

KERNELS, GEOMETRY AND CLASSIFICATION

Doctor SVN Vishwanathan

April 29, 2008

1 Introduction

Last time, we discussed exponential families, probability estimation, and how starting from a probability estimation can lead to a classification algorithm. Now, we go the other way: how geometry can lead to building classifiers. And then, we learn that they are the one and the same—the same classifier can be created by these two methods.

In our general philosophy, we consider what we want to actually do with machine learning. We have some data, where x refers to data points and y to labels:

$$\{(x_1, y_1) \dots (x_m, y_m)\} \tag{1}$$

Given new x values, we look for an algorithm to return a y . Supposing: $x \in \chi$ and $y \in \varphi$, we search for an f :

$$f : \chi \Rightarrow \varphi \tag{2}$$

Deliberately, no assumptions have been made just yet regarding x and y . Since $f(x)$ will predict label y , f needs to be defined.

$$f(x_i) = y_i$$

Ensure that the function actually respects that, though don't enforce it strictly.

We look at the dominant branch of problems, and an easy class of problems; other machine learning branches deal with unsupervised learning.

$$P : \chi \Rightarrow \varphi$$

$$f(x_i) = y_i$$

In this case: use domain $\varphi \in \{+1, -1\}$. How do we learn the right function to distinguish labels -1 and +1 for different data? Deliberately, we start here

making no assumptions on x (“ x is Euclidean”, etc). So, how am I to find the function which respects the existing data points and goes to other data points?

We do, though, make the reasonable assumption that there is some similarity within the data points and labels, ie.

$$x \sim x_j \Rightarrow f(x_i) \sim f(x_j)$$

This assumption rests upon the key point: how will you define similarities? The answer is very domain-dependent. Given any (x, x') , there exists a function k that takes (x, x') and returns a number.

$$k : (x * x) \Rightarrow \mathbb{R}$$

We assume the property that k is large if x_i and x_j are similar, and low otherwise.

Presently, we have a black box and two datapoints, and expect it to return a number. Why is this level of abstraction important? Why not just use vectors? Because there are many instances where representing input data as vectors is not a good idea.

Example: whether an email is spam, or “ham”—a normal email. Emails are sequences of characters. What function can take two emails and say whether they are similar?

$$k : (x * x) \Rightarrow \mathbb{R} \tag{3}$$

This is the philosophy of kernels. We don’t care about data type, just similarity.

So what kind of function could have that kind of property? We could define many—and many may not have nice mathematical properties. Function space analysis is a wide area. Impose a simple requirement: look at functions, defined as follows:

$$\phi : X \Rightarrow \mathbb{R}^n$$

Define the function to be:

$$k : (x * x) = \langle \phi(x), \phi(x') \rangle$$

Simple example:

$$k(x, x') = \langle x, x \rangle$$

$$k(x, x') = \sum_{i=1}^n x_i \cdot x'_i$$

What do we know about the dot product? There is a lot of geometry in a dot product. It is the sum of the angle between two vectors. The length of x will be $x \cdot x = \sqrt{x}$. The dot product can also represent distances. The third property of the dot product: Suppose that you want to look at $\|x - x'\|$. This is:

$$\sqrt{\langle x - x', x - x' \rangle}$$

$$k(x, x') = \langle \phi(x), \phi(x') \rangle \tag{4}$$

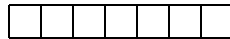
Any algorithm using these notions can work with a generalized version of the dot product.

A question, keeping matters general: how can we specify ϕ ? One way would be to come up with an explicit function. A popular way to represent a document is to represent all words in that bunch of documents, and have a dictionary of words; sorting the words in the dictionary, each word could get assigned a number. The words in the document may be represented as long vectors.



The document consists of words; for each word in the document, increment a frequency score. The bag-of-words model has a document represented by a long vector.

For another map: take a string, a horizontal vector.



