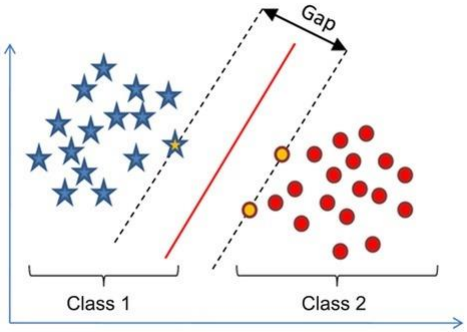
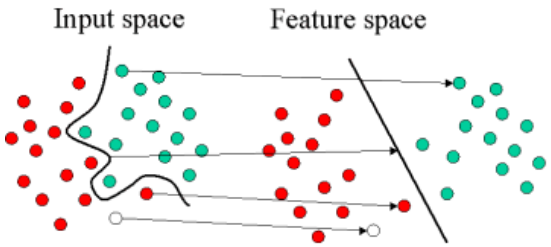
Support Vector Machine

2018/2019 Semester I

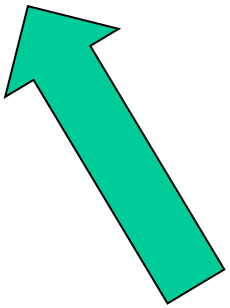


Associate Professor
HUANG, Ke-Wei



Today's Class

1. SVM Overview and Linearly Separable SVM
   - Predicting binary label
   - Linearly Separable SVM is unrealistic but good for explaining SVM.



2. Linearly Inseparable: two solutions
   1. SVM with Soft Margin
   2. SVM with Kernel Function

3. Sketch of the SVM Regression

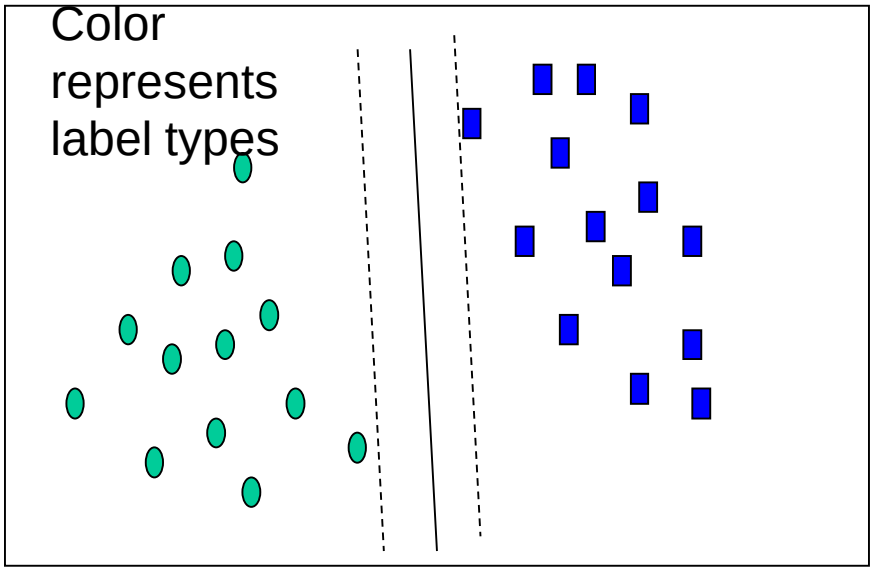
SVM—Support Vector Machines

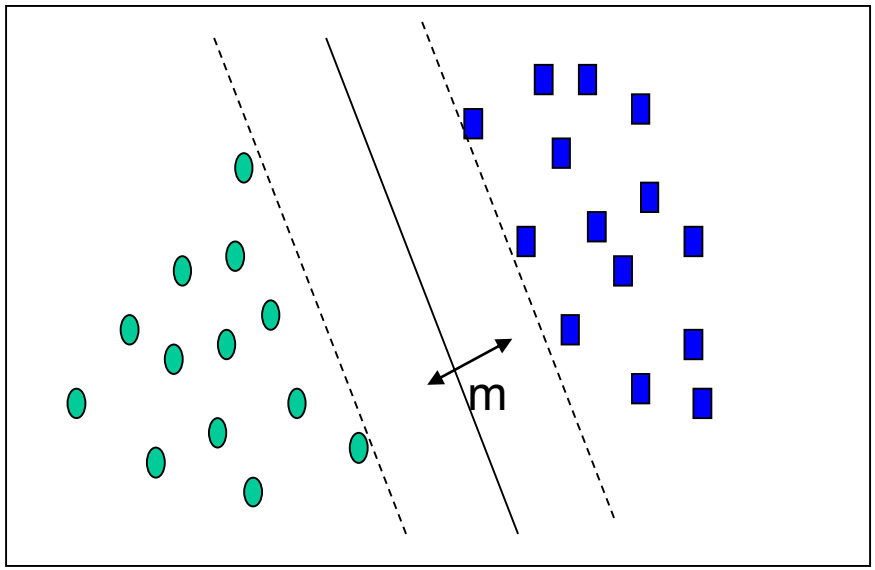
- The underlying (complicated) mathematics of SVM has been invented for many years but the introduction of SVM into data mining is quite recent.
- Vapnik and colleagues (1992)—groundwork from Vapnik & Chervonenkis' statistical learning theory in 1960s.
- One reason makes SVM popular around 2000 is that it has a very strong mathematical foundation and can be mapped to well-studied convex optimization problems.
- The other reason SVM became popular is because performance is generally good if you tune the parameters well (before 2010).

SVM—History and Applications

- Let's start our discussion with classifying a binary label.
  - SVM can be used to predict numerical variables (regression, later today)
  - SVM can predict multi-class labels but is tricky.
- Mathematics of SVM is complicated. Most textbooks will start with 1 binary label and only 2 numeric features to visualize how SVM works.
- The goal of a SVM is to create a flat boundary called a **hyperplane, which divides the high-dimensional space of examples to create fairly homogeneous** partitions on either side.

SVM—When Data Is Linearly Separable

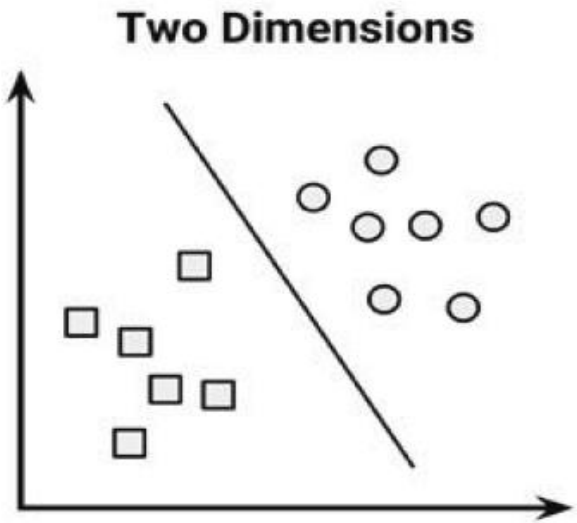


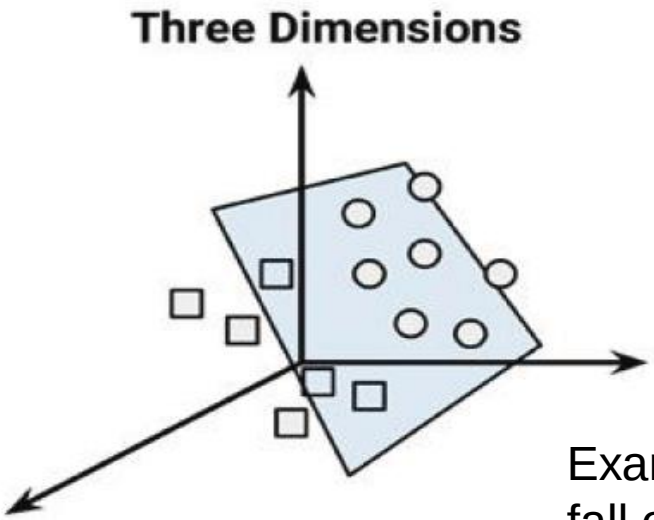


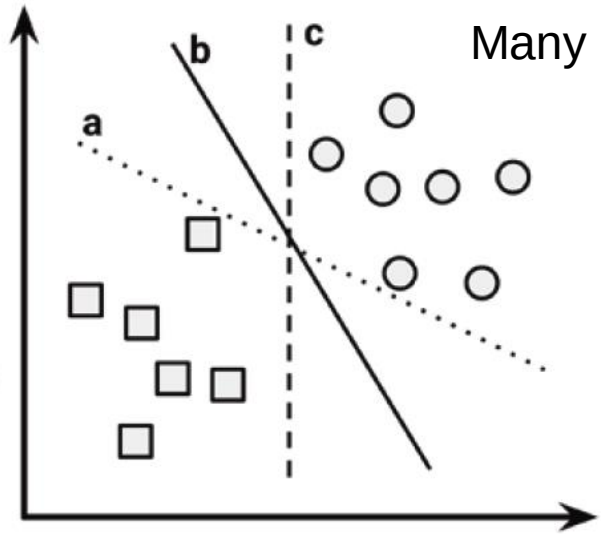
Let data be ($\mathbf{X}_1$, $\mathbf{X}_2$, y) where y has two types: green and blue.

