

# Mathematics For Data Science

Brian J. Mann

Feb 10-11, 2016

# Who am I?

- ▶ PhD from University of Utah in 2014 (Geometric Group Theory)
- ▶ Worked for **Amazon Web Services** doing supply chain optimization and forecasting
- ▶ [brian.mann@galvanize.com](mailto:brian.mann@galvanize.com)
- ▶ Github: [brianmanngalvanize](https://github.com/brianmanngalvanize)

# Objectives

- ▶ Learn some fun mathematics
  - ▶ Kernel trick for support vector machines
  - ▶ VC Generalization Bound (why machine learning works!)
- ▶ See how understanding how machine learning is working “under the hood” can improve your intuition about model selection and hyper-parameter choice

# Support Vector Machines (SVM)

- ▶ Idea: try to separate classes by an optimal hyperplane
- ▶ Here *optimal* means that the minimum distance from the hyperplane to any of the training points (the *margin*) is maximal.

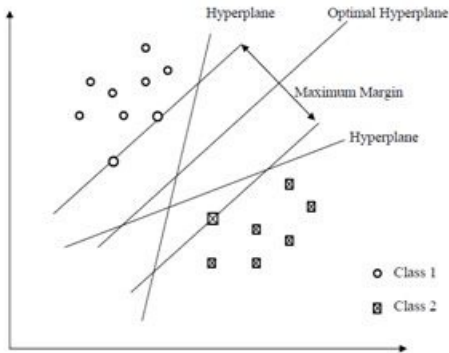


Figure 1:Maximal Margin

Ok, that sounds great. What's the problem?

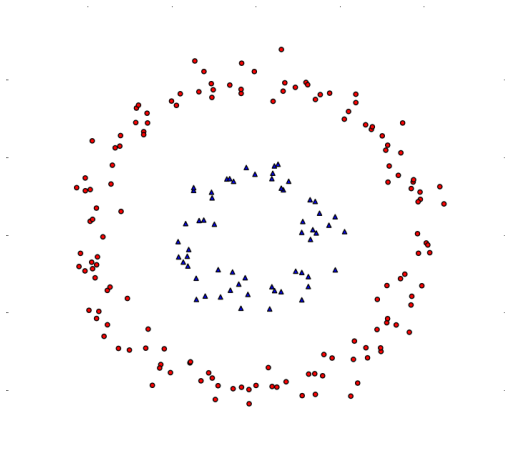


Figure 2: Oh.

# The solution

- Map our data to a higher dimensional space where it's (almost) linearly separable

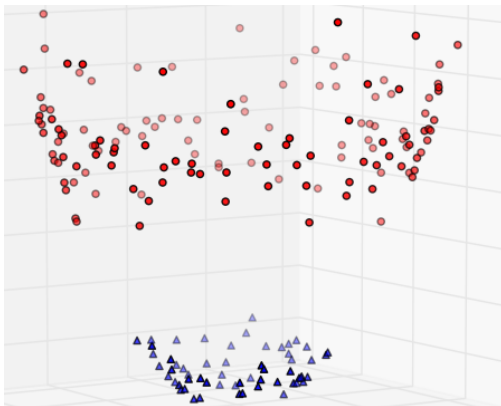


Figure 3:Yay!

# Issue 1: Memory

- ▶ Even for polynomial transformations, the numbers of dimensions (features) in the target space can grow very quickly
- ▶ Consider the transformation  $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^5$  given by

$$(x_1, x_2) \mapsto (x_1^2, x_1 x_2, x_2^2, x_1, x_2, 1)$$

- ▶ More generally, a mapping  $\phi_d : \mathbb{R}^N \rightarrow \mathbb{R}^{\binom{N+d}{d}}$  that maps a vector to the vector of all monomial terms in  $N$  variables of degree  $\leq d$
- ▶  $\binom{N+d}{d}$  grows very quickly as  $N, d \gg 0$

## Issue 2: Computation

- ▶ The optimal hyperplane for the transformed data is

$$f(x) = \sum_i a_i \cdot \langle \phi(x_i), \phi(x) \rangle + b = 0$$

- ▶ Need to compute the dot product of high-dimensional vectors (in fact, sometimes they might be infinite dimensional!)



# A solution

- ▶ What if there was a way to compute  $\langle \phi(x), \phi(y) \rangle$  directly without ever computing  $\phi(x)$  or  $\phi(y)$ ?
- ▶ There is!
- ▶ This is what *kernel functions* do for us

# What is a kernel function?

A *kernel function* is a continuous function

$$K : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$$

which satisfies

1.  $K(x, y) = K(y, x)$  (symmetric)
2.  $K$  is positive-semidefinite i.e.

$$\sum_i \sum_j K(x_i, x_j) c_i c_j > 0$$

for all finite sequences  $x_1, \dots, x_n$  and all  $c_i, c_j \in \mathbb{R}$

These definitions are a little opaque, but the idea is that a kernel function generalizes the idea of an inner product (also known as a linear kernel).

