CS229 Note Lecture 10

Let $|\mathcal{H}| = k$, and let γ, δ be fixed, then with probability at least $1 - \delta$ we have

$$\epsilon(\hat{h}) \leq \left(\min_{h \in \mathcal{H}} \epsilon(h) \right) + 2\sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}$$

h suffices that

$$m \geq \frac{1}{2\gamma^2} \log \frac{2k}{\delta} = O\left(\frac{1}{\gamma^2} \log \frac{k}{\delta} \right)$$

VC Dimension

Suppose $\mathcal{H}$ is parameterized by d real numbers, so

$$k = |\mathcal{H}| = 2^{64d}$$

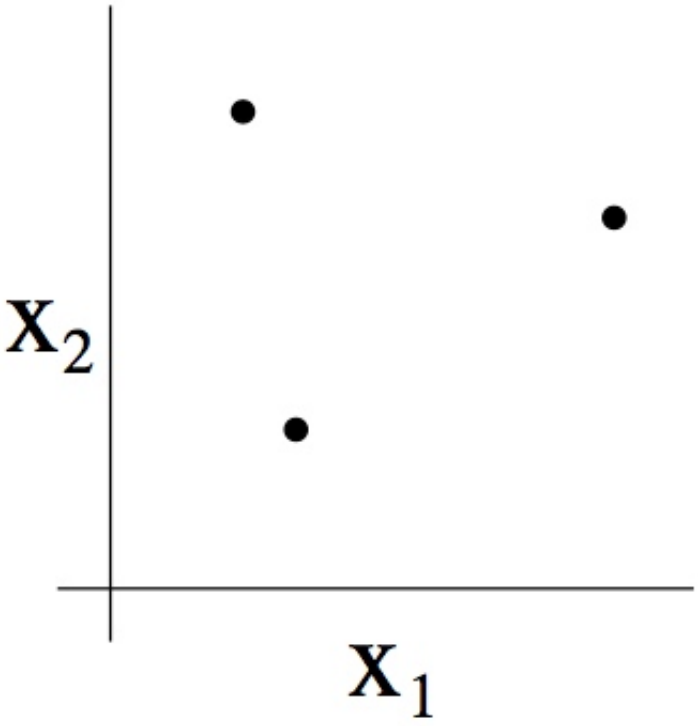
where d is represented by 64-bit float number. And if we plug it into $O(\frac{1}{\gamma^2} \log \frac{k}{\delta})$, in order to get this sort of guarantee, we should satisfy

$$\begin{aligned} m &\geq O\left(\frac{1}{\gamma^2} \log \frac{2^{64d}}{\delta} \right) \\ &= O\left(\frac{d}{\gamma^2} \log \frac{1}{\delta} \right) \end{aligned}$$

