Page 92 - WiFi A5 Notebook 2

Support Vector Machines

* Works really well on some moderately sized datasets
* Does not do all that well on large dataset (standard dataset)

Consider figure 8.1.

Is one of these classifiers better?

How do we quantity what we see?

What if we extend out from a decision bound?

The distance it takes to get to a datapoint is referred to a margin.

This region is a hyper-cylinder in n dimensions.

If we maximize this we have a maximum margin (linear) classifiers.

The data points that are closest to this line have a name as well: Support Vectors!

livescribe

Page 93 - WiFi A5 Notebook 2

We can now make two arguments:
1. Margin should be as large as possible
2. Picking support vectors are the most useful datapoints because they are the one we might get wrong.

After training we can throw away all but support vectors!

We've used this before and we will again

$$y = \vec{w} \cdot \vec{x} + b$$

if this is > 0 then we say this is the positive class (+) otherwise it is the negative class (−)

Now we need to define the margin.

If $\vec{x}$ is a support vector then call it $\vec{x}^+$ or $\vec{x}^-$

$\Rightarrow$

Page 95 - WiFi A5 Notebook 2

The book is pretty wrong on page 172. Here is the correct derivation:

For a training set, the margin of the ith sample is:

$$\gamma_i = y_i(\vec{w} \cdot \vec{x}_i + b)$$

where .

Our problem is then:

$$\max_{\vec{w} \cdot \theta} \min_{i} \gamma_i \quad \text{with} \quad \|\vec{w}\| = 1$$

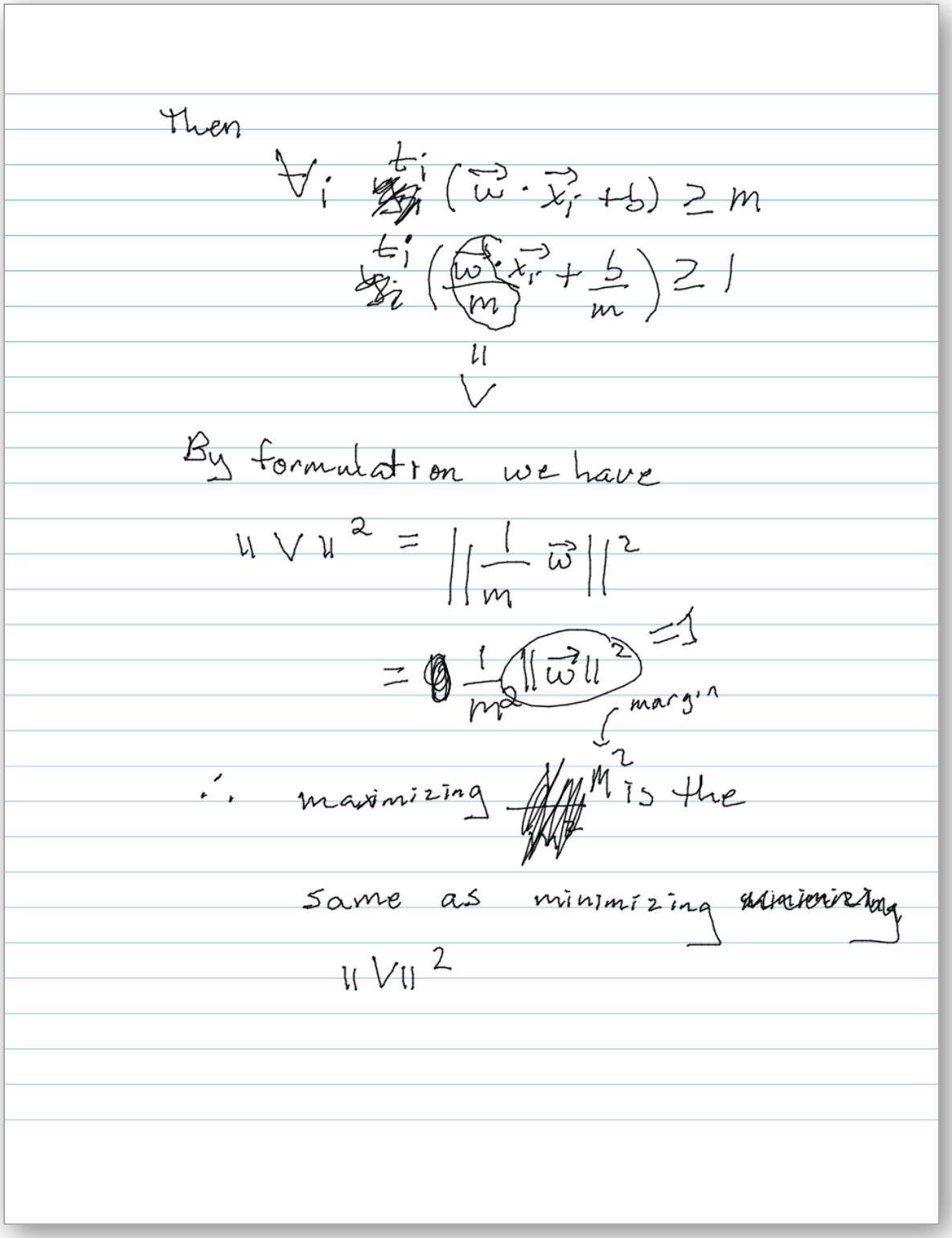
pick support vectors as closest to boundary

