Nate Crouse

Support Vector Machines

We naturally want to minimize our probability of error. A common approach is to minimize the squared distance of the true value from the predicted value. The error needs be a function of the parameters for the model. The true risk, or test error, in the equation below cannot be known since it's not possible to know the joint distribution of the predictors and the response in $dF(\boldsymbol{x}, y)$.

$$R(\boldsymbol{\alpha}) = \int_{\boldsymbol{x},y} L\big(y, f(\boldsymbol{x}, y)\big)dF(\boldsymbol{x}, y)$$

$F(\boldsymbol{x}, y)$ represents the Cumulative Distribution Function (CDF), whose derivative $dF(\boldsymbol{x}, y)$ or $f(\boldsymbol{x}, y)$ is known as the probability density function (PDF). Another non-starter is that, if the loss $L\big(y, f(\boldsymbol{x}, y)\big)$ is an indicator function, the integral cannot be solved. For these reasons an approximation for $R(\boldsymbol{\alpha})$ must be found. The empirical risk (basically a sample average) represents this; where a loss value is generated for each sample (observation) we have. The empirical risk is defined as the risk measured over the training data. The definition of empirical risk we use can be seen in the equation below:

$$R_{emp}(\boldsymbol{\alpha}) = \sum_{n=1}^{n} L\big(y, f(\boldsymbol{x}, \boldsymbol{\alpha})\big)$$

Having only sample data available, this is the best we can do. Here y is the true outcome label or value, and $f(\boldsymbol{x}, \boldsymbol{\alpha})$ is the estimate. As the function f increases in complexity the loss function will tend to zero, which leads to overfitting, which is explained later. One representation of the empirical risk is represented as $\frac{1}{n}\sum_i \delta\left(c^{(i)} \neq \hat{c}\big(x^{(i)}; \theta\big)\right)$, where $c^{(i)}$ is the true label, $\hat{c}\big(x^{(i)}; \theta\big)$ is the predicted label using data $x^{(i)}$ and parameterized by θ, and n is the number of samples. This represents the proportion of incorrectly classified data points.

The complexity is the representational power of the model. The ability of the model to learn the various types of response labels. The more complex the model, the better at representing the feature to target relationship. For example, when trying to find a decision boundary for two class labels using a classifier represented by functions of the form $\alpha_1 x_1 + \alpha_2 x_2$ in $\mathbb{R}^2$ where x_1 and x_2 are features, and α_1,α_2 are parameters, we will always construct a line through the origin. But a more complex group of functions like those in the form of $\alpha_0 + \alpha_1 x_1 + \alpha_2 x_2$will be able to fit a line through any value of x_2 at α_0 if x_1 goes in the horizontal direction and x_2 in the vertical direction. This more complex group of functions is more flexible, and more likely to help describe the relationship in the data. However, this better representation can be so good that the learner will begin incorporating the noise in the training data. In this situation, when new data, unseen by the training model, needs to be fit, the predictions of misclassified target values will be high. This is the concept of overfitting. The result is a tradeoff. As the model increases in complexity the error on the training data shrinks to zero, which is good, but as complexity increases when fitting testing data using a model fit using the training data there will be a point when the error on the testing set stops decreasing and begins to increase. Finding this minimum point is key and brings us to the Vapnik-Chervonenkis dimension (VC dimension).

The VC dimension deals with the concept of shattering points. A classifier can shatter points $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(n)}$ if and only if $\forall\ y^{(1)}, y^{(2)}, \cdots, y^{(n)}$, the classifier $f(\boldsymbol{x})$ results in no errors on the training data $\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right), \cdots, \left(\boldsymbol{x}^{(1)}, y^{(1)}\right)$. The example in figure 1 below shows two points with two labels and the function $\alpha_0 + \alpha_1 x_1 + \alpha_2 x_2$ as the decision boundary. The boundary line includes the direction the normal vector is pointing, where points are labeled in orange on the side the normal vector is pointing and blue for those behind it. Here we can see that the function can shatter the points and achieve zero error in the training data by correctly classifying the points in any situation. The same would also be true if we added a third point. If a fourth point is added, a situation can be created where the function cannot separate the points as seen in figure 2, and so the VC dimension is 3 when we're in $\mathbb{R}^2$. The generalization of this is that for a space of $\mathbb{R}^n$ the maximum number of vectors that can be shattered by a hyperplane is $n + 1$. Depending on the function we use, the use of the word maximum comes into play. An example of a function in $\mathbb{R}^2$ that has a VC value of 1 would be a circle centered at the origin with radius α (a non-linear function), $x_1^2 + x_2^2 - \alpha = \boldsymbol{x}^T \boldsymbol{x} - \alpha$. Figure 3 shows the case for the circle function where two points are arranged in a way that cannot be shattered, since the points inside the circle would need to be orange and the points outside blue.