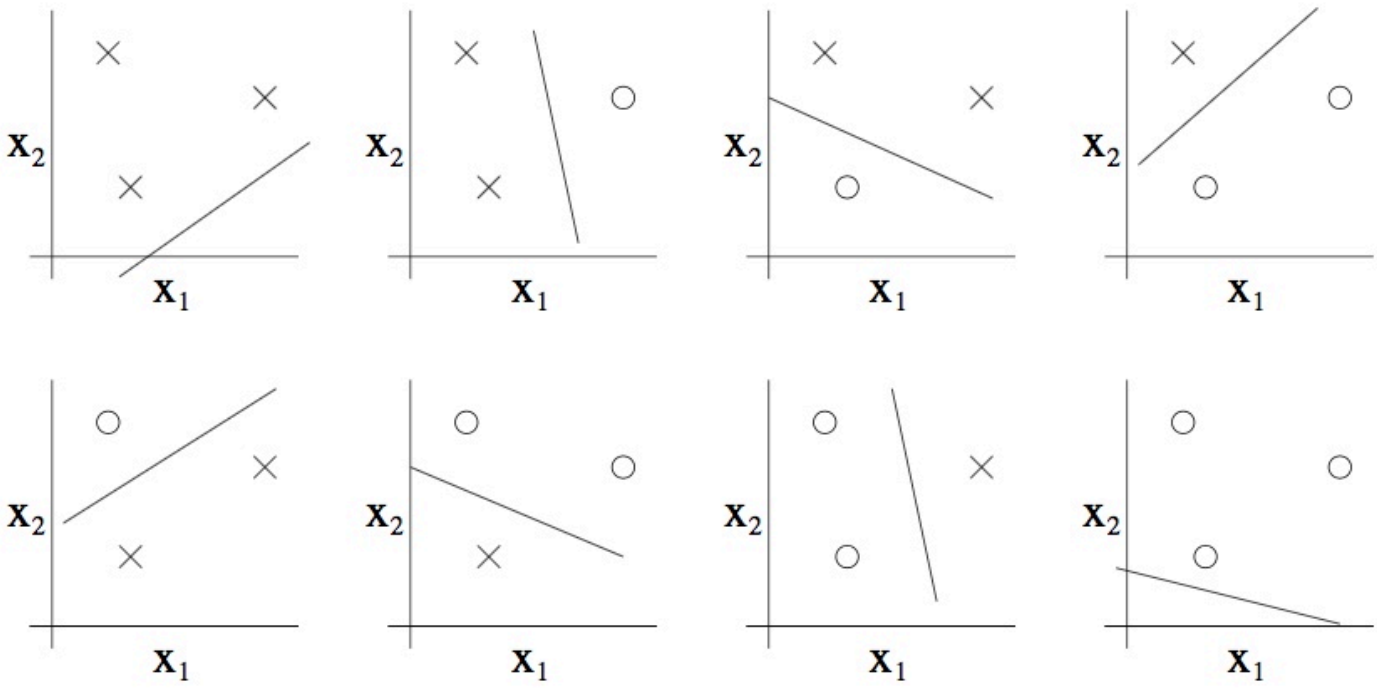
Given is a set $S = \{x^{(1)}, \dots, x^{(d)}\}$, we say $|\mathcal{H}|$ shatters S if $\mathcal{H}$ can realize any labeling on it.

Shatter

$\mathcal{H} = \{\text{linear clasifier in 2D}\}$, and S is

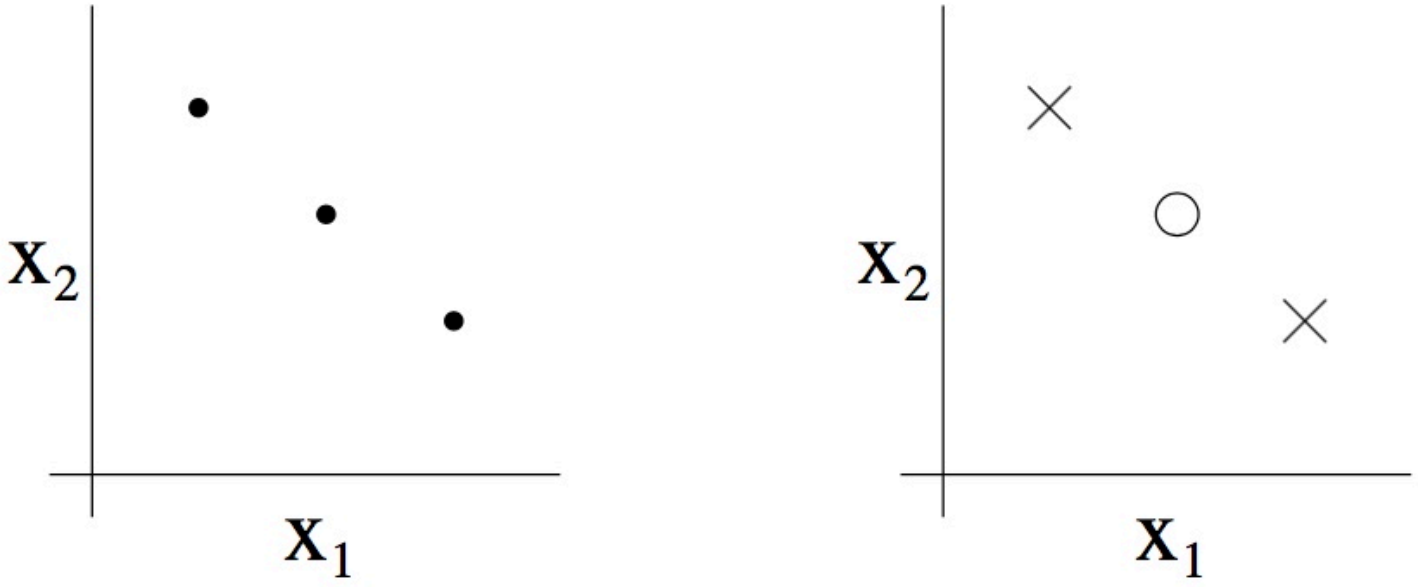


so there is 8 possible labelings that computes with these 3 points



we say $\mathcal{H}$ can shatters this set.