There are many lines (hyperplanes) that can separate two classes but we want to **find the best line => *SVM searches for the hyperplane with the largest margin (distance)*, i.e., maximum marginal hyperplane (MMH)**

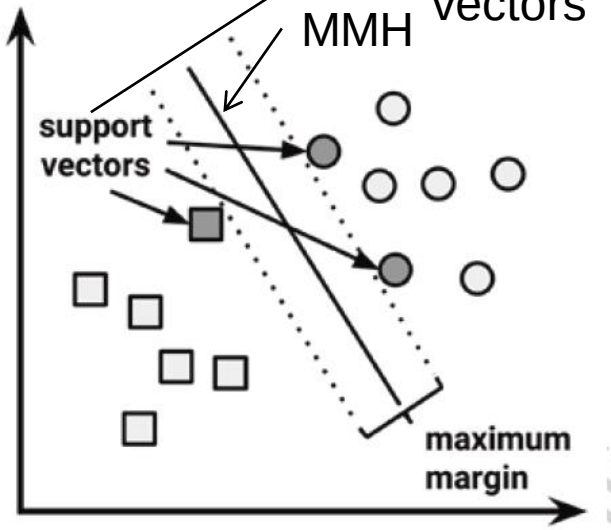
SVM—Linearly Separable







Many lines



MMH

Examples that fall on the lines are support vectors

SVM—Linearly Separable

- In terms of Mathematics, a separating hyperplane (e.g., a line in 2D space, or a plane in 3D) can be written as

  $$\mathbf{W} \bullet \mathbf{X} + b = 0,$$

  where $\mathbf{W} = \{w_1, w_2, \ldots, w_n\}$ is a weight vector similar to regression coefficients. **X is a vector** of your data matrix and *b* is a scalar (bias, like intercept) for estimation.
- In 2-D (only 2 input features), it can be written as

  $$w_1 x_1 + w_2 x_2 + b = 0$$
- In 3-D (only 3 input features), it can be written as

  $$w_1 x_1 + w_2 x_2 + w_3 x_3 + b = 0$$

SVM—Linearly Separable

- In the 2D example, we have 3 weights (including **b**). We can scale 3 weights by any number and the line is still the same. We will scale it the following way.

- The hyperplane defining the regions of y=+1 or -1 is

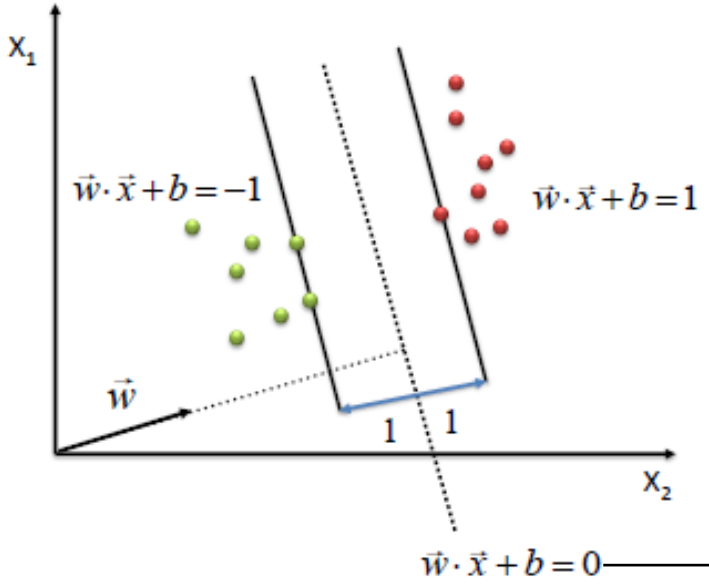
  $H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

  $H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$

- This form is specified for theoretical reasons for us to further develop other mathematical properties.

Support Vectors

- Conceptually, the **support vectors** are the essential or critical training examples — they lie closest to the decision boundary.
- **If all other training examples are removed and the training is repeated, the same separating hyperplane would be found.**



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

2-norm of w, see next slide

$s.t.$

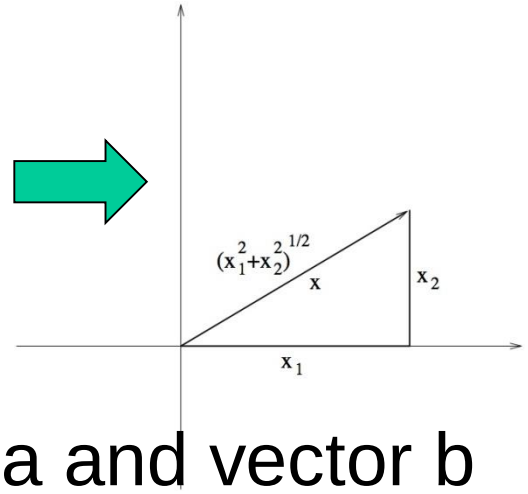
$(w\cdot x+b)\geq 1, \forall x \text{ of class } 1$

$(w\cdot x+b)\leq -1, \forall x \text{ of class } 2$

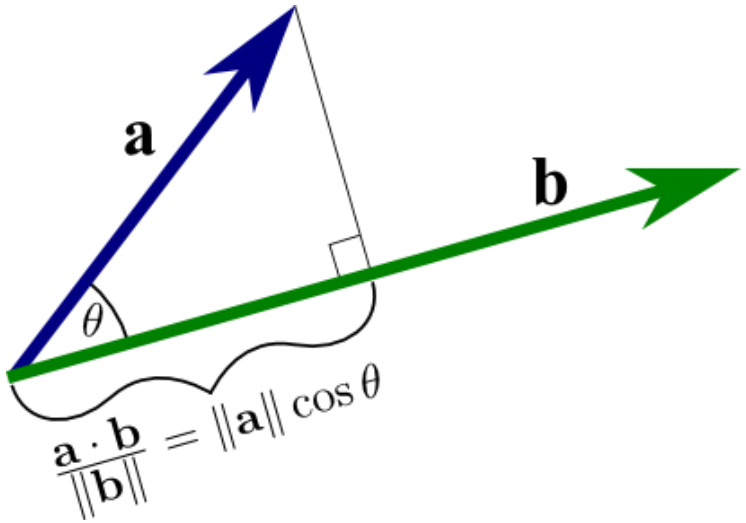
Decision boundary

Review of Linear Algebra

A vector's Euclidean norm (2-norm) is the length of that vector to the origin.



• Inner Product between vector a and vector b is a **similarity measure** in terms of the angle.



Euclidean distance and Cosine Similarity are two major functions to calculate the similarity between two vectors (two rows of data) in data mining.

Solving Linearly Separable SVM

The linearly separable SVM classifier is derived by solving the following optimization problem.

$$\min_{w,b} \quad \|w\|^2$$

subject to: $y_i(w \times x_i + b) \geq 1 \quad \forall \; i$

- The objective function is the same as the previous slides because when we maximize that ratio $2/||w||^2$, it is equivalent to minimizing $||w||^2$
- The two constraints on the previous slide is the same as this inequality because y equals to either +1 or -1. When you insert values, you will see this inequality is the same as those two inequalities on the previous slide.

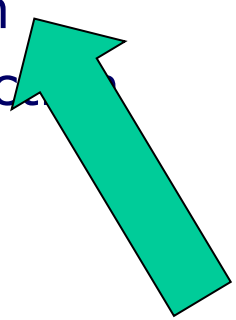
Solving Linearly Separable SVM

- The optimization problem on the previous slide is a quadratic optimization problem, which is well-studied in applied math. and Operations Research (OR).
- Quadratic optimization problem means maximize/minimize a quadratic function like the objective function in our problem, subject to a set of linear constraints.
- Objective functions are quadratic functions of weights.
- Constraints are linear functions of weights.
- Many mathematical methods and packages for solving this problem (some shortly later.).

Today's Class

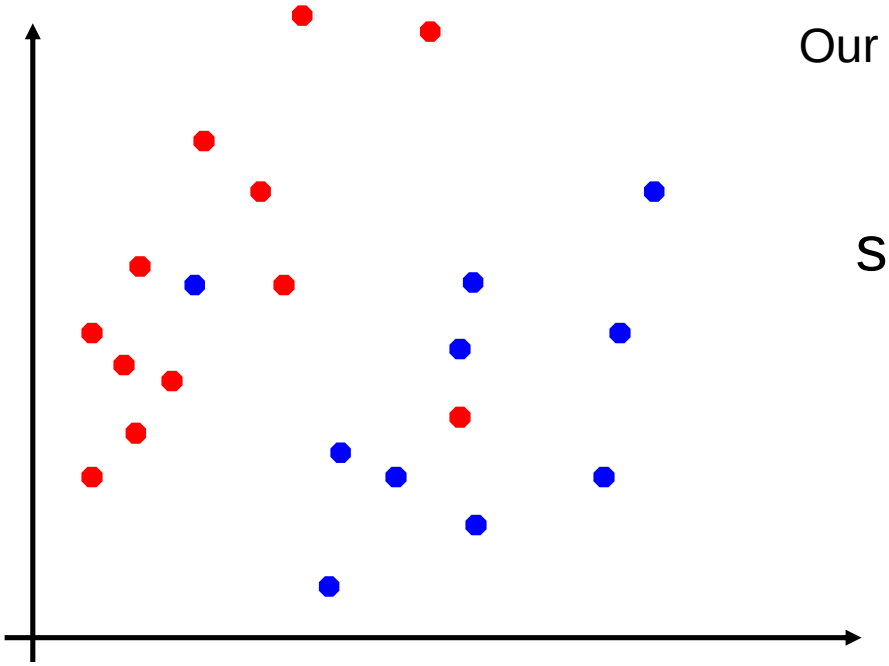
1. SVM Overview and Linearly Separable SVM
   - Predicting binary label
   - Linearly Separable SVM is unrealistic but good for explaining SVM.