Figure 1.

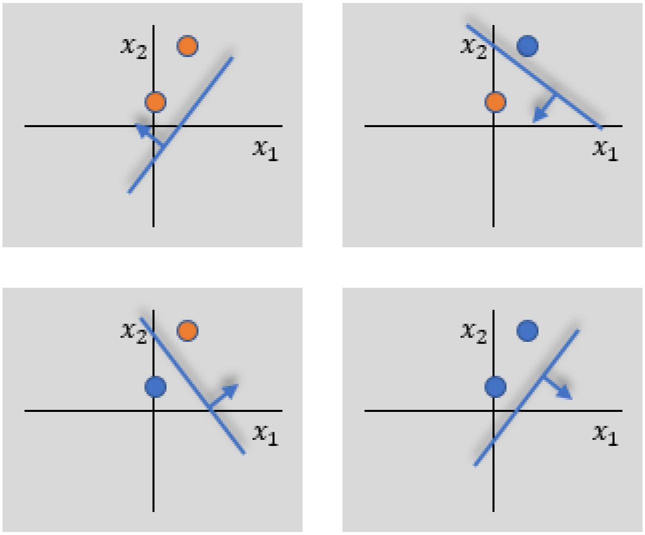


Figure 2.

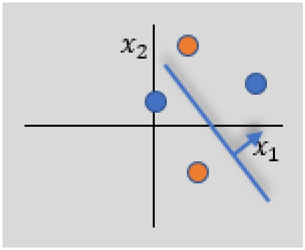
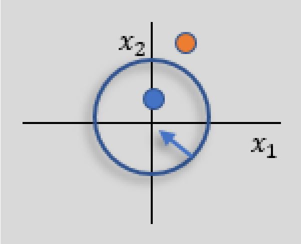


Figure 3.



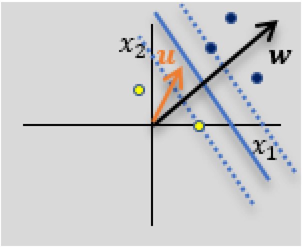
Vapnik was able to show that for a "high probability" of $(1-\eta)$, where η is a very small number, known as the failure probability, that we can define the upper bound on true risk as:

$$R(\boldsymbol{\alpha}) \leq R_{emp}(\boldsymbol{\alpha}) + \sqrt{\frac{h\,log(2n/h) + h - \log(\eta/4)}{n}}$$

The estimated risk on the right side of the inequality is made up of two parts. The first term, $R_{emp}(\boldsymbol{\alpha})$, is the linear empirical risk and is defined as $\frac{1}{2N}\sum_{i=1}^{N}|y - f(\boldsymbol{x},\boldsymbol{\alpha})|$. The summation portion, $|y - f(\boldsymbol{x},\boldsymbol{\alpha})|$, generates values of 0 or 2, and whose sum computes the number of correctly classified points as the numerator. The second term on the right is known as the structural risk. The structural risk is made up of the η term just mentioned, the number of samples n, and the VC dimension h. The VC dimension is a quantity that we can use to measure the complexity in the model. When the VC value, h, is small, the true risk and the estimated risk could be very similar. As the VC term h increases the complexity increases, and thus the structural risk increases. This is assuming that the number of data points n is fixed. If n increases and we have more and more data, the denominator grows and allows the VC dimension to be higher, allowing the support of more complex models without causing the estimated risk to get out of control. The structure provides the complexity, so the structural risk applies to the complexity only, not using the data we're attempting to classify. The structural risk is acting as a penalty term that adds a penalty for more complex models, similar to that used in the Akaike Information Criteria (AIC) or Bayesian Information Criteria (BIC) values where the number of parameters is used as a penalty for model selection.

We maximize the margin between the two classes as to limit the VC dimension so the plane can only separate the patterns x_n according to their labels. We can define a vector $\boldsymbol{w}$, that is perpendicular to the margin lines, and can be any length, and suppose we also have some vector $\boldsymbol{u}$ that is some data point that belongs to an unknown class. A graphical representation can be seen in figure 4. Positive points are in blue and negatively labeled points are in yellow.