Let m be the maximum margin

$$\forall_i \; |\vec{w} \cdot \vec{x}_i + b| \geq m$$

Let t_i be +1 if this is a positive class. Let it be −1 if it is negative.

Page 96 - WiFi A5 Notebook 2



livescribe

Page 97 - WiFi A5 Notebook 2

So our problem is now:

$$\min \frac{1}{2}\|\vec{v}\|^2$$

subject to $t_i(\vec{v}\cdot\vec{x}_i + b) \geq 1$ for all i

The book just uses $\vec{w}$ to represent $\vec{v}$, so to stay consistent we will go back to their formulation which is equivalent.

$$\min \frac{1}{2}\vec{w}^T\cdot\vec{w} \quad \text{subject to} \quad t_i(\vec{w}\cdot\vec{x}_i + b) \;\forall i$$

This problem is actually one that can be solved with quadratic programming, we will not cover that here. But we will formulate the problem.

When we find the optimal solution the following will be satisfied

Page 98 - WiFi A5 Notebook 2

designates optimal

$$\lambda_i^* (1 - t_i (\vec{w}^{*T} x_i + b^*)) = 0$$

$$1 - t_i (\vec{w}^{*T} \vec{x}_i + b) \leq 0$$

$$\lambda_i^* \geq 0$$

Lagrangian multipliers which are positive values.

These are known as the Karush-Kuhn-Tucker (KKT) conditions.

The first condition says that for $\lambda_i \neq 0$

$$1 - t_i (\vec{w}^{*T} \vec{x}_i + b^*) = 0$$

This is only true for support vectors so we only have to consider them.

In jargon these form the active set. For the SV the inequalities ($\leq$) are equalities so we can formulate the lagrangian function $\Rightarrow$

Page 99 - WiFi A5 Notebook 2

$$L(\vec{w}, b, \lambda) = \frac{1}{2}\vec{w}^T\vec{w} + \sum_{i=1}^{n} \lambda_i (1 - t_i(\vec{w}^T\vec{x}_i + b))$$

We differentiate with respect to $\vec{w}$ & b:

$n \leftarrow$ # of SVs

$$\nabla_w L = \vec{w} - \sum_{i=1}^{n} \lambda_i t_i \vec{x}_i$$

and

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \lambda_i t_i$$

If we set derivatives equal to 0, we get:

$$\vec{w}^* = \sum_{i=1}^{n} \lambda_i t_i x_i$$

$$\sum_{i=1}^{n} \lambda_i t_i = 0$$

Page 100 - WiFi A5 Notebook 2

We can substitute back into the equation and rearrange and get:

$$L(\vec{w}^*, b^*, \lambda) = \sum_{i=1}^{n} \lambda_i - \sum_{i=1}^{n} \lambda_i t_i - \frac{1}{2}(A)$$

$$A = \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j t_i t_j \vec{x}_i^T \vec{x}_j$$

This equation is known as the dual problem & we aim to maximize it with respect to λ_i.

Constraints

$\lambda_i \geq 0$ for all i

and $\sum_{i=1}^{n} \lambda_i t_i = 0$

Page 101 - WiFi A5 Notebook 2

So how do you find $\vec{w}^*$?
It is right there in an equation.

How about b^*?

Consider all Support Vectors:

$$b^* \approx \frac{1}{N_s} \sum_{\text{support vectors } j} \left(t_j - \sum_{i=1}^{n} \lambda_i t_i \vec{x}_i^T \vec{x}_j \right)$$

Once we know these we are all set!

But what about problems not linearly separable?

Add in slack variables.

See section 8.1.3.

Derivations are the same.
Coming up next: Kernels!