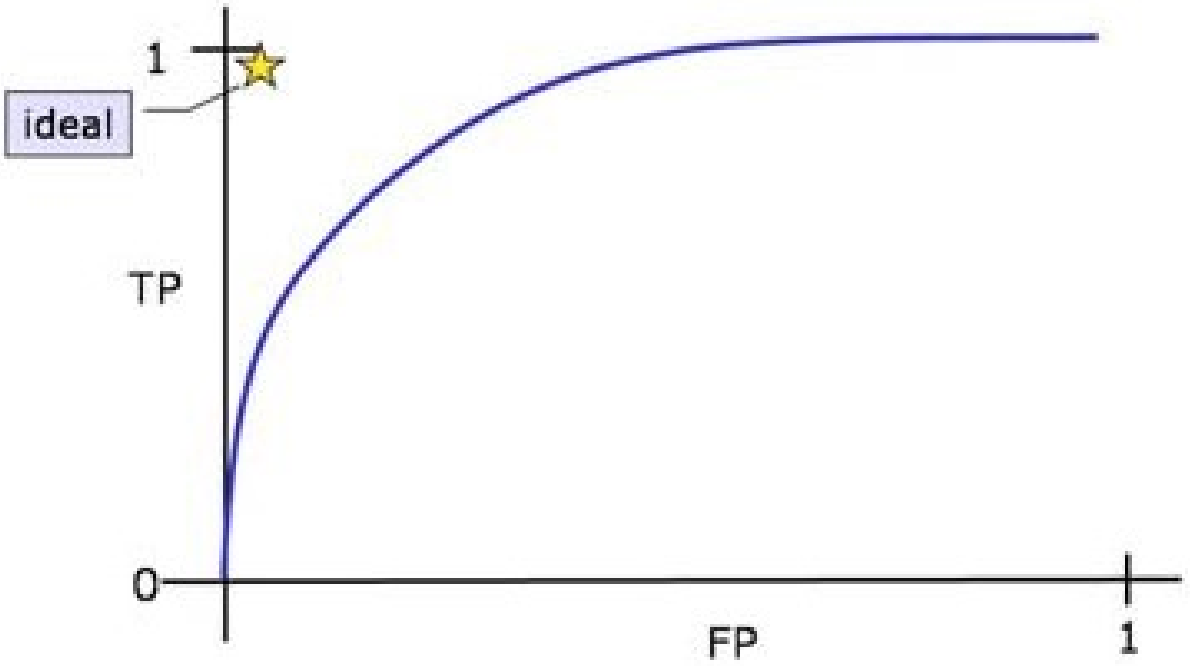
- Classifiers that have a threshold (naïve Bayes, neural nets, SVMs) can be adjusted, post learning, by changing the threshold, to make different trade-offs between type 1 and type 2 errors

Tunable Classifiers

- Classifiers that have a threshold (naïve Bayes, neural nets, SVMs) can be adjusted, post learning, by changing the threshold, to make different trade-offs between type 1 and type 2 errors
  - C_1,C_2: costs of errors
  - P: percentage of positive examples
  - x: tunable threshold
  - TP(x): true positive rate at threshold x
  - FP(x): false positive rate at threshold x
- Expected Cost = $C_1P(1\text{-}TP(x)) + C_2(1\text{-}P)FP(x)$
- choose x to minimize expected cost