Consider another situation, if S is



we can't found a linear boundary to separate this set, so we say $\mathcal{H}$ does not shatters this set.

VC(Vapnik-Chervonenkis) Dimension

Given $\mathcal{H}$, $(VC(\mathcal{H}))$ is the size of the larger set that is shattered by this set by $\mathcal{H}$.

E.g. if $\mathcal{H} = \{\text{linear clasifier in 2D}\}$, then $VC(\mathcal{H}) = 3$.

More generally, in n-dimensions, $VC(\{\text{linear classifiers in n-dimension}\}) = n + 1$.

Theorem: Let $\mathcal{H}$ be given, and let $VC(\mathcal{H}) = d$, then with probability at least $1 - \delta$, we have that $\forall h \in \mathcal{H}$

$$\left|\epsilon(h) - \hat{\epsilon}(\hat{h})\right| \leq O\left(\sqrt{\frac{d}{m}\log\frac{m}{d} + \frac{1}{m}\log\frac{1}{\delta}}\right)$$

Thus, with probabilty at least $1 - \delta$, we also have

$$\epsilon(\hat{h}) \leq \epsilon(h^*) + O\left(\sqrt{\frac{d}{m}\log\frac{m}{d} + \frac{1}{m}\log\frac{1}{\delta}}\right)$$

Corollary: In order to guarantee $\epsilon(\hat{h}) \leq \epsilon(h^*) + 2\gamma$ with probability at least $1 - \delta$, it suffices that

$$m = O_{\gamma,\delta}(d)$$

Formally, this shows that sample complexity is upper bounded by the VC dimension. And for most reasonable $\mathcal{H}$, the VC dimension is usually linear in the number of parameters of our model. It turns out that in the worse case, sample complexity is also lower bounded by VC

dimension.

SVM

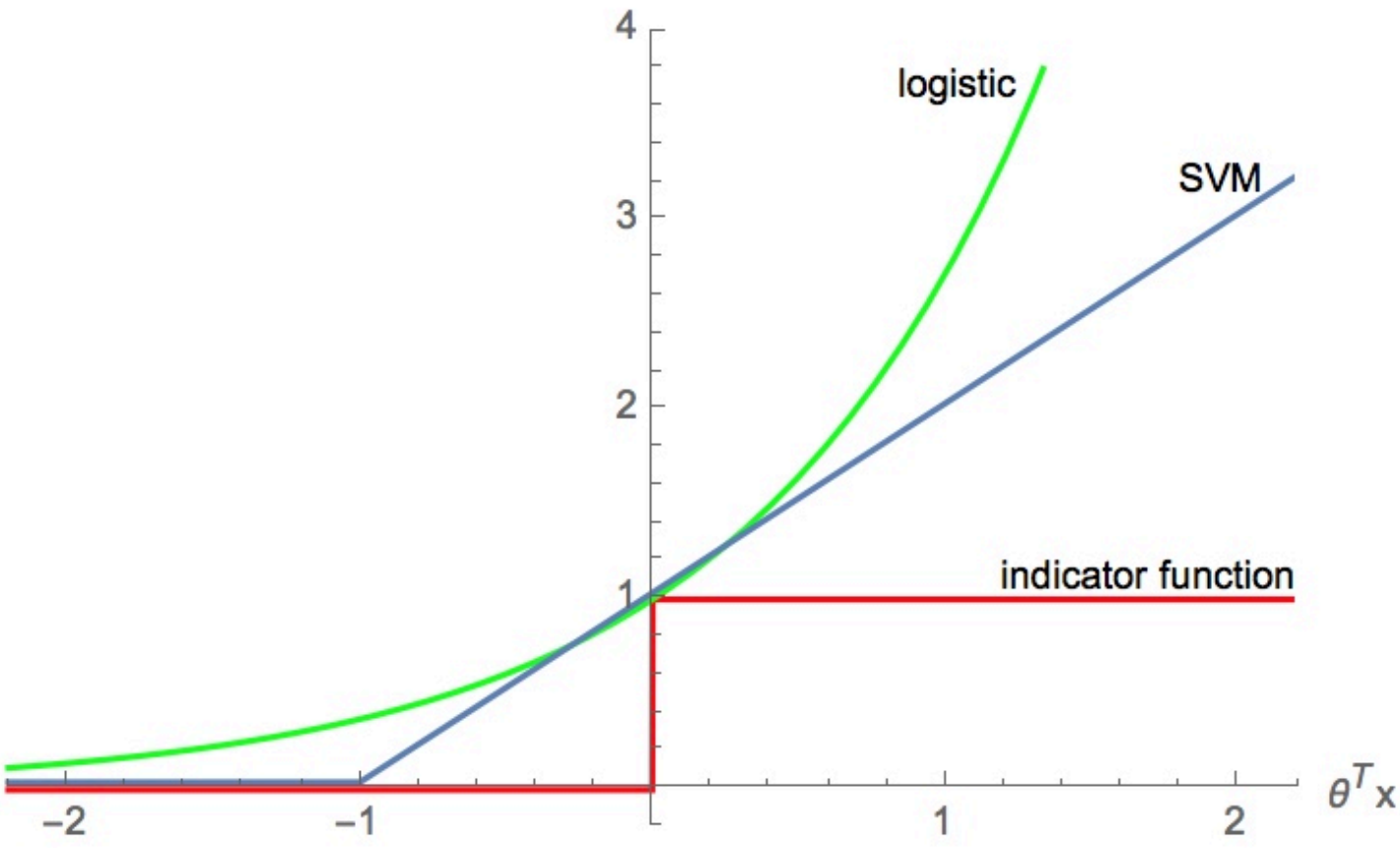
The class of linear separators with large margin actually has low VC dimension.