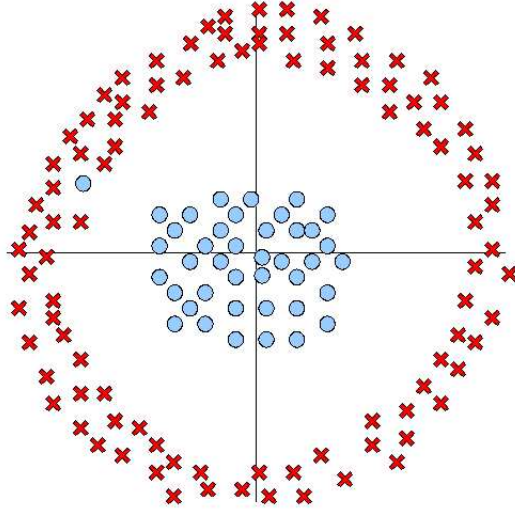
Look at all possible substrings within the string, and create a map. Find out how many times did these substrings occur? (Of course, a near-infinite set of all possible substrings is difficult to represent.)

The key is that you do not need ϕ . The kernel algorithm will never ask for the ϕ element, just the dot product, $\langle \phi(x), \phi(x') \rangle$ (4).

Suppose the data is in a Euclidean space. Why, then, do a mapping in ϕ ? If the data is already Euclidean, why think of a ϕ ?

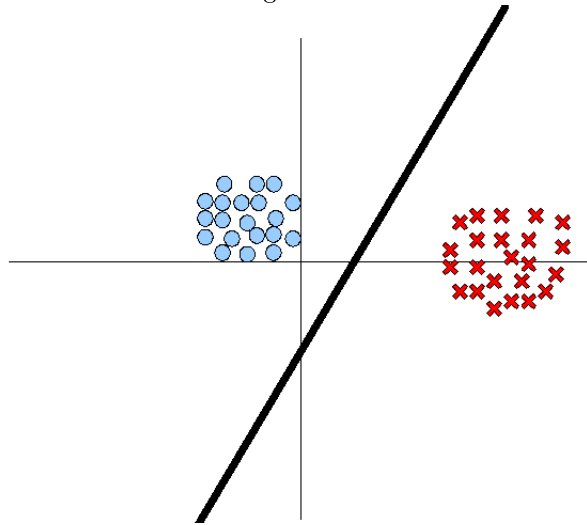
For example: a binary class where we seek to do a classification.

Figure 1: classifying dots and crosses



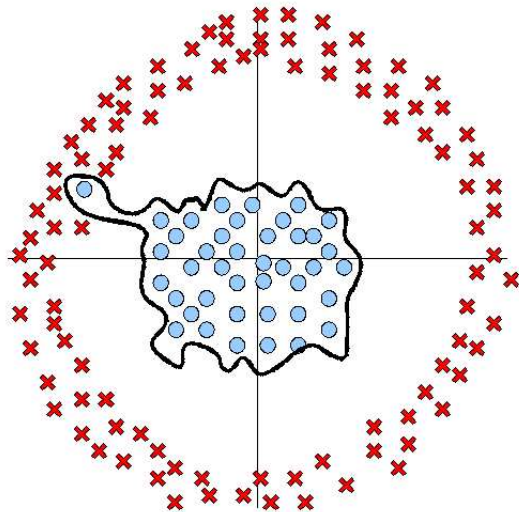
We want an ellipsoid decision boundary. We can possibly look at the points as vectors, and get an image more like the below, with a linear decision boundary.

Figure 2: linear decision boundary



In general, intuition tells us: if the boundary is a k -order polynomial, even an ugly one like this:

Figure 3: ugly k -order classification polynomial

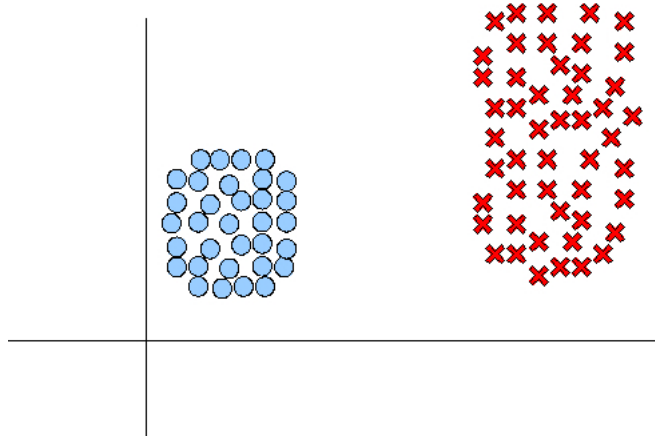


We can transform it into a plane.