# Mercer's Theorem

*Mercer's Theorem* says that if  $K : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$  is a kernel function, then there exists a vector space with an inner product (a *Hilbert space*)  $V$  and a mapping  $\phi : \mathbb{R}^N \rightarrow V$  so that

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

- ▶ In English, if  $K$  is a kernel function, it consists of a transformation followed by an inner product in some higher dimensional space,  $V$ .
- ▶ Kernels allow us to compute high-dimensional (sometimes infinite!) inner products in  $V$  in terms of our original inputs in  $\mathbb{R}^N$ .

## Example: Polynomial kernel

- ▶  $K(x, y) = (\langle x, y \rangle + c)^d$
- ▶  $c$  and  $d$  are chosen *a priori* by the user, not trained
- ▶ Find  $c$  and  $d$  by cross-validation
- ▶ Comes from the polynomial transformation  $\phi_d$  (ignoring some constants)

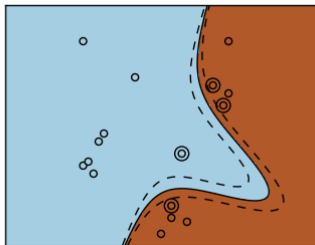


Figure 4: Polynomial Kernel

## Example: RBF (Gaussian) kernel

- ▶  $K(x, y) = e^{-\gamma \|x - y\|^2}$
- ▶  $\gamma$  is chosen *a priori* by the user
- ▶ Largest when  $x$  and  $y$  are close, decays exponentially
- ▶ With the RBF kernel, SVM looks for clusters of similarly labeled points
- ▶ Can learn much more complicated decision boundaries compared to polynomial or linear kernels (watch for overfitting, adjust  $\gamma$ )

## More RBF kernel

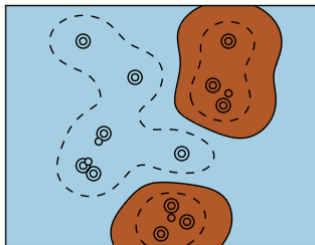


Figure 5:RBF Kernel

## More RBF kernel

What are  $\phi$  and the dimension of  $V$  in this case?

- ▶ Let  $\gamma = 1/2$  for ease of computation, then

$$K(x, y) = \sum_{j=0}^{\infty} \frac{\langle x, y \rangle^j}{j!} \exp\left(-\frac{\|x\|^2}{2}\right) \exp\left(-\frac{\|y\|^2}{2}\right)$$

- ▶ With a little algebra one gets

$$\phi(x) = \left( \frac{e^{-\frac{\|x\|^2}{2j}}}{\sqrt{j!}^{1/j}} \binom{j}{n_1, \dots, n_k} \right)_{j=0, \dots, \infty, \sum_{i=1}^k n_i = j}^{1/2}$$

- ▶  $V$  is infinite dimensional ( $V = l^2$  the space of square-summable sequences)

# Overview

- ▶ Understand how the kernel trick works
- ▶ Pick the right kernel for what you think the decision boundary might look like
- ▶ Questions?



# VC Dimension and the VC Bound Theorem

**Question:** Why should you expect that your training error tells you anything about the error of your model on new data? In other words, why/how do you know that your model will generalize at all?

We need to start somewhere

### Theorem (Hoeffding Inequality)

Suppose  $X_1, \dots, X_n$  are iid random variables and let

$$\bar{X} = \frac{X_1 + X_2 + \dots + X_N}{N}$$

then

$$\mathbb{P}(|\bar{X} - \mathbb{E}(\bar{X})| > \epsilon) \leq 2e^{-2\epsilon^2 N}$$

# In English

- ▶ For any threshold value  $\epsilon$ , if you sample a random variable enough times, the probability that the sample mean differs from the true mean by more than  $\epsilon$  is nearly 0.
- ▶ i.e. If you flip a fair coin enough times, you expect the ratio of heads to tails get arbitrarily close to 1:1 with high probability.