Given N data points in $\mathbb{R}^D : X = \{x_1, \ldots, x_N\}$ with $\|x_n\| \leq R$, we have

$$VC(\mathcal{H}) \leq \left\lceil \frac{R^2}{4\gamma^2} \right\rceil + 1$$

Approximation of ERM Algorithm

We are going to focus on just one training example. Suppose we have a function



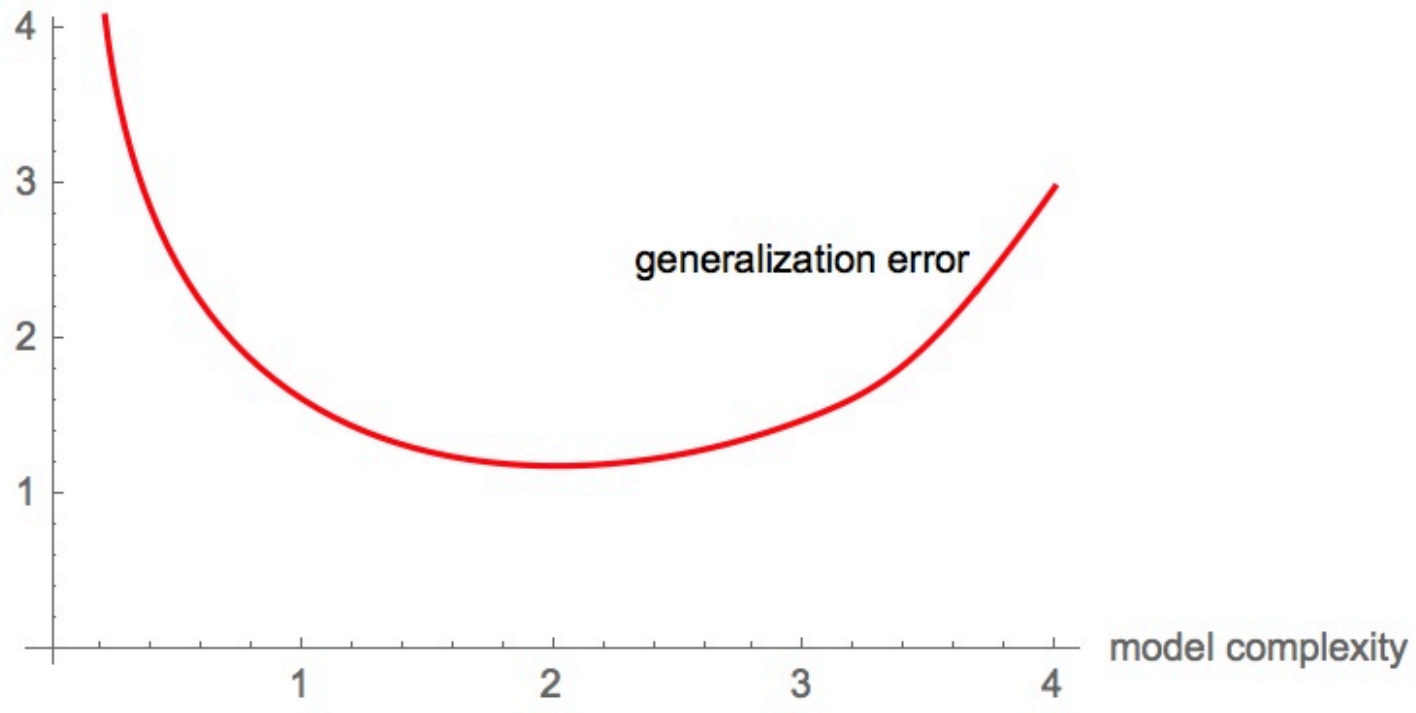
for one specific training example, it will be positive or negative. And depending on that the value of this data $\theta^T x$, we either get it right or wrong.