Figure 4.



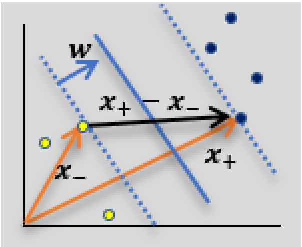
If the dot product is taken between $\boldsymbol{w}$and $\boldsymbol{u}$ we'll have the length of the projection of $\boldsymbol{u}$ onto $\boldsymbol{w}$, so when this is larger than some value c, we'll know we can classify $\boldsymbol{u}$ as being positive (right side of the line). This can be written as $\boldsymbol{w}\cdot\boldsymbol{u}\geq c$ or without loss of generality $\boldsymbol{w}\cdot\boldsymbol{u}+b\geq 0$, where $b=-c$. This will define the decision rule. Here we do not know what value of b to use, or what the vector $\boldsymbol{w}$ should be, so we begin to add some constraints. If we take the dot product of $\boldsymbol{w}$ with some positive sample, $\boldsymbol{x}_+$, that's been observed, add b, and say we want this greater than or equal to 1; using the decision boundary $\boldsymbol{w}\cdot\boldsymbol{u}+b\geq 0$ we have $\boldsymbol{w}\cdot\boldsymbol{x}_+ + b\geq 1$. For negative samples, $\boldsymbol{x}_-$, to the left of the decision boundary we have $\boldsymbol{w}\cdot\boldsymbol{x}_- + b\leq -1$. To simplify the two inequalities into one, we define y_i as being +1 for positive samples and -1 for negative samples, and get:

$$y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)-1\geq 0 = y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)\geq 1 = \begin{cases} \boldsymbol{w}\cdot\boldsymbol{x}_+ + b\geq 1\ (positive\ samples) \\ \boldsymbol{w}\cdot\boldsymbol{x}_- + b\leq -1\ (negative\ samples) \end{cases}$$

We add the constraint that $y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)-1=0$ for $\boldsymbol{x}_i$ values that lie exactly on the margin. (i)

Continuing with the notion of maximizing the width of the margins, we need to find the total width of the margins. Since positive points lie to the right of the line, and negative points to the left of the line, we can take the vector difference of the positive point $\boldsymbol{x}_+$ with a negative point $\boldsymbol{x}_-$ that lie on the margin as seen in figure 5. If the dot product is taken using this difference and the normalized vector $\boldsymbol{w}$, we will have the total width of the margin, $(\boldsymbol{x}_+ - \boldsymbol{x}_-)\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$.

Figure 5.



Using the equations in (i), if the point is positive ($y_i=1$):

$$y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)-1=0 \Rightarrow\ y_i\boldsymbol{w}\cdot\boldsymbol{x}_i+y_ib-1=0 \Rightarrow\ \boldsymbol{w}\cdot\boldsymbol{x}_i+b-1=0\ \Rightarrow\ \boldsymbol{w}\cdot\boldsymbol{x}_i=1-b$$

And for negative points ($y_i=-1$):

$$y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+b)-1=0 \Rightarrow\ y_i\boldsymbol{w}\cdot\boldsymbol{x}_i+y_ib-1=0 \Rightarrow\ -\boldsymbol{w}\cdot\boldsymbol{x}_i-b-1=0\ \Rightarrow\ \boldsymbol{w}\cdot\boldsymbol{x}_i=-1-b$$

Plugging these results into $(\boldsymbol{x}_+-\boldsymbol{x}_-)\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$, we have

$$(\boldsymbol{x}_+-\boldsymbol{x}_-)\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}=(\boldsymbol{w}\boldsymbol{x}_+-\boldsymbol{w}\boldsymbol{x}_-)\frac{1}{\|\boldsymbol{w}\|}=\big((1-b)-(-1-b)\big)\frac{1}{\|\boldsymbol{w}\|}=\frac{2}{\|\boldsymbol{w}\|}$$

So the total width across the margins that want to maximize is $\frac{2}{\|\boldsymbol{w}\|}$. This can be rearranged into

$$\max\frac{2}{\|\boldsymbol{w}\|}\propto\max\frac{1}{\|\boldsymbol{w}\|}\propto\min\|\boldsymbol{w}\|\propto\min\frac{1}{2}\|\boldsymbol{w}\|^2$$

Where the last step will be used for mathematical convenience.

