Topic 4: Support vector machines concepts and overview of estimation

Support Vector Machines

Introduction and motivation

As seen in earlier topics of this Week (Topic 2, Topic 3), when classifying into one of two p-dimensional multivariate normal populations, the scores are either linear (when the same covariance matrices are used) or quadratic (when the covariance matrices are different). Even optimality for such simple classifiers could be shown due to the multivariate normality assumption.

However, when the two populations are **not** multivariate normal, the situation is more difficult, the bounds between the populations may be more blurry and significantly more non-linear classification techniques may be necessary to achieve a good classification. Support vector machines (SVM) are an example of such non-linear statistical classification techniques.

They usually achieve superior results in comparison to more traditional non-linear parametric classification techniques such as *logit analysis* or non-parametric techniques such as *neural networks*. Mathematically, when using SVM, we try to formulate the classification as an empirical risk minimisation problem and to solve the problem under additional restrictions on the allowed (nonlinear) classifier functions.

Expected versus Empirical Risk minimisation

Let Y be an "indicator" with values $+1$ and -1 that indicate if certain p dimensional observation belongs to one of two groups of interest. We want to find a "best" classifier in a class $\mathcal{F}$ of functions f. Each classifier function $f(\boldsymbol{x})$ is meant to deliver a value of $+1$ or -1 for a given observation vector $\boldsymbol{x}$. To this end, we consider the *expected risk*

$$R(f) = \int \frac{1}{2}|f(\boldsymbol{x}) - y| dP(\boldsymbol{x}, y)$$

Since the joint distribution $P(\boldsymbol{x}, y)$ is unknown in practice, we consider the *empirical risk* over a training set $(\boldsymbol{x}_i, y_i), i = 1, 2, \ldots, n$ of observations instead:

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2}|f(\boldsymbol{x}_i) - y_i|$$

The loss in the risk's definition is the "zero-one loss" given by

$$L(\boldsymbol{x}, y) = \frac{1}{2}|f(\boldsymbol{x}) - y|$$

and, thanks to the chosen labels ± 1 for Y obviously has the values 0 (if classification is correct) and 1 (if classification is wrong).

Minimising the empirical (instead of the unknown expected) risk means to find $f_n = \arg\min_{f \in \mathcal{F}} \hat{R}(f)$ as an approximation to $f_{\text{opt}} = \arg\min_{f \in \mathcal{F}} R(f)$. (Here and elsewhere, $\arg\min_a h(a)$ is that a which minimises the value of $h(a)$.) Generally speaking the two solutions f_n and f_{opt} do not coincide and without further assumptions may be quite different.

Loosely speaking, it can be shown that the maximum difference between the expected risk of a classifier and its empirical risk is a decreasing function of the sample size and an increasing function of the complexity of the class of models $\mathcal{F}$, measured by the *Vapnik–Chervonenkis (VC) dimension* h.

For a linear classification rule $f(\boldsymbol{x}) = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x} + b)$ with a p dimensional predictor $\boldsymbol{x}$, the VC dimension is $h = p + 1$. In general, the VC dimension of a given set of functions is equal to the maximal number of points that can be separated in *all possible ways* by that set of functions. (You are allowed to arrange the points however you want.)

At first glance, the "more rich" the function class $\mathcal{F}$ the better the classification rule would be. Indeed you can construct a classifier that has zero classification error on the training set. However, this

classifier will be too specialised for the given training set with no ability to generalise for other sets. Hence such a classifier would be undesirable. "More rich" is tantamount to require bigger complexity of $\mathcal{F}$ or equivalently higher value of h, and so the possible difference between the expected risk and the empirical risk would grow: the classifier could perform very well on the dataset, but very poorly on the population.