The indicator function(or step function) is clearly a non-convex function. And so the linear classifiers minimizing the training error is an NP-hard problem.

It turns out that both logistic regression and SVM can be viewed as using a convex

approximation for this problem. And in particular, logistic regression is trying to maximize likelihood, and so it is trying to minimize $-\log p(y^{(i)} \mid x^{(i)}; \theta)$, we can plot the likelihood curve as picture above. So we can think logistic regression as trying to approximate ERM. SVM also can be viewed as trying to approximate indicator function too.

Model Selection



We are going to talk about model selection in the abstract, let's say we are trying to choose the degree of a polynomial or choose the parameters τ, which was the bandwidth parameter in locally weight linear regression. Another model selection problem is if we are trying to choose parameters C in SVM.

So, suppose we have some finite set of models $\mathcal{M} = \{M_1, M_2, M_3, \ldots\}$, so how to select an appropriate model?

Cross Validation

Hold-out Cross Validation

In hold-out cross validation, we teach a training set, and randomly split the training set into two subsets S_{train}(70%) and S_{CV}(30%). Then training the model on S_{train} and test on S_{CV}, and pick the model with the lowest error on S_{CV}.

K-fold Cross Validation

Take all data S, and then divided it into k pieces. What we'll do is repeatedly train on $k-1$ pieces, test on the remaining piece, then average over the k results. And $k=10$ is fairly common choose.

Leave-one-out Cross Validation

Same as k-fold cross validation but $k=m$.

Feature Selection

In the feature selection, we would like to select a subset of the features that may be or hopefully the most relevant ones for a specific learning problem.

With n features, then there are 2^n possible subsets, this is a huge space. So in feature selection, what we most commonly do is use various searcheristics sort of simple search algorithm.

Forward Search Algorithm(Forward Selection Algorithm)

1. Initialize $\mathcal{F}=\emptyset$
2. Repeat
    i. For $i=1,\dots,n$, try adding feature i to $\mathcal{F}$ and evaluate the model using cross validation.
    ii. Set $\mathcal{F}=\mathcal{F}\bigcap\{\text{best feature found in (i)}\}$.
3. Output best feature subset.

This algorithm is a instantiation of **wrapper model feature selection**, and the term wrapper comes from the fact that it is a procedure that wraps our learning algorithm and repeatedly makes calls to the learning algorithm to evaluate how well it does using different feature subsets.

Backward Search Algorithm(Backward Selection Algorithm)

1. Initialize $\mathcal{F}=\{1,\dots,n\}$
2. Repeat delete one features at a time.
3. Output best feature subset.

Filter Method

Filter feature selection methods give heuristic, but computationally much cheaper, ways of choosing a feature subset.

For each feature i, compute some measure of how informative x_i is about y.

There is one other informative measure that used very commonly which is called **mutual information**

$$\begin{aligned}\mathrm{MI}(x_i, y) &= \sum_{x_i \in \{0,1\}} \sum_{y \in \{0,1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)} \\ &= \mathrm{KL}(p(x_i, y) || \, p(x_i)p(y))\end{aligned}$$

Having chosen some measure like mutual information or something else, then we pick the top k features, and we can choose k use cross validation.