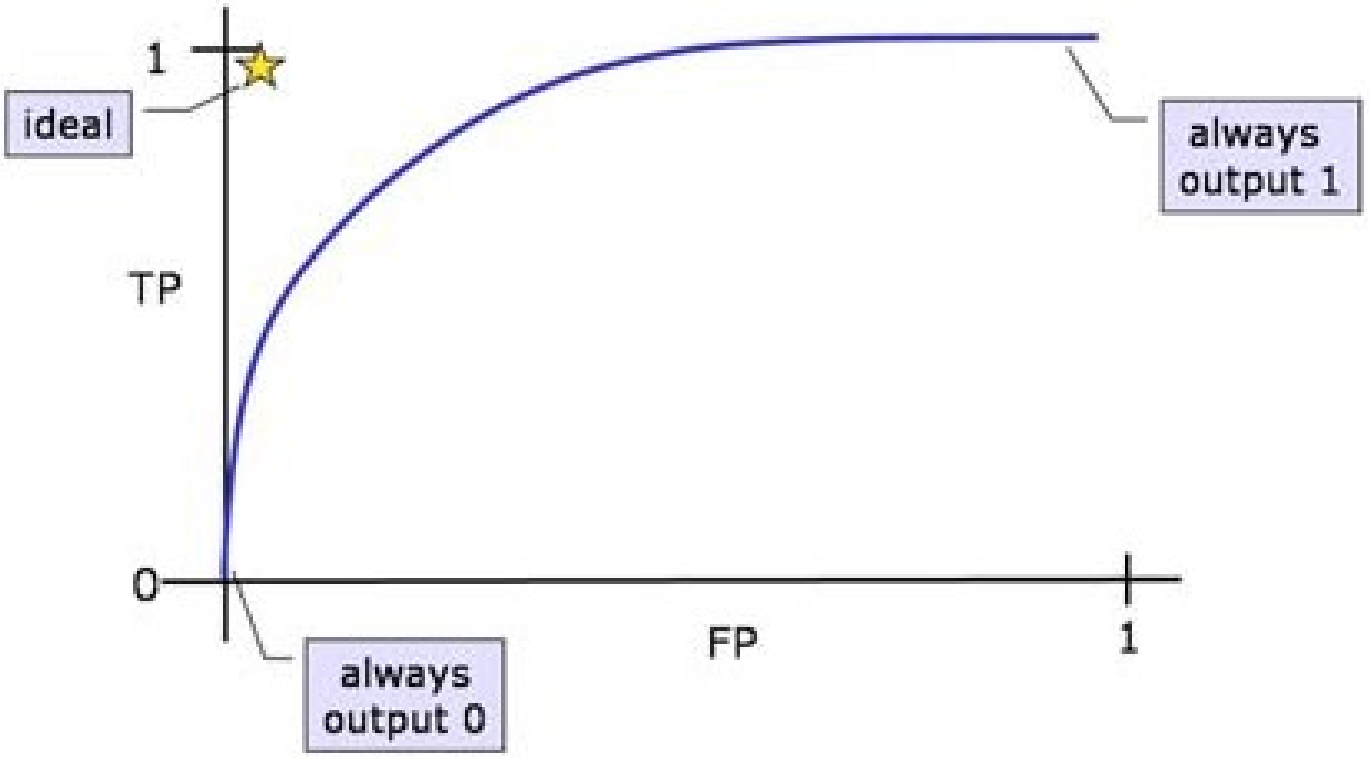
ROC Curves

- "receiver operating characteristics"



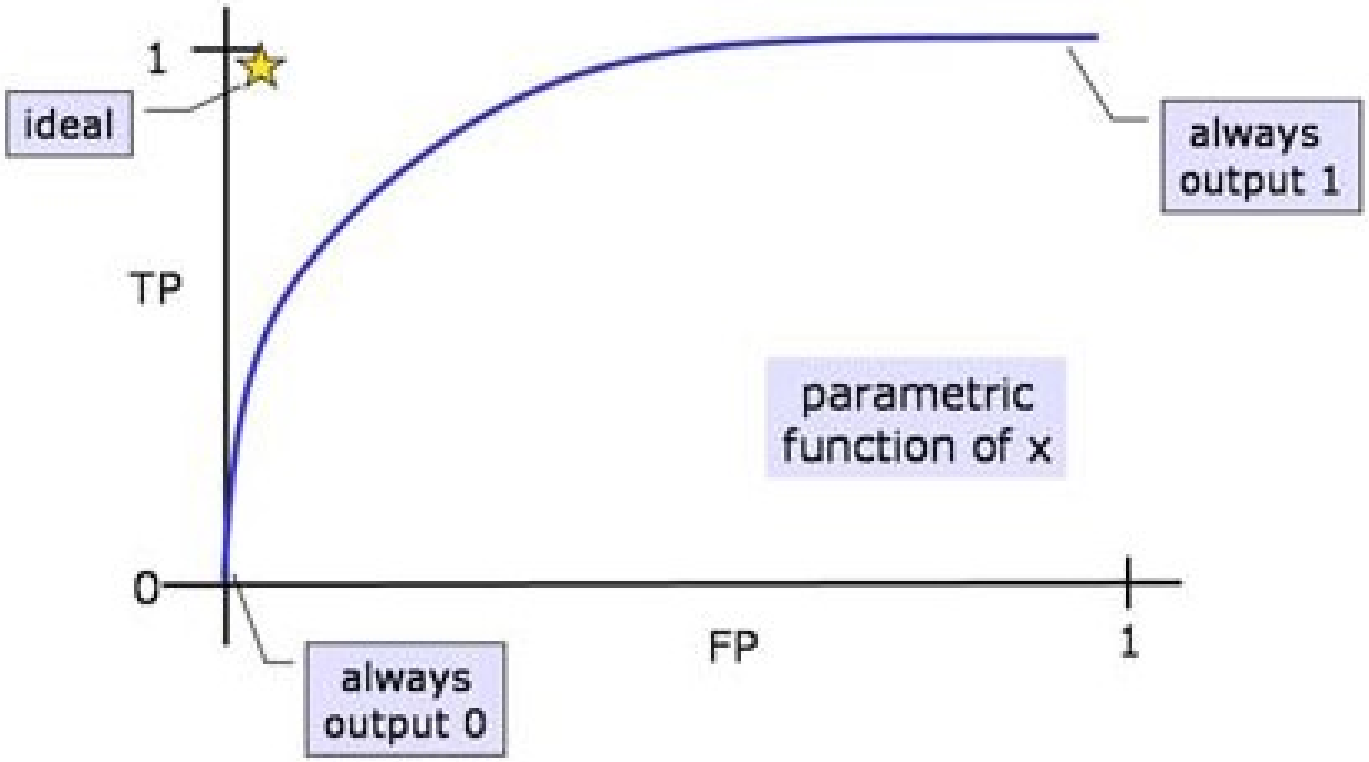
ROC Curves

- "receiver operating characteristics"