# Assumptions

- ▶ For the purposes of this talk, we'll focus on binary classification problems
- ▶ All of this can be extended to other supervised learning problems
- ▶ Positive and negative classes will be represented as  $+1$  and  $-1$

# What does this have to do with machine learning?

- ▶ Suppose  $h$  is some hypothesis (a function that classifies observations as either  $+1$  or  $-1$ )
- ▶  $E_{train}(h)$  = error rate on your training set
- ▶  $E_{gen}(h)$  = true error rate of  $h$

$E_{gen}(h)$  and  $E_{train}(h)$  are random variables that satisfy the hypotheses of the Hoeffding Inequality so

$$\mathbb{P}(|E_{train}(h) - E_{gen}(h)| > \epsilon) \leq 2e^{-2\epsilon^2 N}$$

# Naive error generalization bound

Suppose:

1. We have a finite hypothesis set  $\{h_1, \dots, h_M\}$
2. Given some training data, we apply some learning procedure to find the optimal hypothesis  $g$

It is NOT TRUE that

$$\mathbb{P}(|E_{train}(g) - E_{gen}(g)| > \epsilon) \leq 2e^{-2\epsilon^2 N}$$

since  $g$  is chosen after the data is generated

# Naive error generalization bound

However, the event

$$|E_{train}(g) - E_{gen}(g)| > \epsilon$$

is in the union of the events

$$|E_{train}(h_i) - E_{gen}(h_i)| > \epsilon$$

so

$$\mathbb{P}(|E_{train}(g) - E_{gen}(g)| > \epsilon) \leq 2Me^{-2\epsilon^2 N}$$

# Problem with this naive bound

- ▶ The bound

$$\mathbb{P}(|E_{train}(g) - E_{gen}(g)| > \epsilon) \leq 2Me^{-2\epsilon^2 N}$$

only works for finite hypothesis sets

- ▶ Not really useful for any real-world examples



# Can we do better?

- ▶ Even with finitely many hypothesis functions the bound is still quite bad since the events

$$|E_{train}(h_i) - E_{gen}(h_i)| > \epsilon$$

probably have large overlaps

- ▶ In fact, they overlap enough to allow us to work with infinite hypothesis sets (i.e. hyperplanes in  $\mathbb{R}^N$ )

# Shattering

- ▶ Let  $\mathcal{H}$  be our hypotheses set
  - ▶  $\mathcal{H}$  might be all separating hyperplanes in  $\mathbb{R}^N$
  - ▶ or the set of all of the possible decision functions you can get with a neural network of some fixed topology
- ▶ A *dichotomy* is a choice of label  $+1$  or  $-1$  for each point in a data set
- ▶ For any finite data set  $\{x_1, \dots, x_N\}$ , each  $h \in \mathcal{H}$  gives a dichotomy  $h(x_1), \dots, h(x_N) \in \{+1, -1\}^N$
- ▶ Define

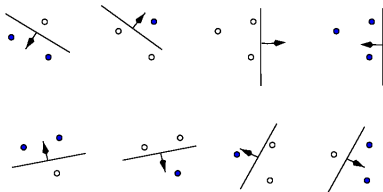
$$m_{\mathcal{H}(N)} = \max_{x_1, \dots, x_N} |\{h(x_1), \dots, h(x_N) | h \in \mathcal{H}\}|$$

# Shattering

- ▶  $m_{\mathcal{H}}(N) \leq 2^N$
- ▶  $\mathcal{H}$  shatters  $x_1, \dots, x_N$  if  $|\{h(x_1), \dots, h(x_N) | h \in \mathcal{H}\}| = 2^N$
- ▶ In this case  $m_{\mathcal{H}}(N) = 2^N$
- ▶ In other words,  $\mathcal{H}$  shatters a set of points if it could obtain zero training error on that set

# Examples

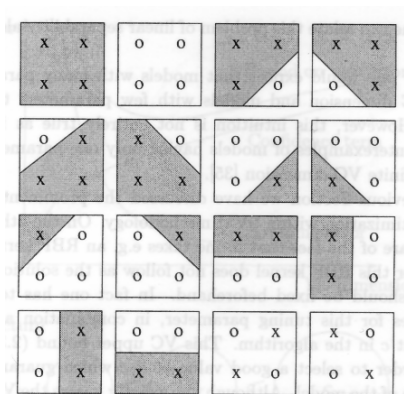
- ▶ Linear decision boundaries shatter 3 points in  $\mathbb{R}^2$



- ▶ Convex polygons in  $\mathbb{R}^2$  shatter arbitrarily many points
  - ▶ Choose  $N$  points on the unit circle
  - ▶ Take the convex hull of the +1's