2. Linearly Inseparable: two solutions
   1. SVM with Soft Margin
   2. SVM with Kernel Func[illegible]



3. SVM Regression

SVM Classification with Soft Margin



Our current problem is given by

$$\min_{w,b} \quad \|w\|^2$$

subject to:

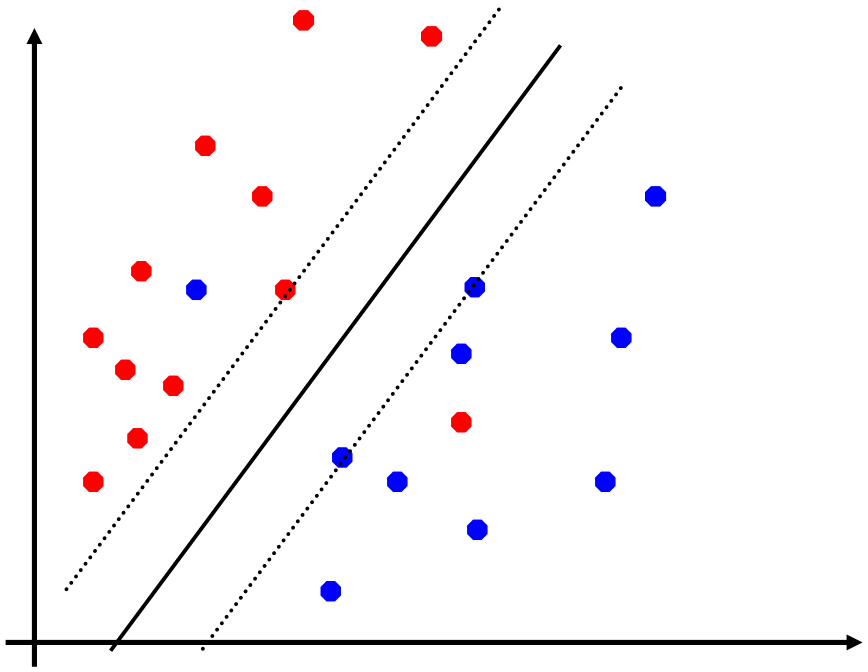
$$y_i(w \times x_i + b) \geq 1 \quad \forall \; i$$

When there is no line to linearly separate points (as in almost all problems), what to do?

Method 1: We will allow some errors (soft margin)

Method 2: We will play the Kernel Function tricks: when we transform/map the original features space into a higher dimensional space, it becomes separable.

Soft Margin Classification



$$\min_{w,b} \quad \|w\|^2$$

subject to:

$$y_i(w \times x_i + b) \geq 1 \quad \forall i$$

We'd like to learn something like this, but our constraints won't allow it ☹

Adding Slack Variables

$$\min_{w,b} \quad \|w\|^2$$

subject to:

$$y_i(w \times x_i + b) \geq 1 \quad \forall\, i$$

$$\min_{w,b} \quad \|w\|^2 + C\sum_i \xi_i$$

subject to:

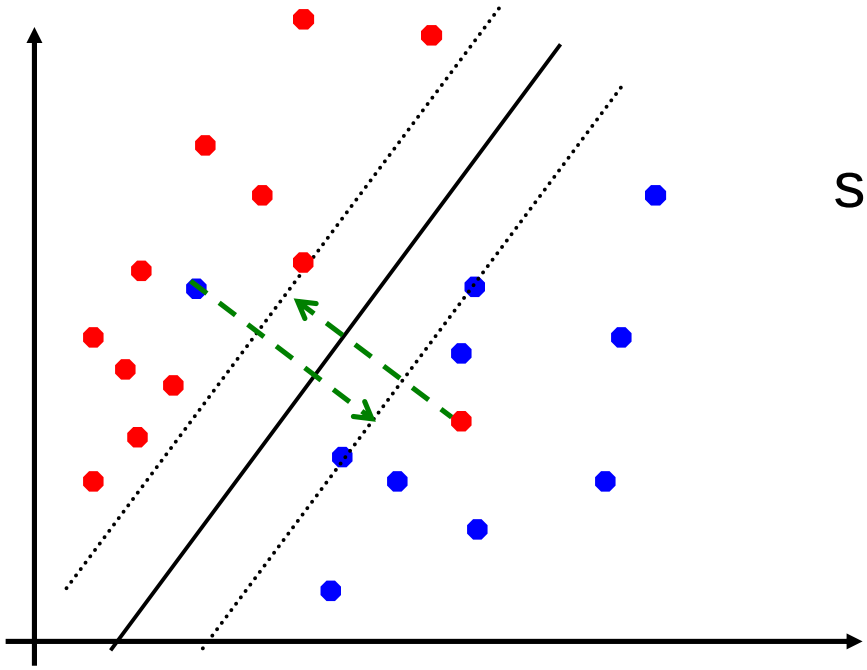
$$y_i(w \times x_i + b) \geq 1 - \xi_i \quad \forall\, i$$

$$\xi_i \geq 0$$

slack variables:
one for each
example/observation

What effect does this have?

Slack Variables



slack penalties

$$\min_{w,b} \quad \|w\|^2 + C\sum_i \xi_i$$

subject to:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0$$

Now we allow observations to be mis-classified to the other side. But We will penalize that kind of errors by adding C*ξi in the objective function.

Slack Variables

margin

trade-off between margin maximization and penalization

$$\min_{w,b} \quad \|w\|^2 + C\sum_i \xi_i$$

penalized by how far from "correct"

subject to:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i \quad \forall i$$

allowed to make a mistake

$$\xi_i \geq 0$$

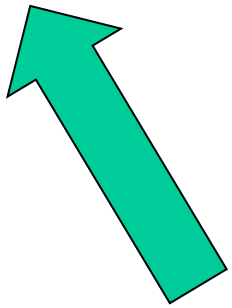
Still a quadratic optimization problem!

Today's Class

1. SVM Overview and Linearly Separable SVM
   - Predicting binary label
   - Linearly Separable SVM is unrealistic but good for explaining SVM.

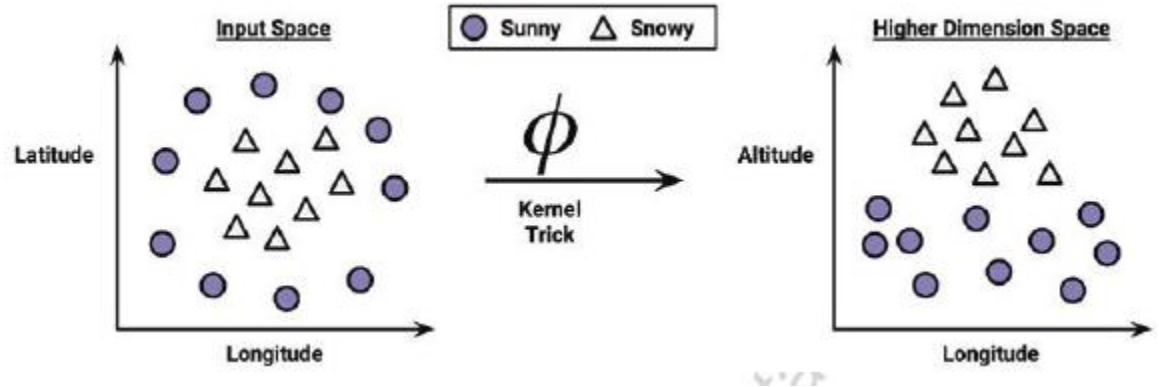
2. Linearly Inseparable: two solutions
   1. SVM with Soft Margin
   2. SVM with Kernel Function, **the most important trick**!

3. SVM Regression



SVM—Linearly Inseparable

- In many real-world applications, the relationships between variables are nonlinear.
- A key feature of SVMs is the ability to map the problem into a higher dimensional space using a process known as the **kernel trick.**
- **In doing so, a nonlinear relationship may suddenly** appear to be quite linear.
- One intuitive, non-math. example.



