Lesson 6: Kernel method & SVM

The best line

The discriminant line that generalizes best is the **farthest from the two classes** ("commits least to the training data" – not to believe the training data while still fitting the data).

Support vector machine (massive handwaving)

Least commitment -> We are interested in maximizing the size of the margin.

The general linear classifier formula is: $y = w^T x + b$

- y is the **classification indicator**
- w is the parameter of the plane
- b is how far it moves out of origin

The equation for the discriminant line (hyperplane) is $0 = w^T x + b$

Translate this line up or down (e.g. to the first positive or negatice data point it encounters) to get $1 = w^T x + b$ or $-1 = w^T x + b$

We want w such that the distance between these two lines in maximal (and the choice of 1 and -1 is really arbitrary – we just want w "up to" scaling).

Take the difference of 2 equations: $w^T(x_1 - x_2) = 2$ (where x_1 and x_2 are the nearest data points the line can hit if it moves upward or downward, i.e. **support vectors**).

Divide both side by $||w||$. Then $w^T/||w||(x_1 - x_2) = 2/||w||$. This is the distance between the hyperplanes.

Now we have the difference of x_1 and x_2 projected onto the normalized w vector... Since w is parallel to the vector from x_1 to x_2. Some handwaving magic. The left handside is the **margin** m.

$$m = 2/||w||$$

So handwavily, the "distance" between the -1 line and 1 is $2/||w||$ and we can maximize this by minimizing w's magnitude (maximize the margin). But it needs to be still consistent with the data.

Support vector machines

$max \frac{2}{||w||}$ while classifying everything correctly.

Classifying everything correctly in mathmatical expression:

$$y_i(w^T x_i + b) \geq 1, \ \forall i$$

- (Note that support vectors are at 1 so in the equation there is an 1 not 0.)

The above problem is hard, so alternatively we:

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

This is equivalent to a quadratic programming problem which can be solved by jumping off a cliff:

$$max \ W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

Where α_i is a new set of parameters such that:

$$\alpha_i \geq 0, \ \sum_i \alpha_i y_i = 0$$

Some properties of the solution.

- We can recover $w = \sum_i a_i y_i x_i$ and b.
- a_i are mostly 0. This means which means only some of the x_i matter. These data points are support vectors.
- The data points are being dot producted $x_i^T x_j$. This is a kind of measuring the similarity of points in direction.

The locality because of throwing the far data points away is just like k-NN except we learn which points are important (as opposed to considering all neighbors).

SVMs: linearly married

What if some points can not be classified correctly? Then while maximizing the margin, we can impose a cost for misclassification.

What if lines are a bad hypothesis? Transform the data points and throw them into a higher-dimensional space where a line might be a better hypothesis.

For example:

$$\Phi(q) =< q1^2, q2^2, \sqrt{2} q_1 q_2 >$$

Then we can find:

$$\Phi(x)\Phi(y) = x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 = (x_1y_1 + x_2y_2)^2 = (x^T \cdot y)^2$$

This makes the classifier into a circle instead of a line. We transformed the similarity of direction into whether fall in or out a circle.

This also makes the computation very efficiently. Instead of computing Φ, by just squaring the dot product, we can represent the different similarity – **the kernel trick**.

There are infinite number of transformations and some of them will represent the data structure.

Kernels

We can replace the dot product $x_i^T x_j$ with a kernel $K(x_i, x_j)$.

Implicitly throws data points into higher dimensional space and still get some kind of **similarity**. But because of the kernel trick, no additional computation is needed.

Kernel choice requires **domain knowledge**.

$$K(x, y) = x^T y$$
$$K(x, y) = (x^T y)^2$$

Polynomial kernel:

$$K(x, y) = (x^T y + c)^p$$

RBF kernel:

$$K(x, y) = exp(-\frac{||x - y||^2}{2\sigma^2})$$

- When x and y are very similar, the K will be 1. When they are very far apart, the K will be close to 0.
- Just like Gaussian.

Another kernel:

$$K(x, y) = tanh(\alpha x^T y + \theta)$$

- This act like sigmoid.

Kernels must satisfy **Mercer condition** – must act like a similarity or a distance.

SVMs – review

- margins ~ generalization and overfitting
- margins: big is better
- optimization problem for finding max margins: quadratic programming and dual problem
- support vectors
- kernel trick $K(x, y)$ which extends the inner product $x^T y$
- Mercer condition

Back to boosting

Boosting doesn't seem to conventionally overfit (testing error keeps going down).

Confidence – how confident is a classification? How strongly you believe in a particular answer. (e.g. Number of neighbors agreeing in k-NN or local variation in linear regression).

From the formula $H_{final}(x) = sgn(\sum_t \alpha_t * h_t(x))$, we can get the inner part:

$$\frac{\sum_t \alpha_t * h_t(x)}{\sum_t \alpha_t}$$

This is the confidence. Hard examples fall near the center (toward 0). As we boost further, the hard points also start having higher confidence. The margin keeps increasing and does not overfit.

Boosting will **overfit** if the underlying weak learners overfit.

Pink noise – uniform noise. Boosting overfits.

White noise – Gaussian noise.