The ugly polynomial shows that, instead of learning, we have only mapped the data into a high-dimensional space (see figure 3). This is also part of the kernel idea: we do not want complicated functions when simple functions can work.

Can we design a simple classifier for 'dots' (-1) and 'crosses' (+1), in this arrangement?

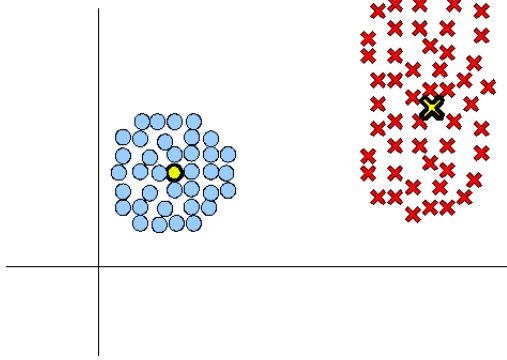
Figure 4: classification problem



Nearest neighbour: given a point, we use its nearest neighbours. The problem is that nearest neighbour is a lazy algorithm. What if we instead classified based

on the nearest mean?

Figure 5: nearest mean classification



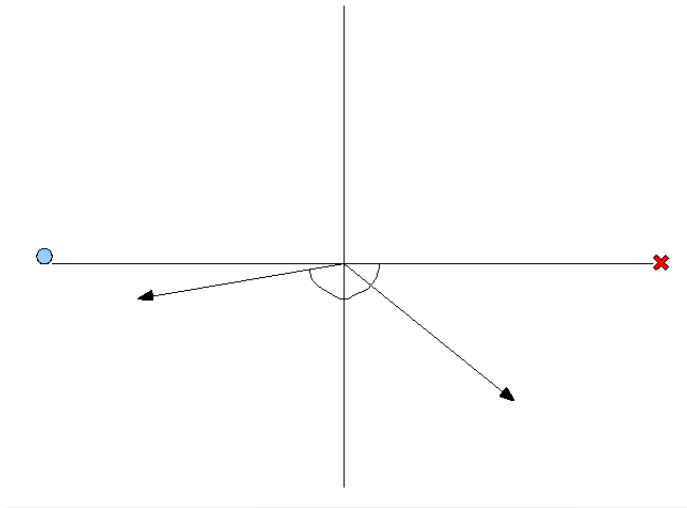
Given a decision point:

$$\text{if } \|x - c_+\| \leq \|x - c_-\| \Rightarrow x' \cdot y = 1$$

$$\text{else } x' \cdot y = -1$$

One geometric property of vectors: given two vectors, how do you characterise the region of all points closer to the dot or closer to the cross? There is an easy way to find this. It is a simple geometric expression.

Figure 6: vector expression



The angle of the vector determines the result. We could take the sine function of the angle and form a c point: a geometric classification.

$$w = C_+ - C_-$$

This is the other vector of interest.

$$C = \frac{C_+ + C_-}{2}$$

We normalize:

$$\langle x - C, w \rangle \geq 0$$

$$f(x) = \text{sign of } \langle x - C, w \rangle$$

$$f(x) = \text{sgn}(\langle x - \frac{C_+ + C_-}{2}, C_+ - C_- \rangle)$$

Now, we perform a little manipulation:

$$f(x) = \text{sgn}(\langle x, C_+ \rangle - \langle x, C_- \rangle + \langle \frac{C_+ + C_-}{2}, C_+ + C_- \rangle)$$

$$f(x) = \text{sgn}(\langle x, C_+ \rangle - \langle x, C_- \rangle + b)$$

$$b = \frac{1}{2} \|C_-\|^2 - \|C_+\|^2$$

The decision boundary can be represented by dot products. How do we get to them? By using the linearity of the product.

$$\langle x, C_+ \rangle = \frac{1}{m_+} \sum_{y_i=1}^i \langle x, x_i \rangle$$

How can you write the decision function?