Simplest Example from 2D to 4D

- Imagine an example that within a circle, the labels are 1; outside a circle, the labels are 0. This is not linearly separable.
- Also, assume the function is $8(x_1 - 1)^2 + 50(x_2 - 2)^2 = 1$.
- But if we map from 2D space (x_1, x_2) to 4D space (x_1, x_2, x_1^2, x_2^2) then the above equation becomes "linear" => MAGIC!
- This is a bit similar to the "naïve features engineering" approach I did when I explained Neural Network. But now it is automated if you choose polynomial Kernel SVM.

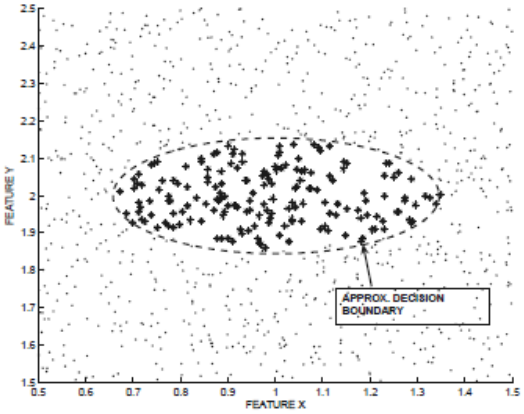
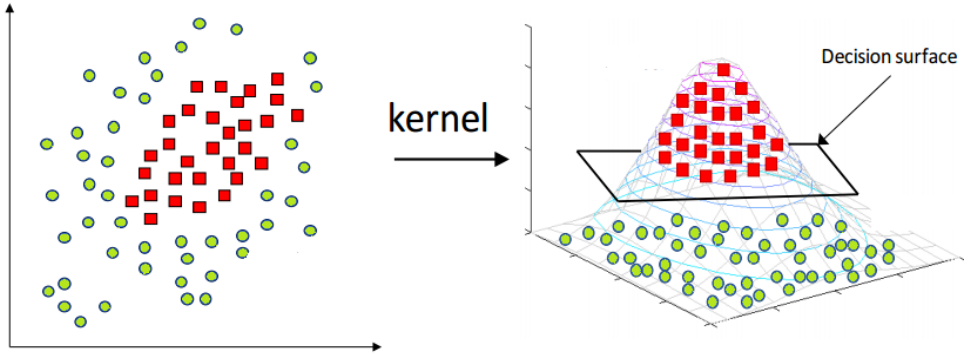
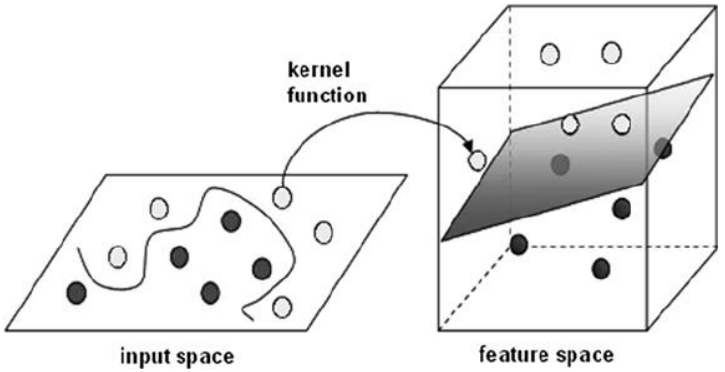


Figure 10.8: Nonlinear decision surface

Other Visualizations of Kernel Mapping

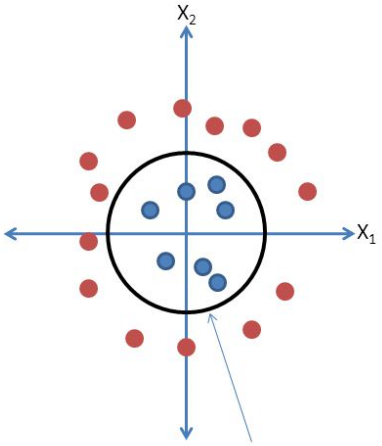




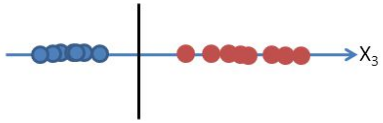
The "Kernel Trick"

The Kernel Trick is to add a new input variable that is computed from the existing ones.

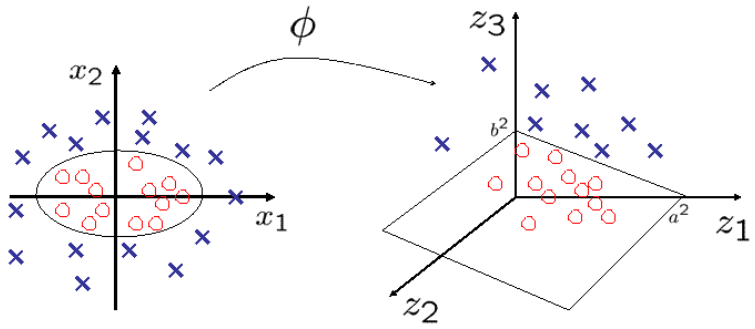
Let $X_3 = \sqrt{X_1^2 + X_2^2}$



In the original feature space, the linear separator looks like a circle.



Now there's a linear separator!



$$\phi : (x_1, x_2) \longrightarrow (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

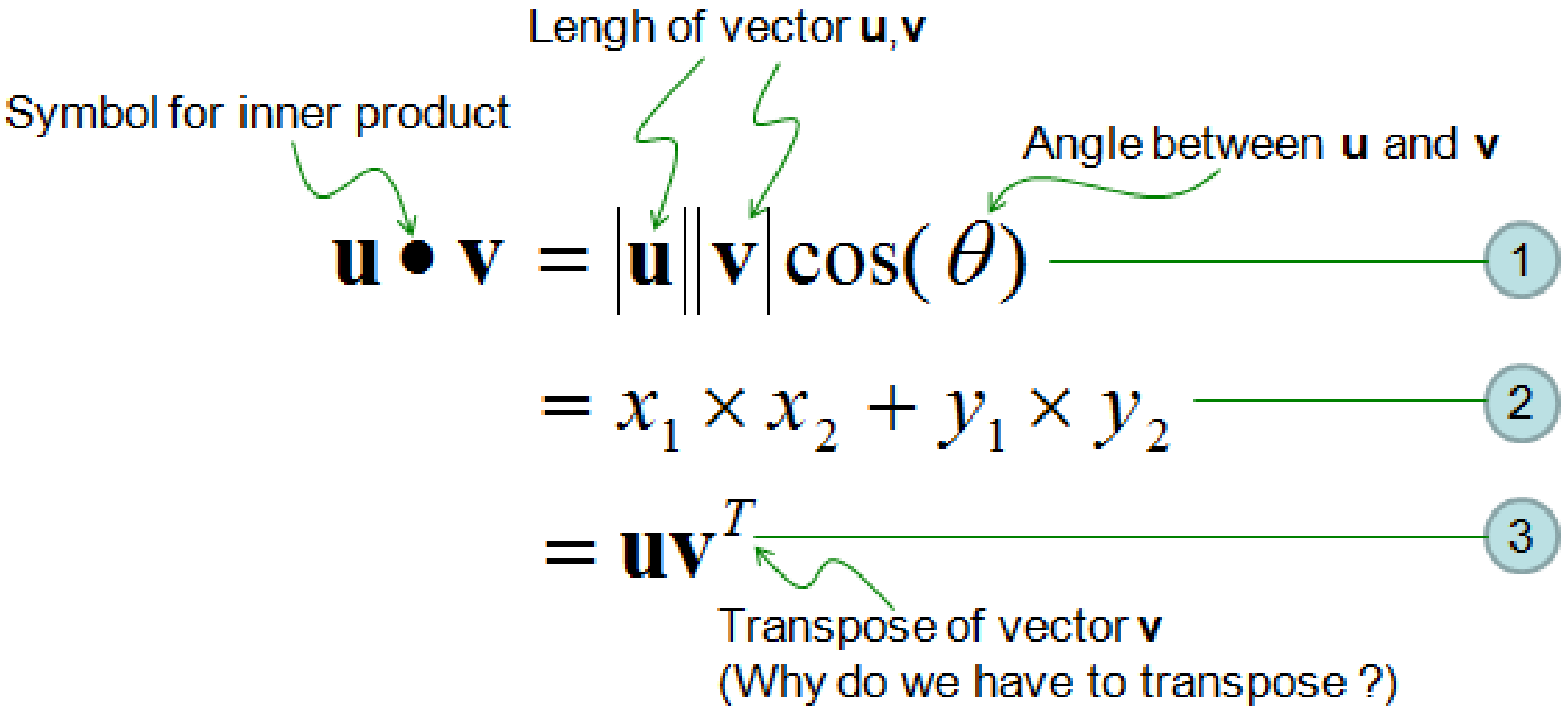
$$\left(\frac{x_1}{a}\right)^2 + \left(\frac{x_2}{b}\right)^2 = 1 \longrightarrow \frac{z_1}{a^2} + \frac{z_3}{b^2} = 1$$

General Math. Formulation

- SVM has several nice mathematical properties. One is the optimization problem we discussed so far can be mapped to an equivalent optimization problem that only involves the "dot product" calculations of two vectors of data examples: X_i and X_j. (This is very technical, see Appendix and the uploaded file on IVLE for more details if you are interested in.)
- Also, the "dot product function" can be considered as a similarity metric between two vectors (two rows of your data).
- This can be leveraged so that after we transformed the original data to $\Phi(\overline{X})$, we only need to care about the kernel function K(), which also plays the role of similarity function.

$$K(\overline{X_i}, \overline{X_j}) = \Phi(\overline{X_i}) \cdot \Phi(\overline{X_j})$$

Inner Product of Two Vectors



As a result, inner product is similar to “cosine similarity” for text mining that we will learn later this semester.

