Manish Tripathi <mtripathi@pivotal.io>

Final Thursday Thunder : Week 9, Learning 9: SVM- Reducing Training size +Kernels

Manish Tripathi <mtripathi@pivotal.io> Wed, Mar 16, 2016 at 1:21 PM
To: PDS-ALL <PDS-ALL@pivotal.io>

Hi

This is my last Thunder post. Though it's not Thursday yet. :-/. Wanted to send a couple more on Boosting, Bagging but unfortunately I am ending the Thunder with this post. I will miss sending these. :-|

This post will be a bit long, as it would have two interesting topics. Please bear with me.

What in SVM again?

So. if you remember my earlier post on Support Vectors then this is an extension on that. In the earlier post, I wrote that how SVM loss function just requires Support Vectors and these points are called "Support " Vectors, since they ***support or*** define the weights of your hyperplane.

Ok. I got that. What else? Don't waste my time by repeating.

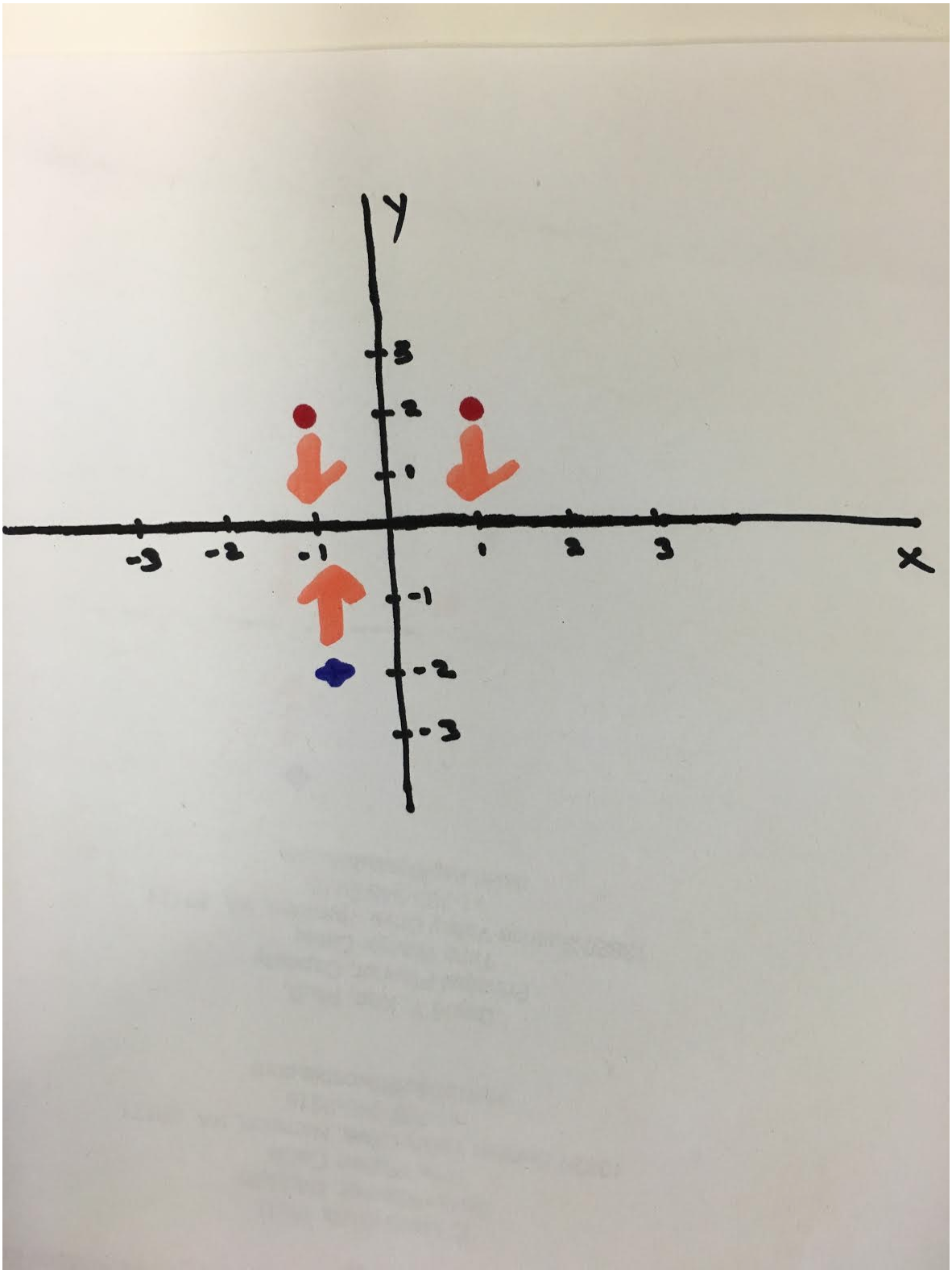
So here is an interesting idea. If somehow you can visualize your data points in 2 or 3 dimensions, then by looking at your points you can actually reduce your training size by dropping all the points which are not Support Vectors. *And still end up with same weights.*

What? How is that possible.