$$f(x) = \text{sgn} \left(\frac{1}{m_+} \sum_{i_+} \langle x_i, x \rangle - \frac{1}{m_-} \sum_{i_-} \langle x_i, x \rangle + b \right) \quad (5)$$

It is a black box as long as the first summation returns an equation. We can assume that x is the mapped version of x , as long as the kernel function satisfies the second summation. We can write it as a kernel function:

$$f(x) = \text{sgn} \left(\frac{1}{m_+} \sum_{i_+} k(i_+, x) - \frac{1}{m_-} \sum_{i_-} k(x_{i_-}, x) + b \right) \quad (6)$$

That is the kernel-generation function.

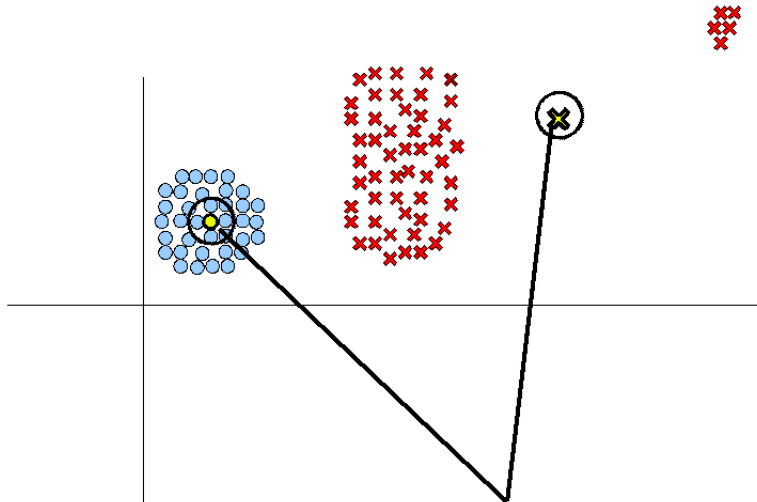
$$f(x) = \text{sgn} \left(\sum_i \alpha_i y_i k(x_i, x) + b \right) \quad (7)$$

(The *sgn* is a hyperplane: it goes well with the intuition that there exists a single decision boundary. See 6.)

We have used geometry to form an algorithm to classify points. What is very nice about this algorithm? That it is simple. We also have a predefined rule.

Summing over i for a lot of data points can be bad. And if we had an outlier point, that could throw off the mean; MLEs are bad because of outliers.

Figure 7: outlier example

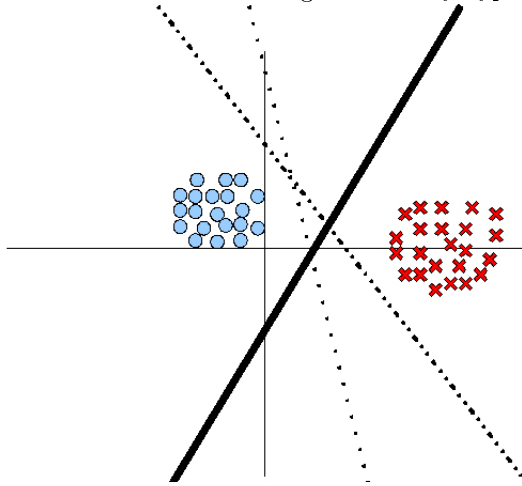


Doing summations over all points can be very expensive.
Decision functions of this form are easy.

$$f(x) = \text{sgn}(\langle w, x \rangle + b)$$

Now, assume there is a plane where all the circle and cross points are clearly separated on a linear plane (2). Of course, there are infinitely many hyperplanes you can use for the labelling. Which should you choose?