Different Kernel functions

- There are 4 commonly used Kernel functions that are supported by R package e1071, function svm()

1. Linear SVM = Linear Kernel Function
2. Polynomial SVM
3. Sigmoid kernel
4. Gaussian RBF kernel

- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$.
- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i{}^T \mathbf{x}_j + r)^d$, $\gamma > 0$.
- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2)$, $\gamma > 0$.
- sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i{}^T \mathbf{x}_j + r)$.

For this package, γ is multiplying the Euclidean distance function. In some other cases, γ could be a denominator.

Different Kernel functions and parameters

1. Linear SVM = Linear Kernel Function
   - NO parameters for you to tune.
   - Cosine similarity
2. Polynomial SVM
   - 3 parameters to tune.
   - d is the most important one.
   - Cosine similarity
3. Sigmoid kernel
   - 2 parameters to tune.
   - Cosine similarity
4. **Gaussian RBF kernel**
   - **1 parameter to tune.**
   - **Euclidean distance as similarity**

SVM Parameters for Tuning

1. *C*: the penalty on the soft margin.
   - Called cost in R, e1071, SVM()
2. Kernel function?
3. Depending on the chosen Kernel functions, you have 0 to 3 additional parameters for tuning.
   - Degree, γ, coef0

linear: $u'v$
polynomial: $(\gamma u'v + coef0)^{degree}$
radial basis: $e^{(} - \gamma|u - v|^2)$
sigmoid: $tanh(\gamma u'v + coef0)$

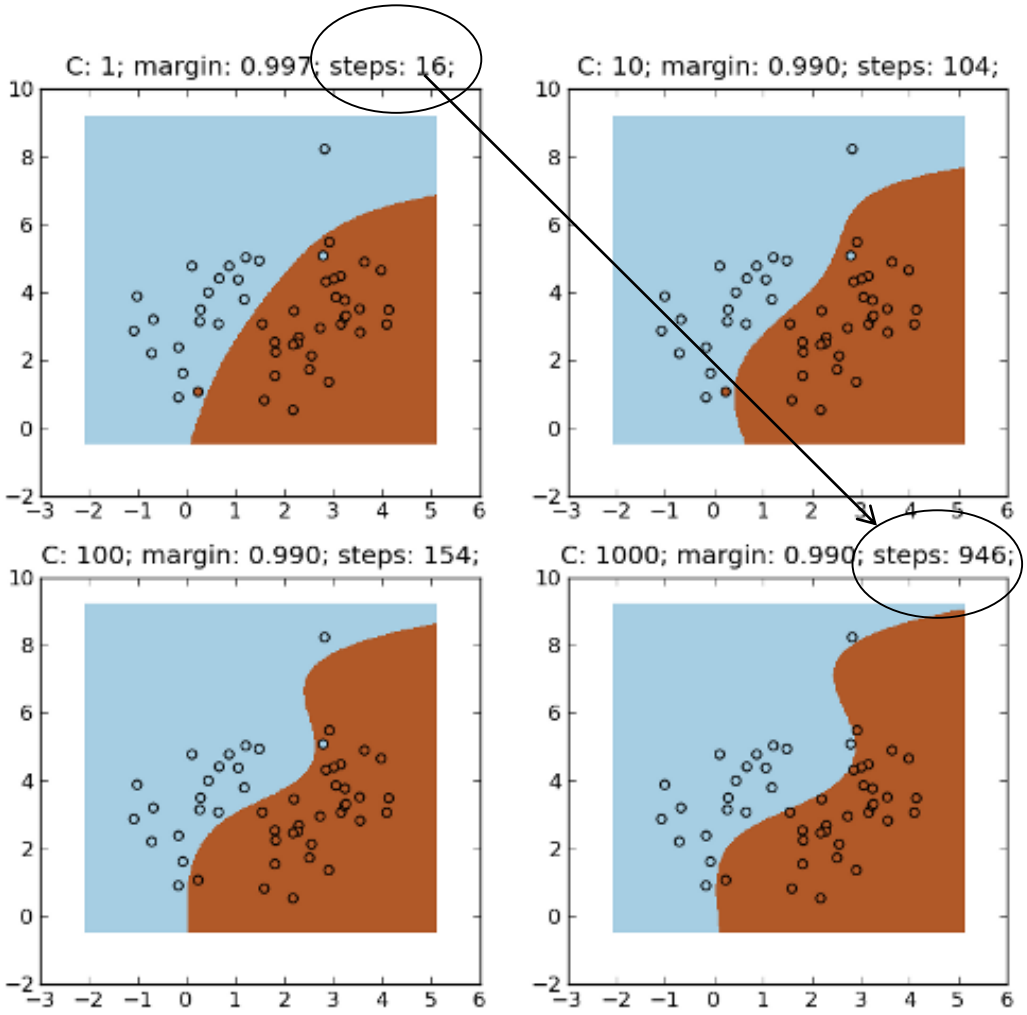
4. Tolerance: Tolerance of termination criterion (default: 0.001), similar to threshold in NN. This is not that important.

1. Tuning of the C parameter Effects

The larger the C value, the heavier the penalty on the mis-classified examples. SVM will try to fit the boundary as much as possible.

(1) Larger C, smaller training error, larger chance to overfit

(2) Larger C, slower, you need more steps to find the optimal boundary for classification

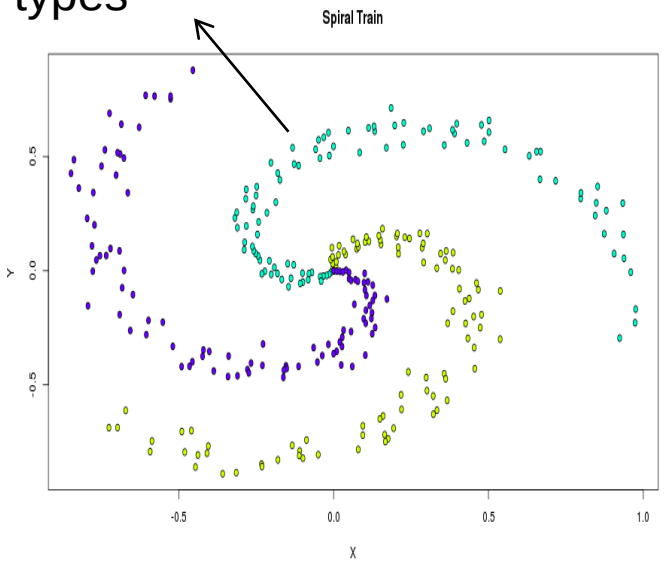


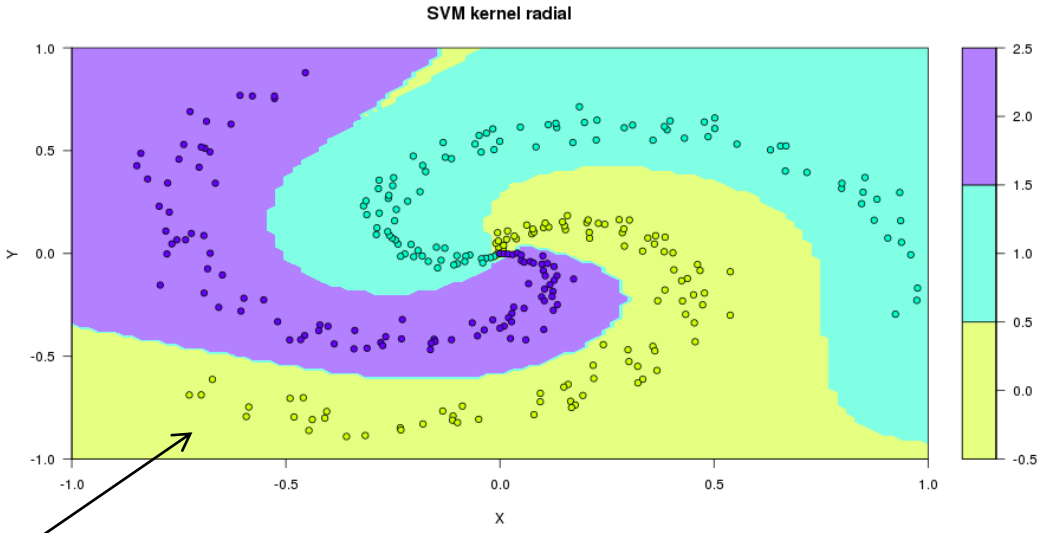
2. Selecting Kernel Functions

- In general and in theory, the best choice is **RBF kernel function for SVM**. This is very different from Neural Network.
  - There are several theoretic reasons to choose RBF over the other 3.
- If you have confidence by domain knowledge that the boundary is linear, then do not use Kernel functions. This case is rare.
- If you have confidence by domain knowledge that the mapping is polynomial, then use this polynomial kernel.
- Tanh is included in many packages only because of its power and popularity in neural network. Theoretical studies suggest RBF is a better choice than Tanh.