Basic idea of SVMs

A *linear classifier* is one that given feature vector $\boldsymbol{x}_{\text{new}}$ and weights $\boldsymbol{w}$, classifies y_{new} based on the value of $\boldsymbol{w}^\top \boldsymbol{x}_{\text{new}}$; for example,

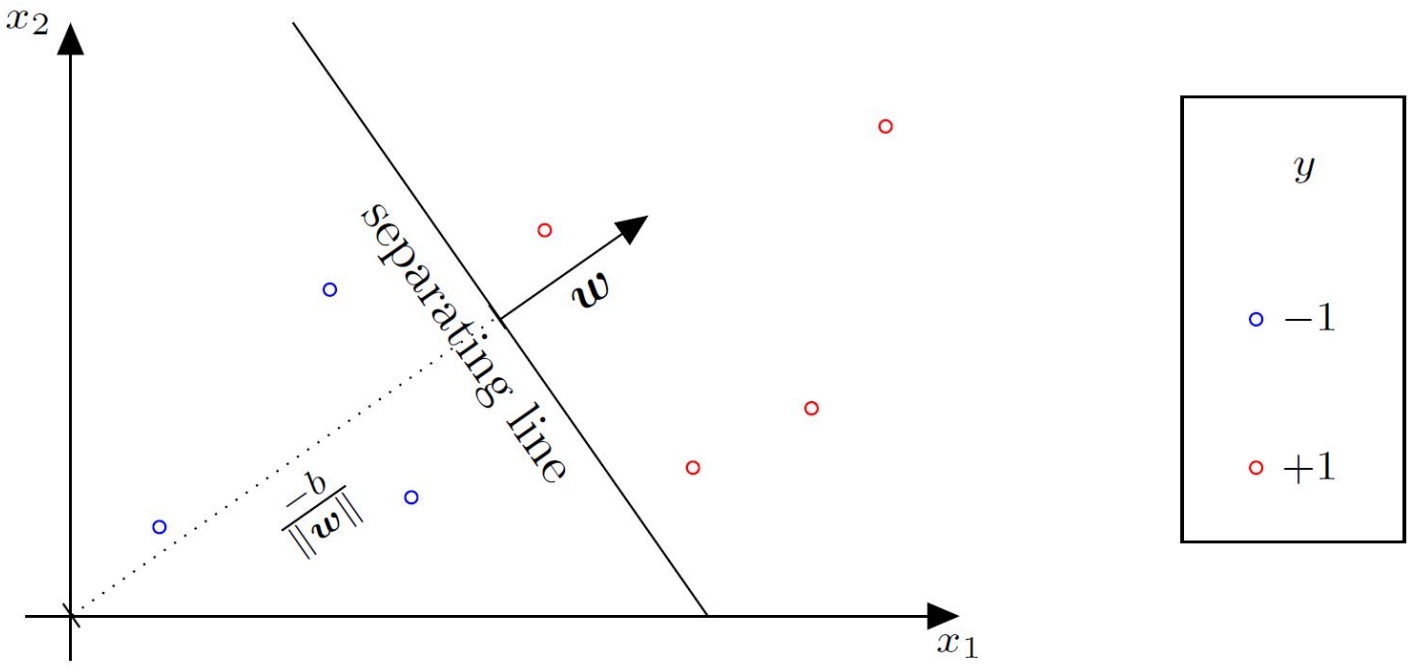
$$\hat{y}_{\text{new}} = \begin{cases} +1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x}_{\text{new}} + b > 0 \\ -1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x}_{\text{new}} + b < 0 \end{cases}$$

for a threshold $-b$. Here, we see that every element of $\boldsymbol{x}$, x_i, gets a weight w_i:

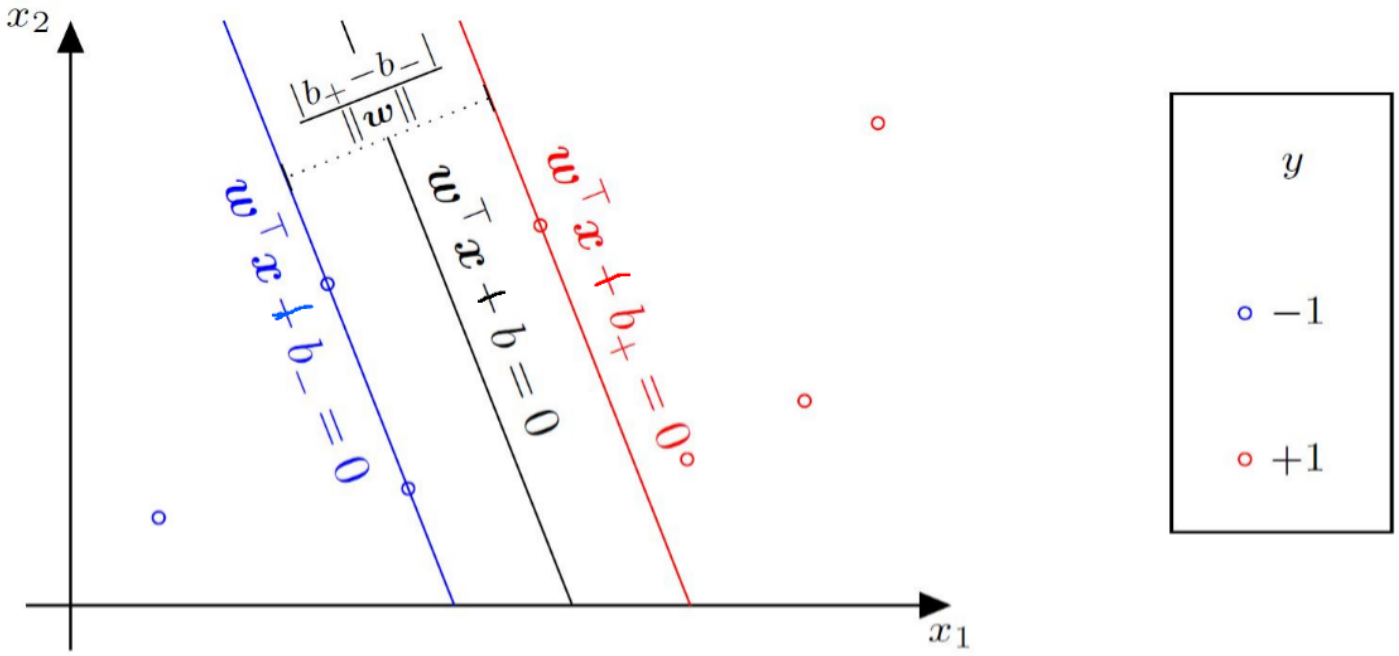
The **Sign** of w_i determines whether increasing x_i pushes the prediction toward $y_i = -1$ or $y_i = +1$.

The **Magnitude** of w_i determines how strongly.

The regions of $\boldsymbol{x}$ for which the model predicts $+1$ as opposed to -1 are defined by $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$. Points $\boldsymbol{x}$ that satisfy that equation exactly form a line (if $d = 2$), a plane (if $d = 3$), or a hyperplane (if $d \geq 3$). We call the data *linearly separable* if a hyperplane that separates them exists. Let us focus on this linearly separable case (and consider the nonseparable case later.) The following diagram illustrates one such line:



Now, usually, there are infinitely many different hyperplanes which could be used to separate a linearly separable dataset. We therefore have to define the "best" one. The "best" choice can be regarded as the middle of the widest empty strip (or higher dimensional analogue) between the two classes, one that maximises the *margin* $\frac{|b_+ - b_-|}{\|\boldsymbol{w}\|}$ in the following illustration:



The scale of $\boldsymbol{w}$ and b is arbitrary: for arbitrary $\alpha \neq 0$, any $\boldsymbol{x}$ that satisfies $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ also satisfies $(\alpha \boldsymbol{w})^\top \boldsymbol{x} + (\alpha b) = \alpha(\boldsymbol{w}^\top \boldsymbol{x} + b) = 0$, so $(\boldsymbol{w}, b)$ and $(\alpha \boldsymbol{w}, \alpha b)$ define the same plane. We fix $|b_+ - b| = |b_- - b| = 1$, and only vary $\boldsymbol{w}$: our "outer" hyperplanes become

$$\boldsymbol{w}^\top \boldsymbol{x} + (b - 1) = 0$$
$$\boldsymbol{w}^\top \boldsymbol{x} - (b - 1) = 0$$

Then, the margin of $\frac{|b_+ - b_-|}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|}$ is maximised by minimising $\|\boldsymbol{w}\|$. Therefore, a *Linear Support Vector Machine* minimises $\|\boldsymbol{w}\|^2$ subject to separating -1s and $+1$s.