Figure 8: many hyperplanes



One possible way to resolve the ambiguity is to choose a plane that maximally separates the data clusters. We use geometry to find the minimum difference looking at the orthogonal projection of the data points onto the plane:

$$\langle w, x \rangle + b = 0$$

We would say: “I want to maximise the separation between data points—the margin should be the closest point to the plan, such that”:

$$\max_{w,b} \min\{\|x - x_i\| : \langle w, x \rangle + b = 0 \text{ } (i=1..m)\} \quad (8)$$

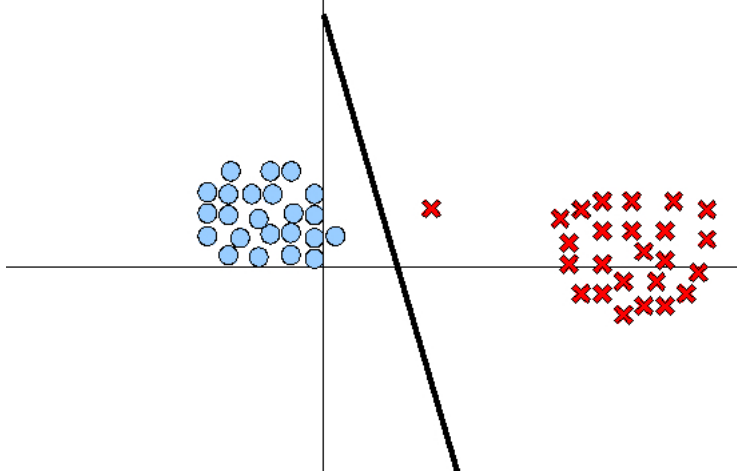
Maximise/minimise with regard to w and b . This is the problem that SVMs (support vector machines) can solve.

What will change the plane in 8 and 8? Which data points will *not* have an effect on the plane? The ones that are not close to the decision boundary.

In the classifier model, one single point in the dataset can change the decision boundary. Now, in the max-min problem, some points can be removed.

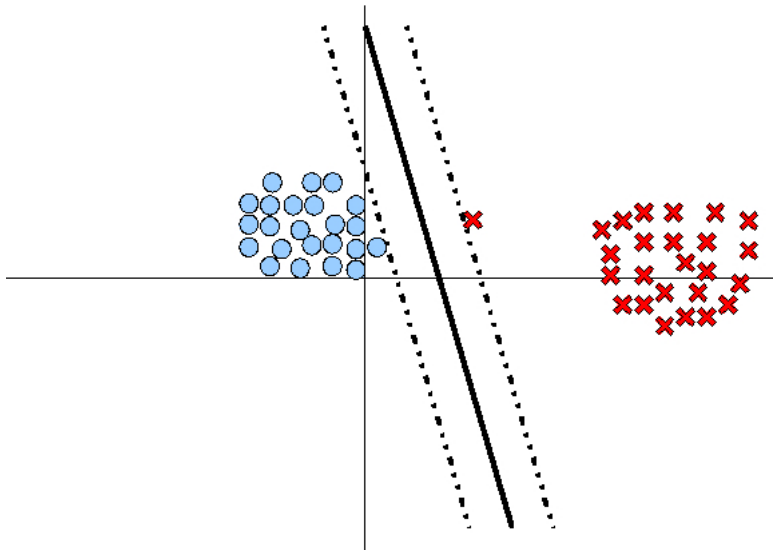
How do we find the plane? One can say, “Let me find a place equidistant from the two closest points from each cluster, maximising the margin of sparation.”.

Figure 9: the margin of separation



There must therefore also be two other planes, running parallel to each other and equidistant from the first plane.

Figure 10: two more equidistant planes



$$\langle w, x_i \rangle + b \geq 0$$

We can say $\langle w, x \rangle + b = a$ is offset plane 2, and $\langle w, x \rangle + b = -a$ is offset plane 1. The a and b parameters can be anything, and are usually set to

1. The reason is that the scaling will be absorbed. By absorbing the value in w , the solution can be brought back to form. (There is no special reason why 1 is chosen here.)

How do you find the margin of separation between the offset planes? If x is a point on the positive plane:

$$\langle w, x_1 \rangle + b = +1$$

Otherwise:

$$\langle w, x_1 \rangle + b = -1$$

If there are no datapoints lying on the positive plane, one could shuffle if more.

$$w \langle x_1, x_2 \rangle = 2$$

$$\left\langle \frac{w}{\|w\|}, x_1 - x_2 \right\rangle = \frac{2}{\|w\|}$$

If you want to maximize the margin:

$$\max \left\langle \frac{w}{\|w\|}, x_1 - x_2 \right\rangle$$

Use an ugly form, so that you can minimize.

$$\min \frac{1}{2} \|w\|^2$$

Maximize the margin of separation and ensure all labelled data points are on the right side (ie. red crosses are positive, blue dots negative. See 9).