The previous assumes that the data is separable, so no points are misclassified. When the data is non-separable we need to introduce slack variables ξ_n, so now $y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n+b)\geq 1$ becomes $y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n+b)\geq 1-\xi_\mathrm{n}$, and $\xi\geq 0$. Also, a new term is added for minimization so that now we need to minimize $\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{n=1}^{N}\xi_n$ with the constraints that $y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n+b)-1+\xi_\mathrm{n}\geq 0$ and $\xi\geq 0$. The C term is a tradeoff parameter that is unspecified.

If the extremum of a function with constraints is needed, then Lagrange optimization will have to be employed. The results of which will give a new expression that can be maximized or minimized without having to worry about the constraints. In this case there are $2N$ constraints, so there are $2N$ multipliers needed. N constraints α_n for the $y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n+b)-1+\xi_\mathrm{n}\geq 0$ constraint, and N multipliers μ_n for the $\sum_{i=1}^{N}\xi_n$ constraints. The Lagrangian is then defined as:

$$L_p(\boldsymbol{w},\xi_n,\alpha_n,\mu_n)=\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{n=1}^{N}\xi_n-\sum_{n=1}^{N}\alpha_n(y_n(\boldsymbol{w}\boldsymbol{x}_n+b)-1+\xi_n)-\sum_{n=1}^{N}\mu_n\xi_n$$

Where each constraint is constrained to be zero. That is, $\mu_n\xi_n=0$ and $y_n(\boldsymbol{w}\boldsymbol{x}_n+b)-1+\xi_n=0$.

Now, taking the derivatives:

$$\frac{\partial}{\partial\boldsymbol{w}}L_p(\boldsymbol{w},\xi_n,\alpha_n,\mu_n)=\boldsymbol{w}-\sum_{n=1}^{N}\alpha_ny_n\boldsymbol{x}_n=0$$

$$\boldsymbol{w}=\sum_{n=1}^{N}\alpha_ny_n\boldsymbol{x}_n$$

$$\frac{\partial}{\partial\xi}L_p(\boldsymbol{w},\xi_n,\alpha_n,\mu_n)=C(1)-\alpha_n(1)-\mu_n(1)=C-\alpha_n-\mu_n=0$$

$$\frac{\partial}{\partial b}L_p(\boldsymbol{w},\xi_n,\alpha_n,\mu_n)=-\sum_{n=1}^{N}\alpha_ny_n(1)=\ -\sum_{n=1}^{N}\alpha_ny_n=\sum_{n=1}^{N}\alpha_ny_n=0$$

These are the Karush Khun Tucker conditions:

$$\boldsymbol{w}=\sum_{n=1}^{N}\alpha_ny_n\boldsymbol{x}_n$$

$$C - \alpha_n - \mu_n = 0$$

$$\sum\nolimits_{n=1}^{N} \alpha_n y_n = 0$$

$$\mu_n \xi_n = 0$$

$$\alpha_n(y_n(\boldsymbol{w}\boldsymbol{x}_n + b) - 1 + \xi_n) = 0$$

$$\alpha_n \geq 0, \mu_n \geq 0, \xi_n \geq 0$$

Since $\xi_n > 0$ for values that are inside the margin or misclassified, at these points for $\mu_n \xi_n = 0$ we have $\mu_n = 0$ and $C - \alpha_n - \mu_n = 0 \Rightarrow C - \alpha_n - 0 = 0 \Rightarrow \alpha_n = C$. If the sample is outside the margin and correctly classified, then $\xi_n = 0$ and $\alpha_n(y_n(\boldsymbol{w}\boldsymbol{x}_n + b) - 1 + \xi_n) = 0 \Rightarrow \alpha_n(y_n(\boldsymbol{w}\boldsymbol{x}_n + b) - 1) = 0 \Rightarrow \alpha_n = 0$, and if the sample is on the margin $\alpha_n(y_n(\boldsymbol{w}\boldsymbol{x}_n + b) - 1 + \xi_n) = 0$ leads to $0 < \alpha_n < C$.