Estimation

Linear SVM: Separable Case

We write the boundaries of the empty region as

$$\begin{aligned} \boldsymbol{w}^\top \boldsymbol{x} + (b-1) = 0 &\implies \boldsymbol{w}^\top \boldsymbol{x} + b = +1 \\ \boldsymbol{w}^\top \boldsymbol{x} - (b-1) = 0 &\implies \boldsymbol{w}^\top \boldsymbol{x} + b = -1 \end{aligned}$$

and observe that

$$\hat{y}_i = \begin{cases} +1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x}_i + b > 0 \\ -1 & \text{if } \boldsymbol{w}^\top \boldsymbol{x}_i + b < 0 \end{cases} = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x}_i + b).$$

This means that if $\boldsymbol{w}^\top \boldsymbol{x} + b = 0$ separates -1s and $+1$s (i.e., $y_i = \hat{y}_i$ for all $i = 1, \ldots, n$),

$$y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1.$$

Therefore, a linear SVM learning task for can be expressed as a constrained optimisation problem:

$$\arg\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 \text{ subject to } y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1,\ i = 1, \ldots, n.$$

(Here and elsewhere, $\arg\min_a h(a)$ is that a which minimises the value of $h(a)$.)

This problem is solved using the Lagrange Multiplier technique. We omit the details for brevity, but it is valuable to observe the following intermediate result: we can show that the solution to the minimisation problem will be at

$$\begin{aligned} y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) - 1 \geq 0,\ i = 1, \ldots, n \\ \alpha_i \left(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) - 1\right) = 0,\ i = 1, \ldots, n \end{aligned}$$

for some $\alpha_i \geq 0,\ i = 1, \ldots, n$. Notice that the second equation implies that *either* $\alpha_i = 0$ *or* $y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) = 1$ (or both). But that means that if $\alpha_i \neq 0$, the training instance lies on a corresponding hyperplane and is known as a *support vector*.

It turns out that we can further reexpress this problem as maximising

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j,$$

subject to

$$\alpha_i \geq 0,\ i = 1, \ldots, n$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0.$$

This is a *quadratic programming* problem, for which many software tools are available.

Linear SVM: Nonseparable Case

Of course, in real-world problems, it is not possible to find hyperplanes which perfectly separate the target classes. The *soft margin* approach considers a trade-off between margin width and number of training misclassifications. *Slack* variables $\xi_i \geq 0$ are included in the constraints: we insist that

$$y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i. \tag{5.8}$$

The optimisation then becomes

$$\arg\min_{\boldsymbol{w}} \left(\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \xi_i \right) \text{ subject to } y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i,\ i = 1, \ldots, n,$$

or a tuning constant C. Small C means a lot of slack, whereas a large C means little slack. In particular, if we set $C = \infty$, we require separation to be perfect, a *hard margin.*

Now, taking (5.8) and solving for ξ_i gives $\xi_i \geq 1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)$. We want to make ξ_i as small as possible, so we can get $\xi_i = 1 - y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b)$. After some further algebraic manipulation, we can eliminate these slack variables and end up with the following optimisation problem:

$$\arg\max_{\boldsymbol{\alpha}} \left(\sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n} \alpha_j \alpha_k y_j y_k (\boldsymbol{x}_j^\top \boldsymbol{x}_k) \right)$$
$$\text{subject to } \sum_{i=1}^{n} \alpha_i y_i = 0 \text{ and } 0 \leq \alpha_i \leq C,\ i = 1, \ldots, n.$$

Prediction

We can also express the prediction in two ways:

$$\text{Primal: } \hat{y}(x) = \text{sign}\left(\boldsymbol{w}^\top \boldsymbol{x} + b\right) \tag{5.9}$$

$$\text{Dual: } \hat{y}(x) = \text{sign}\left\{ \sum_{j=1}^{n} \alpha_j y_j \left(\boldsymbol{x}_j^\top \boldsymbol{x}\right) + b \right\} \tag{5.10}$$

Primal ($\boldsymbol{w}$) form requires d parameters, while *dual* ($\boldsymbol{\alpha}$) form requires n parameters. This means that for high-dimensional problems — those with $d \gg n$, a huge number of predictors, the dual