2 Putting things together

There is an optimisation problem to solve.

$$\min \frac{1}{2} \|w\|^2$$

It is subject to the constraints:

$$y_i(\langle x_i, w \rangle + b) \geq 1$$

What can we do with the optimisation problem?

How do we know when an optimisation is convex? An optimisation problem is *always* convex.

$$J(a, b) = \min_{a, b} \frac{1}{2} \|w\|^2 \text{ subject to } y_i(\langle x_i, w \rangle + b) \geq 1$$

(Note: α is always positive.)

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum \alpha_i (y_i(\langle x_i, w \rangle + b) - 1) \quad (9)$$

For every constraint, introduce an extra value α_i , and you can solve your constraints; it is a standard trick. (9 is a Lagrangian function). You can show that solving a convex minimization problem is equal to finding a subset of Lagrangian.

$$\min_{w,b} \max_{\alpha} \mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum \alpha_i (y_i(\langle x_i, w \rangle + b) - 1)$$

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

$$\text{subject to } y_i(\langle x_i, w \rangle + b) \geq 1$$

We previously discussed minimizing a convex function: taking gradients with regard to w and b of the Lagrangian function, and setting to zero.

$$\delta_w \mathcal{L} = 0$$

$$\delta_b \mathcal{L} = 0$$

If the problem is feasible, where at least one w satisfies constraints, then it does not go to infinity.

Assume it is finite. Suppose that there are some w who fit the Lagrange constraints.

$$y_i(\langle x_i, w \rangle + b) < 1$$

If this constraint is violated, then:

$$y_i \langle x_i, w \rangle + b - 1$$

If this is less than 0, α can be pumped up.

$$\sum \alpha_i (y_i(\langle x_i, w \rangle + b) - 1)$$

As the α sum decreases, the Lagrangian goes up. It cannot happen that the constraint ($y_i(\langle x_i, w \rangle + b) - 1$) is violated, or that $\alpha \leq 0$. Either the constraint $y_i(\langle x_i, w \rangle + b) \leq 1$ is met, or α is 0—and you can use either.

Now, how do we set the Lagrangian?

Set:

$$\delta_b \mathcal{L} = 0$$

$$\sum \alpha_i y_i = 0$$

Or set:

$$\delta_w \mathcal{L} = 0$$

$$w = \sum \alpha_i y_i x_i$$

The first thing to notice: $w = \sum \alpha_i y_i x_i$. We can do a linear combination of this: we can take three numbers and multiply them to get the solution.

$$f(x) = \text{sgn} (\langle w, x \rangle + b)$$

Therefore:

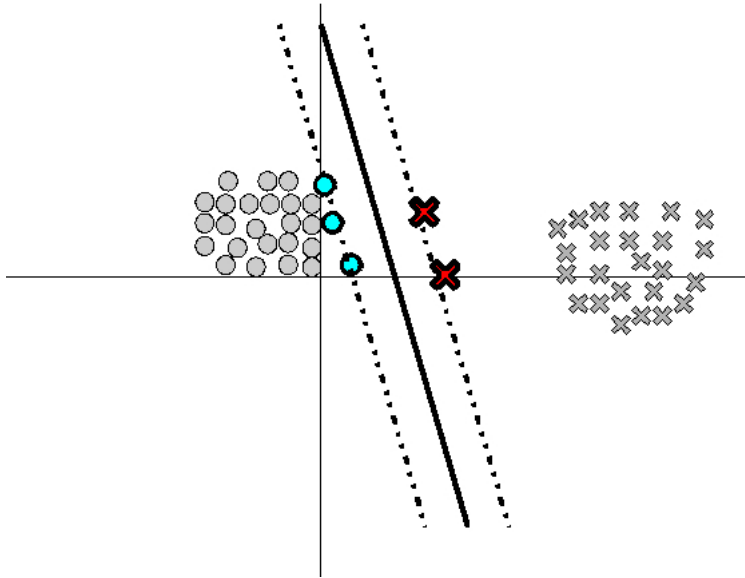
$$f(x) = \text{sgn} (\sum \alpha_i y_i \langle x_i, x \rangle + b)$$

The key difference is the conditions.

$$\alpha_i (y_i (\langle x, w \rangle + b) - 1) = 0, \forall i$$

Only related points contribute to the planes: only they have the condition above.