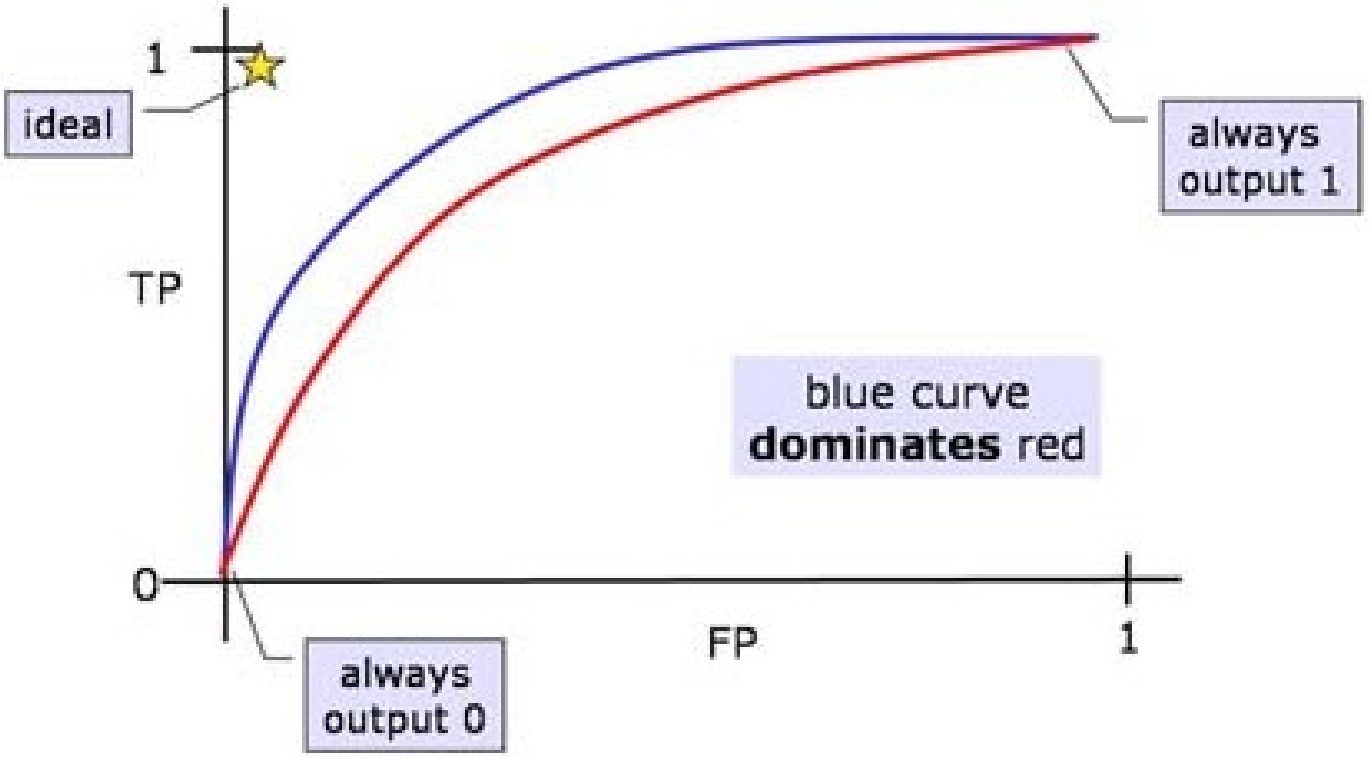
ROC Curves

- "receiver operating characteristics"



ROC Curves

- "receiver operating characteristics"



Many more issues!

- Missing data
- Many examples in one class, few in other (fraud detection)
- Expensive data (active learning)
- ...

References

- Flach, P. (2012). Machine Learning: The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press
- Russell S., & Norvig, P. (2021). Artificial Intelligence A Modern Approach. 4th Edition: Pearson Education https://www.academia.edu/45126798/Artificial_Intelligence_A_Modern_Approach_4th_Edition_?auto=download
- Mitchell T. M. (1997). Machine Learning : McGraw Hill International
- Mohri M., Rostamizadeh A., Talwalkar A. (2018). Foundations of Machine Learning. 2nd Edition: MIT Press
- Nilsson, N.J. (1998). Artificial Intelligence: a new synthesis. Morgan Kaufmann.
- Han, J.., Kamber, M., Pei, J (2012). Data mining: concepts and techniques.Elsevier Inc.
- https://www.ri.cmu.edu/research/
- https://archive.ics.uci.edu/ml/datasets/heart+Disease
- http://www.clopinet.com/isabelle/Projects/SVM/applist.html
- https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original)
- Lucila Ohno-Machado, and Peter Szolovits. HST.947 Medical Artificial Intelligence. Spring 2005. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA.

Thank You