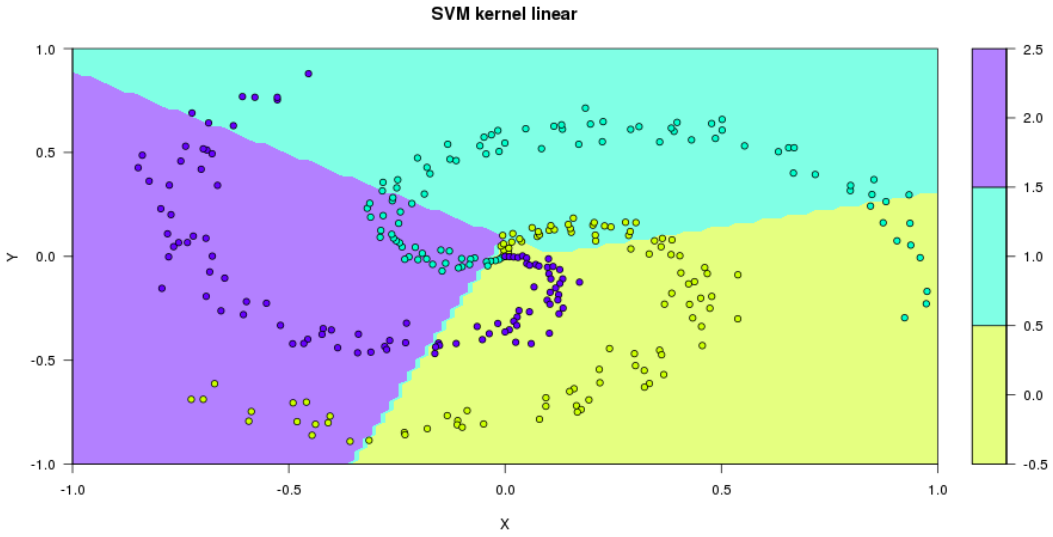
4 Kernel Functions – Spiral Example

3 types

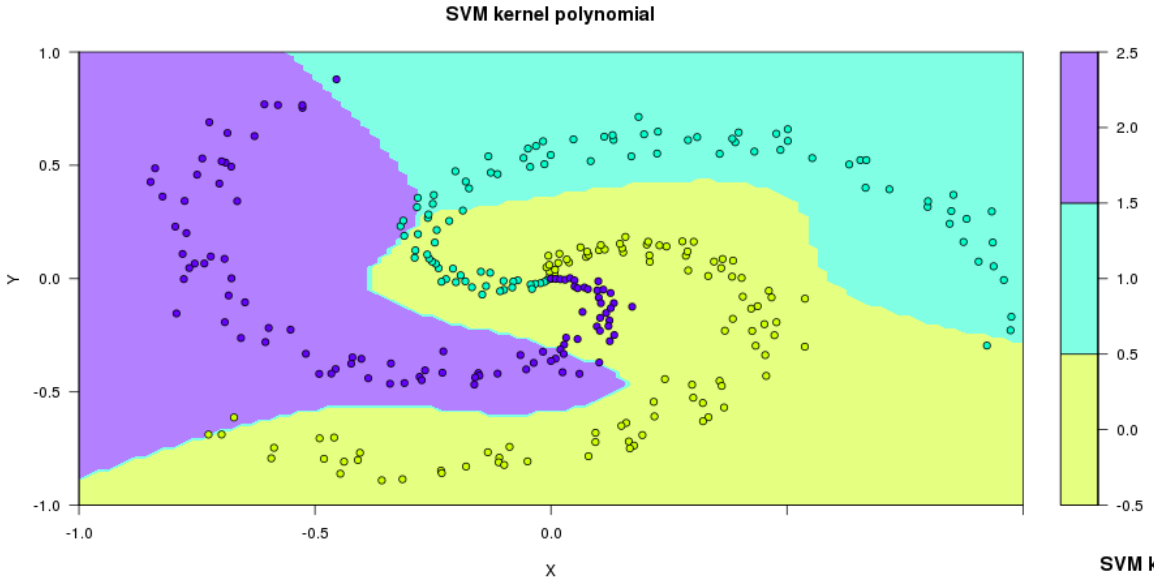


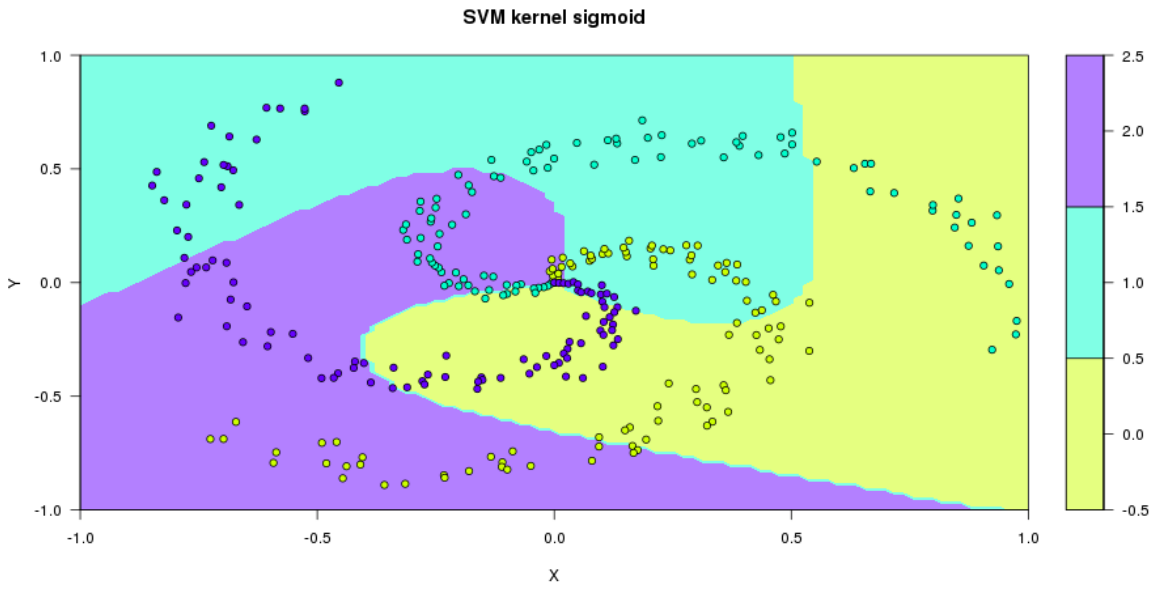


RBF Kernel is used in textbooks (not from our book) as a good example to classify the Spiral Shape easily



4 Kernel Functions – Spiral Example





Radial Basis Kernel for SVM

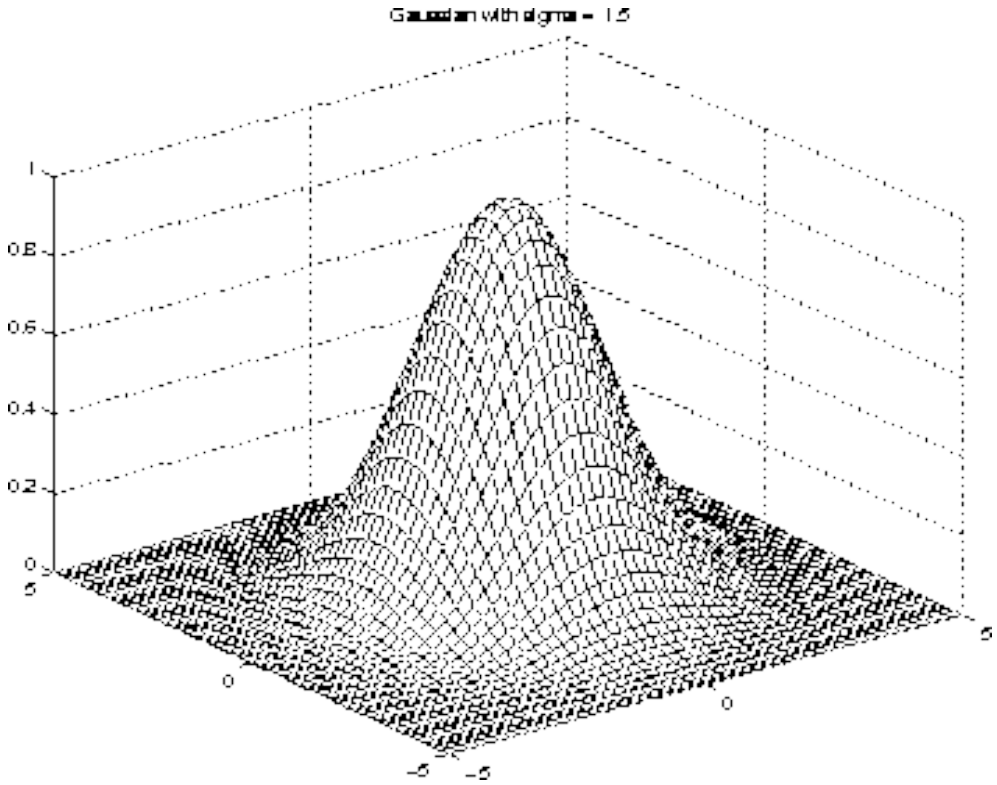
- RBF should be the default Kernel that you start trying.
- Note that SVMs may be very sensitive to the proper choice of parameters, so always check a range of parameter combinations, at least on a reasonable subset of your data.
- Start with a C value between 1 to 1000 and identify a range of good values for C.
- Tune Gamma (γ) of RBF by grid search on Gamma within that range of C to decide the best parameters for C and Gamma.
- RBF with parameters values for C being infinite (very large), Gamma (very large), => getting closer to KNN (K nearest neighbor classifier).