Taking the value of $\boldsymbol{w} = \sum_{n=1}^{N} \alpha_n y_n \boldsymbol{x}_n$ and plugging it into $L_p(\boldsymbol{w}, \xi_n, \alpha_n, \mu_n)$, we have

$$L_p(\boldsymbol{w}, \xi_n, \alpha_n, \mu_n) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum\nolimits_{n=1}^{N} \xi_n - \sum\nolimits_{n=1}^{N} \alpha_n(y_n(\boldsymbol{w}\boldsymbol{x}_n + b) - 1 + \xi_n) - \sum\nolimits_{n=1}^{N} \mu_n \xi_n =$$

$$= \frac{1}{2}\left(\sum_n \alpha_n y_n \boldsymbol{x}_n\right)\left(\sum_m \alpha_m y_m \boldsymbol{x}_m\right) + C\sum\nolimits_{n=1}^{N} \xi_n - \left(\sum_n \alpha_n y_n \boldsymbol{x}_n\right)\left(\sum_m \alpha_m y_m \boldsymbol{x}_m\right) - \sum_n \alpha_n y_n b$$
$$+ \sum_n \alpha_n - \sum_n \alpha_n \xi_n - \sum\nolimits_{n=1}^{N} \mu_n \xi_n =$$

$$= \frac{1}{2}\left(\sum_n \alpha_n y_n \boldsymbol{x}_n\right)\left(\sum_m \alpha_m y_m \boldsymbol{x}_m\right) - \left(\sum_n \alpha_n y_n \boldsymbol{x}_n\right)\left(\sum_m \alpha_m y_m \boldsymbol{x}_m\right) - b\underbrace{\sum_n \alpha_n y_n}_{=0} + \sum_n \alpha_n$$
$$+ C\sum\nolimits_{n=1}^{N} \xi_n - \sum_n \alpha_n \xi_n - \underbrace{\sum\nolimits_{n=1}^{N} \mu_n \xi_n}_{=0} =$$

$$= -\frac{1}{2}\left(\sum_n \alpha_n y_n \boldsymbol{x}_n\right)\left(\sum_m \alpha_m y_m \boldsymbol{x}_m\right) + \sum_n \alpha_n + C\sum\nolimits_{n=1}^{N} \xi_n - \sum_n \alpha_n \xi_n =$$

$$= -\frac{1}{2}\sum_n \sum_m \alpha_n \alpha_m y_n y_m \boldsymbol{x}_n \cdot \boldsymbol{x}_m + \sum_n \alpha_n + C\sum\nolimits_{n=1}^{N} \xi_n - \sum_n \alpha_n \xi_n =$$

For the terms $C\sum_{n=1}^{N} \xi_n - \sum_n \alpha_n \xi_n$, there are 3 cases:

1. When points are inside the margin or misclassified $\alpha_n = C$ and $\xi_n > 0$, so $C\sum_{n=1}^{N} \xi_n - \sum_n \alpha_n \xi_n = C\sum_{n=1}^{N} \xi_n - \sum_n C\xi_n = 0$.
2. When the points are outside the margin and correctly classified $\alpha_n = 0$ and $\xi_n = 0$, so $C\sum_{n=1}^{N} \xi_n - \sum_n \alpha_n \xi_n = 0$.
3. When the points are on the margin, $0 \leq \alpha_n < C$ and $\alpha_n = C - \mu_n$, so $\mu_n > 0$ and the constraint $\mu_n \xi_n = 0$ implies that $\xi_n = 0$. This results in $C\sum_{n=1}^{N} \xi_n - \sum_n \alpha_n \xi_n = 0 - 0 = 0$

Since all 3 cases are zero, we have $C\sum_{n=1}^{N} \xi_n - \sum_n \alpha_n \xi_n = 0$, and we finally have the dual solution

$$-\frac{1}{2}\sum_{n}\sum_{m}\alpha_n\alpha_m y_n y_m \boldsymbol{x}_n\cdot\boldsymbol{x}_m+\sum_{n}\alpha_n+C\sum\nolimits_{n=1}^{N}\xi_n\;-\sum_{n}\alpha_n\xi_n=$$

$$=-\frac{1}{2}\sum_{n}\sum_{m}\alpha_n\alpha_m y_n y_m \boldsymbol{x}_n\cdot\boldsymbol{x}_m+\sum_{n}\alpha_n+0=$$

$$-\frac{1}{2}\sum_{n}\sum_{m}\alpha_n\alpha_m y_n y_m \boldsymbol{x}_n\cdot\boldsymbol{x}_m+\sum_{n}\alpha_n$$

The dual solution is then optimized using quadratic programming. There are a few drawbacks to using SVM. One is that all of the points have to be labeled since the parameter vector $\boldsymbol{w}=\sum_{n=1}^{N}\alpha_n y_n \boldsymbol{x}_n$ requires labeled values. Another is that the simple form of SVM shown here is suitable only for data two-classes.