# Non-example

- ▶ Linear decision boundaries do not shatter any 4 points in  $\mathbb{R}^2$



# Vapnik-Chervonenkis (VC) Dimension

- ▶ The VC Dimension  $d_{VC}$  of  $\mathcal{H}$  is defined to be the largest  $N$  for which  $m_{\mathcal{H}}(N) = 2^N$
- ▶ Said another way

$$d_{VC} \geq N$$

$$\Leftrightarrow$$

there exists a data set of size  $N$  such that  $\mathcal{H}$  shatters it

- ▶ The VC Dimension gives a polynomial upper bound on  $m_{\mathcal{H}}(N)$

$$m_{\mathcal{H}}(N) \leq N^{d_{VC}} + 1$$

- ▶ VC Dimension is a measure of complexity of a hypothesis

# Examples

- ▶ For linear decision boundaries in  $\mathbb{R}^2$ ,  $d_{VC} = 3$
- ▶ For linear decision boundaries in  $\mathbb{R}^N$ ,  $d_{VC} = N + 1$ 
  - ▶ Choose  $N + 1$  points which do not live on the same hyperplane
  - ▶ Can get any dichotomy on these points, so  $d_{VC} \geq N + 1$
  - ▶ With  $N + 2$  points, can find hyperplane through  $N$  points with the remaining two points on either side
  - ▶ Labelling points on this hyperplane  $-1$ , others  $+1 \Rightarrow$  impossible dichotomy
  - ▶  $d_{VC} = N + 1$
- ▶ Convex polygons in  $\mathbb{R}^2$  have  $d_{VC} = \infty$

# The VC Bound

The VC generalization bound states that for any  $\delta > 0$

$$E_{gen}(g) \leq E_{train}(g) + \sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}}$$

with probability  $1 - \delta$



# What does this tell us?

1. Since  $m_{\mathcal{H}}(N)$  is bounded by a polynomial of degree  $d_{VC}$  in  $N$ , the RHS of

$$E_{gen}(g) \leq E_{train}(g) + \sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{\delta}}$$

→  $E_{train}(g)$  as the size of the training set increases

2. Since  $m_{\mathcal{H}}(N)$  or  $d_{VC}$  is a measure of model complexity, more complicated models make the bound worse (overfitting!!!!)

## Examples

Suppose we want  $E_{gen}(g)$  to be within 10% of  $E_{train}(g)$  with 90% confidence for a model with  $d_{VC} = 3$ . How much data do we need?

- ▶ From the VC Bound

$$\sqrt{\frac{8}{N} \ln \frac{4m_{\mathcal{H}}(2N)}{0.1}} \leq 0.1$$

- ▶ So

$$N \geq \frac{8}{0.1^2} \ln \left( \frac{4(2N)^3 + 4}{0.1} \right)$$

- ▶  $N \sim 30,000$
- ▶ In general,  $N \sim 10,000 \times d_{VC}$
- ▶ Empirically,  $N \sim 10 \times d_{VC}$  (VC bound badly overestimates)

# Conclusion

- ▶ The VC Generalization Bound is the theorem that tells you machine learning actually works!
- ▶ Obtain statistical guarantees about how well your model generalizes
- ▶ Extremely slack bound, but better than nothing

## Sketch of Proof (optional)

*Theorem (VC Generalization Bound)*

$$\mathbb{P} \left( \sup_h |E_{train}(h) - E_{gen}(h)| > \epsilon \right) \leq 4m_{\mathcal{H}}(2N)e^{-\frac{1}{8}\epsilon^2 N}$$

1. Instead of working with the space of all possible observations, pick a second test set the same size as the training set

$$\mathbb{P} \left( \sup_h |E_{train}(h) - E_{gen}(h)| > \epsilon/2 \right) \leq$$
$$2\mathbb{P} \left( \sup_h |E_{train}(h) - E_{test}(h)| > \epsilon/2 \right)$$

## Sketch Part 2

2. Replace the infinite hypothesis set with the number of dichotomies that the hypothesis set can have on a finite set  $S$  of points giving

$$\mathbb{P} \left( \sup_h |E_{train}(h) - E_{test}(h)| > \epsilon/2 \right) \leq$$

$$m_{\mathcal{H}}(2N) \times \sup_S \sup_h \mathbb{P} (|E_{train}(h) - E_{test}(h)| > \epsilon/2 | S)$$

## Sketch Part 3

3. Use the Hoeffding bound to show

$$\mathbb{P}(|E_{train}(h) - E_{test}(h)| > \epsilon/2 | S) \leq 2e^{-\frac{1}{8}\epsilon^2 N}$$

# Conclusion

- ▶ Obtain statistical guarantees about how well your model generalizes
- ▶ Extremely slack bound, but better than nothing

# References

- ▶ Yaser S. Abu-Mostafa, Malik Magdon-Ismail, Hsuan-Tien Lin  
*Learning From Data: A Short Course*
- ▶ Eduardo Sontag, *VC Dimension of Neural Networks*,  
[http://www.mit.edu/~esontag/FTP\\_DIR/vc-expo.pdf](http://www.mit.edu/~esontag/FTP_DIR/vc-expo.pdf)
- ▶ Images: <http://www.svms.org/vc-dimension/>