Figure 11: only some points contribute to the planes



Since only a few points are used, outlying points will not cause problems as they do when the mean is used, but outliers could still be a problem. (See fig. 12.)

But how do you actually solve the optimisation problem? Take:

$$\delta_w \mathcal{L} = 0, \quad w = \sum \alpha_i y_i x_i$$

$$\delta_b \mathcal{L} = 0, \quad \sum \alpha_i y_i = 0$$

Plug these back into the Lagrange function (which you should work through yourself at least one), and the result is:

$$\sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, y_i \rangle$$

$$\text{subject to } \sum_{i=1}^m \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0$$

(If you took x and mapped it to an infinite-dimensional vector space, the optimization problem would be infinite.)

The fundamental theory is that transforming into dual gives an optimization problem with one result for every datapoint. Also, this:

$$w = \sum_{i=1}^m \alpha_i y_i x_i; \quad \delta_w \mathcal{L} = 0$$

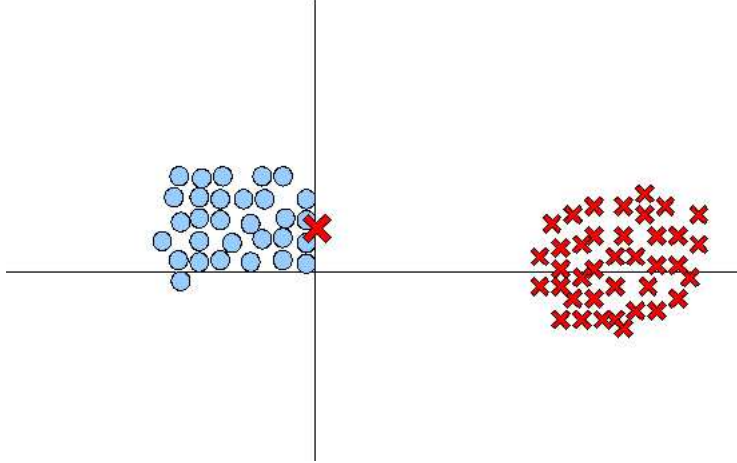
means that you can use an m -dimensional space. The solution will always lie in the span of the data points; the action only happens in a m -dimensional subspace. This is important to note.

The representer theorem optimizes in any m -dimensional space as long as you have the above equations.

3 When data is not linearly separable

We have assumed the data is linearly separable. Now we assume that it is not—and why geometry had probabilistic implications that relate to the previous lecture. In the past, people used probabilistic models and support vector machines without realizing that they were the same thing.

Figure 12: outlier problem



Here we have a single outlier that would make the margin of separation very small. That is not desirable. To handle the problem:

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

$$\text{subject to } y_i(\langle x_i, w \rangle + b) \geq 1 - \epsilon_i$$

We cut slack: error margin ϵ_i . To prevent ourselves from going to infinity, we penalize:

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum \epsilon_i$$

$$\text{subject to } y_i(\langle x_i, w \rangle + b) \geq 1 - \epsilon_i$$

This balances between points for a good division. We cap the outliers' influence: no single point can have too much of a high influence.

This is a standard quadratic problem:

$$\sum_{i=1}^n \alpha_i = \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle x_i, y_i \rangle$$

$$\text{subject to } \sum_{i=1}^m \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0$$

3.1 Quadratic Programming (QP)

Given the above condition, and using column vectors of x :

$$\alpha^T * \chi^T \alpha - \alpha^T e$$

Subject to:

$$C \geq \alpha_i \geq 0$$

$$\alpha^T y = 0$$

The typical QP solver from Matlab/Python/Scipy can only handle problems with a few hundreds or thousands of variables. But we want much more. If the dataset has a million points, $Q(p)$ has a million variables. One popular way of solving $Q(p)$ is SMO, small sequential minimization by optimization.