It is. Let me show you how to find Support Vectors in 2 D with a very basic example.

Lets say you have the following points.
(-1,-2), (1,2), (-1,2) such that they are labelled as (+1,-1,-1)

Here is how it looks.



Red are negative points while blue is positive label.

At a cursory glance , X-axis looks to be the separating hyperplane. Then can u tell me which one of these are

Support Vectors?

Well. Looks like all three are Support Vectors. They are the same distance from the hyperplane. :-/

So here is a pretty cool trick. Remember high school Physics?

I guess so :-|

Ha ha. Lets take the torque and equilibrium analogy. Imagine each negative point (red ones) is putting a force on the hyperplane (x-axis) in the downward direction. While the positive point (blue one) is exerting the same force in the upward direction. See the arrows**.**

This creates a Torque on this hyperplane with fulcrum at origin such that hyperplane would rotate (clockwise) if all three points act on it.

I guess I need to brush up on my Torque thing :-(

Aah.. don't bother. *Torque is nothing but the cross product of Force vector and Distance Vector (distance from fulcrum to point of application of Force)**.***

Since all three points have same distance and Force Vector, the right red point will make it rotate.

Now which one these points is NOT required to maintain the hyperplane in equilibrium i.e it has zero torque now?.

Looks like the red point on the right?.

Correct. Since if we remove that point still the one negative point on left and the blue point cancel each other out and the hyperplane (x-axis) still maintains it's equilibrium.

Ok. So that means the right quadrant negative point is not a Support Vector?. Why?
Correct. It is because Support Vectors are those points which "support" the hyperplane. This point is not supporting the hyperplane. If you remove it, still hyperplane remains in same position by other two points. So the other two points are Support Vectors

***So?*.**
So that means if some how you can visualize all your points in 2 or 3D surface, then by visual inspection one can reduce the training data by removing observations which are not Support Vectors and still get the same weights. This means faster training of the algorithm.

Now don't ask me how to visualize a multi-dimensional data. My knowledge is extremely limited and I don't know much. PCA is used but I have few doubts on it's actual usability. Maybe t-SNE. But I haven't used that.

Theoretically it makes sense as shown above**.**

That's cool. What about Kernels.

So I won't talk about what Kernel Trick is. It's basically to do with the dot product property which prevents transforming each feature by a kernel function but rather one can simply take a dot product of the features in the kernel space. Aah... forget that. Lets talk in basic why Kernel helps.

So the idea behind Kernel is to increase the dimensionality of your data set so that one can then be able to linearly separate the data points which might not be possible in original feature space.

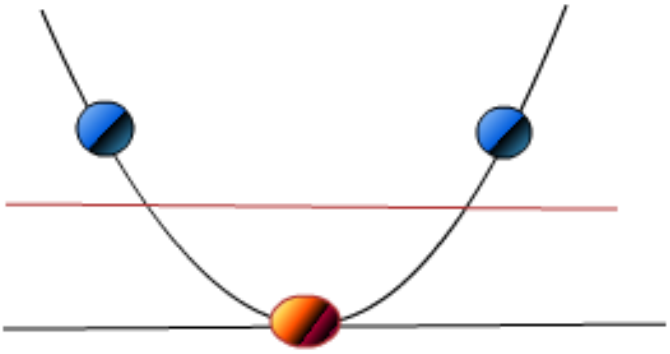
Ok. So how does increasing the dimensionality help in separation?.

Good question. So here is an example.

Imagine you have three points. And you are looking at them from the top. So you are only looking at them in one direction. Here is how it looks.

You are not able to get a hyperplane to separate the red point with two blue points.

Now imagine you are looking at the same points from the front. So like in two dimensions. Here is how it looks.



If you see now, you can get a separating hyperplane which can now separate these two points. The points are same. It's just that you have increased the dimension and now same points appear separated and far away. This is what Kernels in essence do. You increase the dimensions of your feature space and it becomes easier to get a separating hyperplane there.

Cool. Got it.

Glad you understood. 🙂

You ranted a lot these many months and have been very irritating. But will probably miss reading these posts. 😞

Aah.. You don't have to be polite and lie :). You have been sportive ,and a great listener too. Thanks for the patience. Good luck!