RBF Kernel

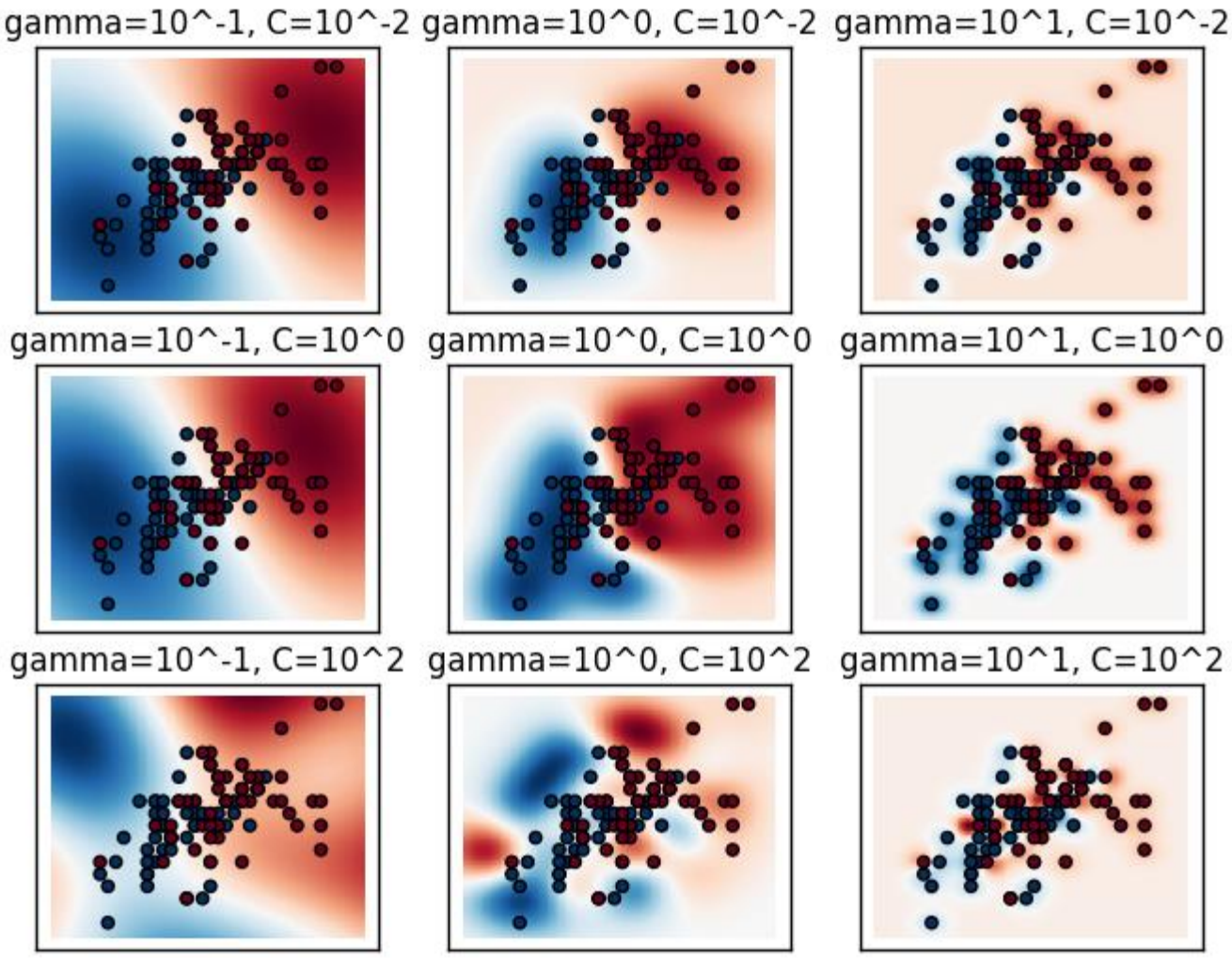
$$K\left(x_i, x_j\right) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right), \gamma \geq 0$$

When two vectors are the same, the distance is 0, the K function is 1.

When two vectors are very different, distance is large, the K function becomes 0.



Example of RBF Tuning



Important Properties of SVM

- SVM boundary is determined only by support vectors ⇔ those few examples that are closest to the classification boundary.
- One important implication is: if we remove any training example that is not on the boundary (not one of the support vector), then results of SVM are still the same.
- So if there are outliers on the correct side, typically they won't be on the boundary and it wont affect the classification at all! (outliers on the wrong side may still affect the boundary)

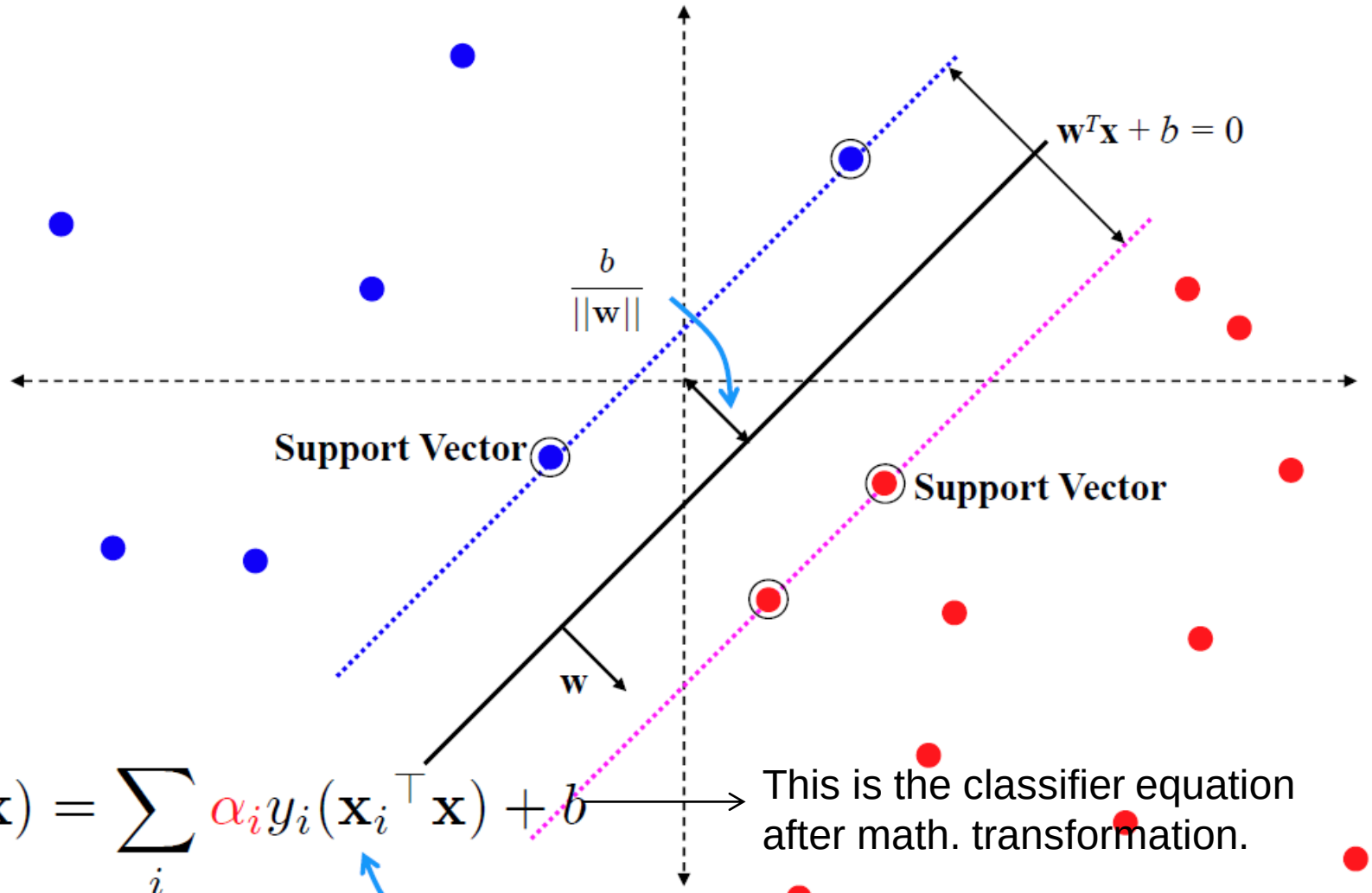
- Another implication is: once we have the classifier, classifying examples in the test set is fast.