SMO picks two variables at one point in time, restricts the optimization problem to those two, and solves them to optimality. A SMO needs two variables for the $\alpha^T y = 0$ constraint.

First, choose two α_i s to optimize.

$$\alpha + (C_i e_i + C_j e_j)$$

(Two constraints, updating α .)

C_i and e_i must be equal. The two coefficients are optimised. The LibSVM and Simple SVM methods are used.

$$\alpha^T x^T x \alpha - \alpha^T e$$

$$\text{subject to } \alpha_i y = 0 \text{ and } C \geq e_i \geq 0$$

The simple SVM selects a subset of the α s at any given point in time, solves it without constraints, finds the direction of descent, and takes a step insofar as it doesn't affect one of the constraints.

When working with conditional models: $P(y|x, \theta)$.

Given an x , for example, an identification of handwritten digits. What is a good θ requirement? Find a θ such that given x_i, y_i, θ , the probability assigned to y_i is higher. The design parameter is a legal thing to ask: the true classification and a high probability rate.

$$\frac{P(y_i, x\theta)}{P(y|x_i, \theta)} \geq 1, \forall y$$

$$\frac{\log P(y_i, x\theta)}{\max_{y \neq y_i} P(y|x_i, \theta)} \geq 0$$

The conditional aspects are still satisfied when two probabilities have a tie, for example, weighting same for 0 and 8. That is not desirable; you might want to push the second highest class label down. To do that, increase 0.

$$\frac{\log P(y_i, x\theta)}{\max_{y \neq y_i} P(y|x_i, \theta)} \geq 1$$

$$\log \frac{\exp(\langle \phi(x_i, y_i), \theta \rangle - g(\theta|x_i))}{\max_{y \neq y_i} \exp(\langle \phi(x_i, y_i), \theta \rangle - g(\theta|x_i))}$$

$$\langle \theta(x_i, y_i), \phi \rangle = \max_{y \neq y_i} \langle \phi(x_i, y_i), \theta \rangle \geq 1 \quad (10)$$

You will use a prior, for example in the Gaussian.

$$\min_{\theta} \frac{1}{2} \|\theta\|^2$$

$$\text{subject to } \langle \phi(x, y), \theta \rangle - \max_{y \neq y_i} \langle \phi(x, y), \theta \rangle \geq 1$$

If you solve this:

$$y = \{\pm 1\}$$

$$\phi(x, y) = \frac{y}{2} \phi(x)$$

If we assume this:

$$\frac{y_i}{2} \langle \phi(x_i), \theta \rangle - (-\frac{y_i}{2}) \langle \phi(x_i), \theta \rangle \geq 1$$

(noting that y is -1 or +1)

Then:

$$y_i \langle \phi(x_i), \theta \rangle \geq 1$$

This is the same SVM problem: from a probabilistic interpretation!

It is a nice interpretation. We can have the notion of distances between classes; we can replace equation 10 with:

$$\langle \phi(x_i, y_i), \theta \rangle - \langle \phi(x_i, y_i), \theta \rangle \geq \Delta(y_i, y) \forall y$$

(The first part of that equation will relate to Tiberio's graphical models class.)

We seek a linear separator: we look for a hyperplane. As long as there is data within an ϵ dual of hyperspace, we will penalise.

$$\text{if } f(x) - y \leq \epsilon$$

For multiclass, we can use:

$$\langle \phi(x_i, y_i), \theta \rangle - \langle \phi(x_i, y), \theta \rangle \geq \Delta(y_i, y) \forall y$$

See also, slides on my (Professor Vishwanathan's) homepage at <http://users.rsise.anu.edu.au/~vishy/>.

Multi-class constraints could be:

$$y_i f(x_i) \geq 1$$

$$f(x_i y_i) \geq f(x_i, y) + \int \forall(y + y_i)$$

For further geometry study, you can read Smola and Schölkopf's *Learning with Kernels*, chapter 1.