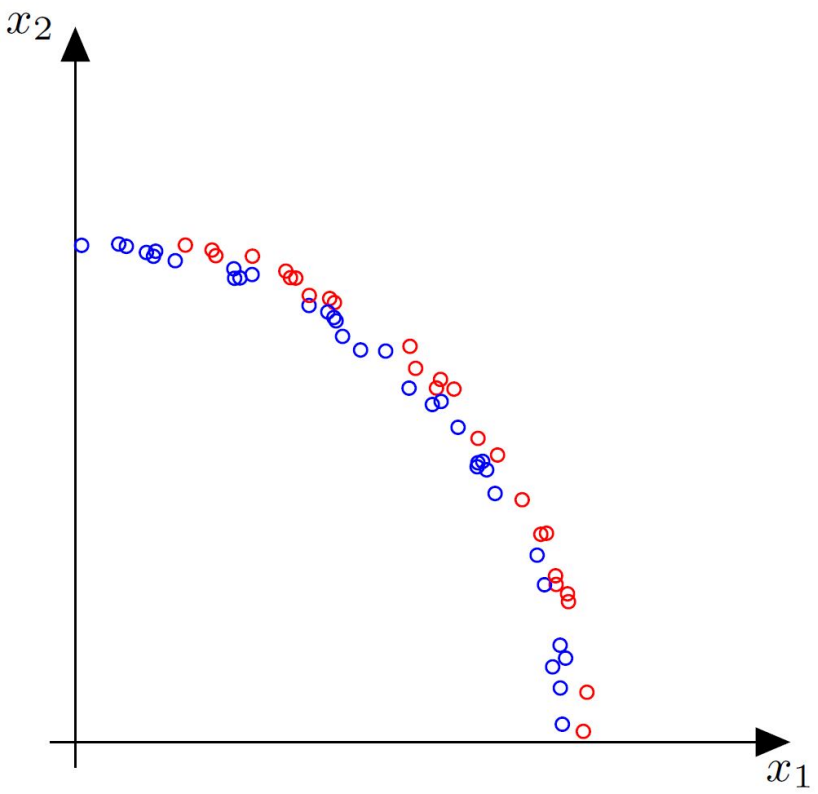
representation can be more efficient.

But it gets better! Notice that only the $\boldsymbol{x}_i$s closest to the separating hyperplane — those with $\alpha_j > 0$ — matter in determining $\hat{y}(\boldsymbol{x})$, so most of them will have no effect. Thus, computationally, effective "n" will actually be much smaller than the sample size, so the above condition can be met far more often than one might expect. Again, those $\boldsymbol{x}_i$s that "support" the hyperplane are called *support vectors*.

In addition, notice that the dual form only depends on $(\boldsymbol{x}_j^\top \boldsymbol{x}_k)$s. This opens the door to nonlinear SVMs.

Nonlinear SVMs

Consider the following data:



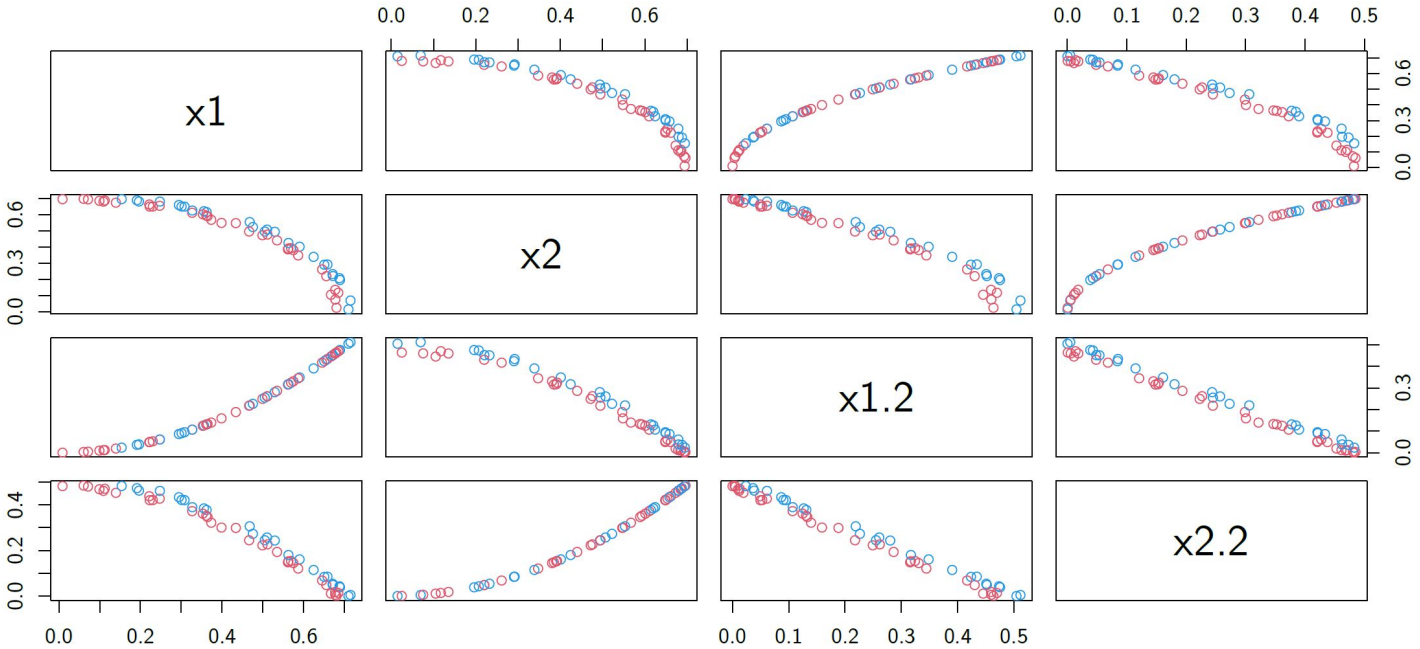
The true classification for these points is