Important Properties of SVM (Conti.)

- When we conduct classification, one interpretation of the SVM model is:
  a) We will use the Kernel function to calculate the similarity score between (1) the focal example for prediction, and (2) each of the support vector examples.
  b) We will compute a weighted sum of all positive support vectors' similarity score.
  c) We will compute a weighted sum of all negative support vectors' similarity score.
  d) Comparing (b) and (c) can tell us the example for classification is a positive or negative case.

- Scaling of features for SVM typically also helps performance. As a result, SVM package, such as the one in e1071, may scale features for you by default.

Important Properties of SVM



$$f(\mathbf{x}) = \sum_i \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

This is the classifier equation after math. transformation.

support vectors

Pros and Cons of SVM

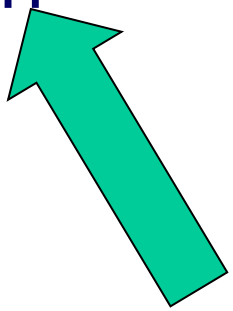
- Advantages
  - Prediction accuracy is generally high
  - Robust, works when training examples contain errors
  - Fast evaluation of the learned target function
  - Global maximum of the solution is guranteed, unlike NN may need to try several times due to backpropagation
  - Strong theoretical research about its foundation
- Criticism
  - Long training time on large dataset in number of rows (roughly n^2 for RBF), linear in number of features for RBF (not bad).
  - Performance depends critically on the choice of C,, Kernel function and parameter(s) of Kernel functions.
  - Difficult to understand the learned function
  - Not easy to incorporate domain knowledge

Today's Class

1. SVM Overview and Linearly Separable SVM
   - Predicting binary label
   - Linearly Separable SVM is unrealistic but good for explaining SVM.

2. Linearly Inseparable: two solutions
   1. SVM with Soft Margin
   2. SVM with Kernel Function

3. SVM Regression

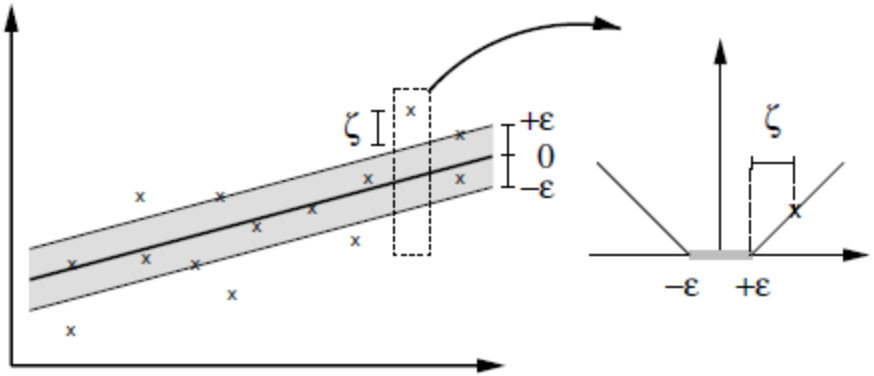


SVM Regression

- The pioneering author of SVM created a version of SVM for regression based on very similar mathematical formulation.
- But the idea and the classifier is very different from the traditional linear regression.
- Let's start with one attribute X and the method can be generalized easily to any number of attributes.
- OLS: Y=b+w*X+e, where *e* is an error term. *b* and *w* are estimated by minimizing sum of squared error.
- Linear SVM regression: the specification is the same, but how to use "error" for estimating *b* and *w* are quite different.

SVM Regression

- The invention is called ε-insensitive loss function.
- That is: if the error is smaller than a threshold, then it is considered as "0" !!!



$$|\xi|_\varepsilon := \begin{cases} 0 & \text{if } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise.} \end{cases}$$

- Also, the algorithm cares about total absolute error, not the sum of squared error (although there are later extensions that use sum of squared errors).

SVM Regression

- The objective function of the optimization problem is given by

$$\min_{b,w} \frac{1}{2}\sqrt{b^2 + w^2} + C\sum_{i=1}^{n}\left(\xi_i + \xi_i^*\right)$$

$$s.t.$$

$$y_i - w \times x_i \leq \varepsilon + \xi_i^*$$

$$w \times x_i - y_i \leq \varepsilon + \xi_i$$

$$\xi_i^*, \xi_i \geq 0 \forall i$$

Sum of absolute errors

A general formulation: l=n

$$\text{minimize} \quad \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*)$$

$$\text{subject to} \begin{cases} y_i - \langle w, x_i\rangle - b \leq \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

- The very last term is the sum of absolute error multiplied by a parameter C.
- All other terms are the same as those in SVM classification. So it implies we are trying to find a separating hyper-plane with maximum band.
- Then this becomes quadratic programming again.

Remarks about SVM Regression

- Here, we start with the explanations by the problem with the "Soft margin" version of SVM.
- Without soft margin, we need to have all observations fall within the band of ε.
- If ε is too large, then the regression line will become the horizontal line that represents the mean of Y.
- If ε is too small, then there is no SVM regression line with a band can include all observations.
- SVM and SVM regression both need you to find the best parameter C, ε, and Kernel to achieve good performance.

SVM Regression with Kernel

- Similar to the SVM classification, Kernel trick is very important.
- Now it may help you understand a bit more about the meaning of mapping to higher dimension.
- Continuing our example of one attribute. You can map it to a quadratic or a cubic polynomial function (or even higher order polynomial function).
- Then we can use a regression with 2 independent variables (or 3 variables) for predicting Y.
- You can use other Kernel function too. Again, the default should be RBF Kernel function.

Appendix: General Math. Formulation

- Our optimization problem is called "Primal" in optimization theory. It can be re-written as

$$\min_{\mathbf{w}\in\mathbb{R}^d} ||\mathbf{w}||^2 + C\sum_{i}^{N} \max\left(0, 1 - y_i f(\mathbf{x}_i)\right)$$

- It can be shown that the Primal Problem is equivalent to its Dual Problem below

$$\min_{\alpha_j} \sum_{jk} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j{}^\top \mathbf{x}_k) \quad \text{subject to } y_i \left(\sum_{j=1}^{N} \alpha_j y_j (\mathbf{x}_j{}^\top \mathbf{x}_i) + b \right) \geq 1, \forall i$$

- The new classifier becomes

$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i (\mathbf{x}_i{}^\top \mathbf{x}) + b$$

General Math. Formulation

$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i (\mathbf{x}_i{}^{\top} \mathbf{x}) + b$$

- α_i and b are the new estimated coefficients of your classifier, like those coefficients in a regression.
- Do note that y_i is -1 or 1 in our case.
- x_i is the given data of attributes.
- x is your focal new record's attribute that you want to classify.
- Each $(x_i^T)x$ conceptually means (cosine) similarity of two vectors.

General Math. Formulation

$$f(\mathbf{x}) = \sum_{i}^{N} \alpha_i y_i (\mathbf{x}_i{}^{\top} \mathbf{x}) + b$$

- Also, note that most α_i are zero in the SVM. This is because α_i is actually "Lagrange Multiplier" and it is zero for non-support vectors and it is positive ($0<=\alpha_i<=C$) for support vectors.
- This equation explains why we only need to know the inner product of $(x_i^T)x$ or the inner product between transformed x.
- Intuitively, for a new observation, x, for classification, we compute the weighted average of "cosine similarity" between x and all support vectors to decide its label should be +1 or -1.

Appendix B: RBF Mapping Is?

RBF Kernel maps each point in 2D space to an infinite vector, $(x, x^2, x^3, x^4 \ldots)$
⇔ using Taylor Expansion to approximate the exponential function.

Let us consider the RBF kernel again for points in $\mathbb{R}^2$. Then, the kernel can be written as

$$k(x, y) = \exp(-\|x - y\|^2) = \exp(-(x_1 - y_1)^2 - (x_2 - y_2)^2)$$

$$= \exp(-x_1^2 + 2x_1y_1 - y_1^2 - x_2^2 + 2x_2y_2 - y_2^2)$$

$$= \exp(-\|x\|^2)\exp(-\|y\|^2)\exp(2x^Ty)$$

(assuming gamma = 1). Using the taylor series you can write this as,

$$k(x, y) = \exp(-\|x\|^2)\exp(-\|y\|^2)\sum_{n=0}^{\infty}\frac{(2x^Ty)^n}{n!}$$

Now, if we were to come up with a feature map Φ just like we did for the polynomial kernel, you would realize that the feature map would map every point in our $\mathbb{R}^2$ to an infinite vector. Thus, RBF implicitly maps every point to an infinite dimensional space.

Appendix C: Cos and Sin Function