$$y = \begin{cases} +1 & x_1^2 + x_2^2 < 0.75^2 \\ -1 & x_1^2 + x_2^2 > 0.75^2 \end{cases},$$

but one can hardly draw a line separating them.

What we can do is transform $\boldsymbol{x}$ so that a linear decision boundary can separate them. In this case, suppose we augmented our $\boldsymbol{x}$ with squared terms:

$$(x_1, x_2) \to (x_1, x_2, x_1^2, x_2^2) :$$



Now, a linear separator exists!

Now, recall that the dual form (5.10) depends only on dot products $\boldsymbol{x}_i^\top \boldsymbol{x}_j$. However, we can specify other *kernels* $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. For example, a "kernel" function of the form $k(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^\top \boldsymbol{v} + 1)^2$ can be regarded as a dot product

$$u_1^2 v_1^2 + u_2 v_2^2 + 2u_1 v_1 + 2u_2 v_2 + 1$$

$$= (u_1^2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2, 1)^\top (v_1^2, v_2^2, \sqrt{2}v_1, \sqrt{2}v_2, 1).$$

which reconstructs the above augmentation. In general, kernel functions can be expressed in terms of high dimensional dot products. Computing dot products via kernel functions is computationally cheaper" than using transformed attributes directly.

A common type of kernel is a *radial basis function*: a function of distance from the origin, or from another fixed point $\boldsymbol{v}$. Usually, the distance is *Euclidean, i.e.*

$$\|\boldsymbol{u} - \boldsymbol{v}\| = \sqrt{(u_1 - v_1)^2 + \cdots + (u_n - v_n)^2}.$$

A common radial basis function is *Gaussian*:

$$\phi(\boldsymbol{u}, \boldsymbol{v}) = \exp\left(-\gamma \|\boldsymbol{u} - \boldsymbol{v}\|^2\right).$$

We can use $\phi(\cdot, \cdot)$ as our SVM kernel.

Multiple classes

Finally, we briefly consider the problem when there are more than two classes. Suppose that there are $K > 2$ categories.

Recall that $\boldsymbol{w}^\top \boldsymbol{x}_i$ gives us a "score" that we normally compare to b. However, we do not have to do so. Instead, for each $k = 1, \ldots, K$, we can fit a separate SVM (i.e., $\boldsymbol{w}_k$ and b_k) for whether an observation is in k vs. not. We can then predict $\hat{y}_{\text{new}}$ by evaluating $\boldsymbol{w}_k^\top \boldsymbol{x}_{\text{new}} + b_k$ for each k and taking highest biggest one. This is called the *One-against-rest* approach.

A computationally more expensive approach that tends to perform better is the *One-against-one*: an SVM is fit for every distinct pair $k_1, k_2 = 1, \ldots, K$, fit an SVM for k_1 vs. k_2, and predict the "winner" of all the rounds (if any). This requires fitting $K(K-1)/2$ binary classifiers, but to smaller datasets.