3: Continuous Inputs

CS1420: Machine Learning

Michael L. Littman
Spring 2018

http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/ . Chapter 19, 22.2, 6.

Context

Inputs: none, finite set, binary vector. Outputs: bit, finite set

Representations: Decision trees, decision stumps, permutations, Bayesian, logistic function

Loss: empirical loss, entropy, squared loss, (one minus) maximum likelihood.

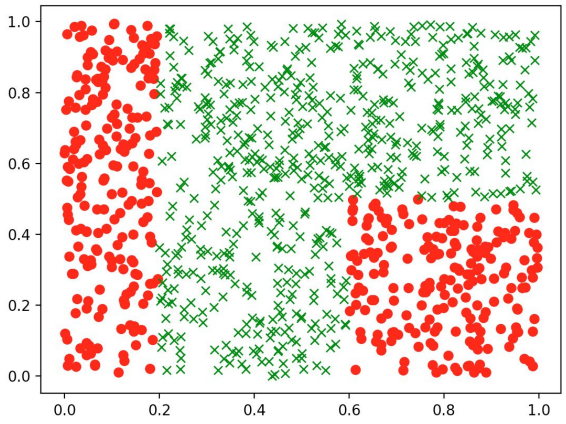
Optimizers: Exhaustive comparisons, greedy search, Hungarian matching, convex gradient descent, direct maximum likelihood via counting.

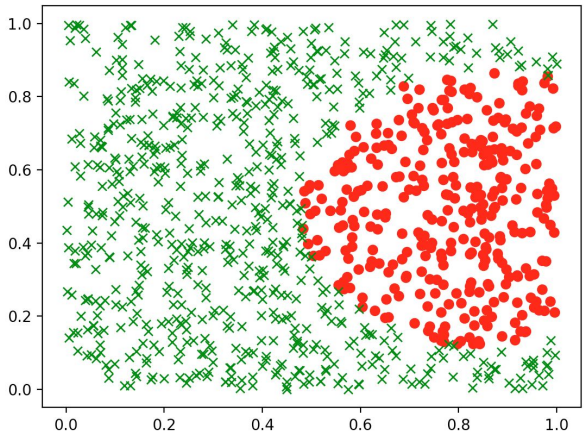
New: Continuous inputs.

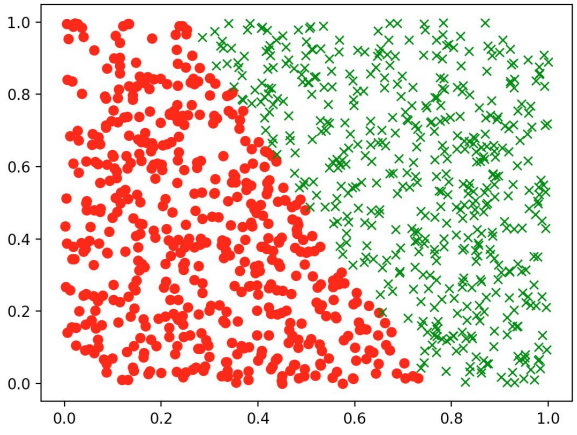
Continuous Inputs

Input: $[0,1]^d$ instead of $\{0,1\}^d$.

Let's think about generalizing the algorithms we already know to the case where inputs are $[0,1]^2$. (1) Logistic regression. (2) Naive Bayes. (3) Decision trees.







Decision Trees with Continuous Attributes

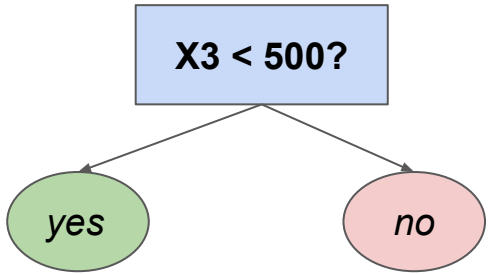
Can’t split on all possible values of an attribute.

But, can make them binary by considering ranges.

Continuous set of possible values!?

Only need to consider splitting at the values in the training data (with regard to ERM)!

(Still more splits to consider than in the binary case, but way fewer than infinity.)



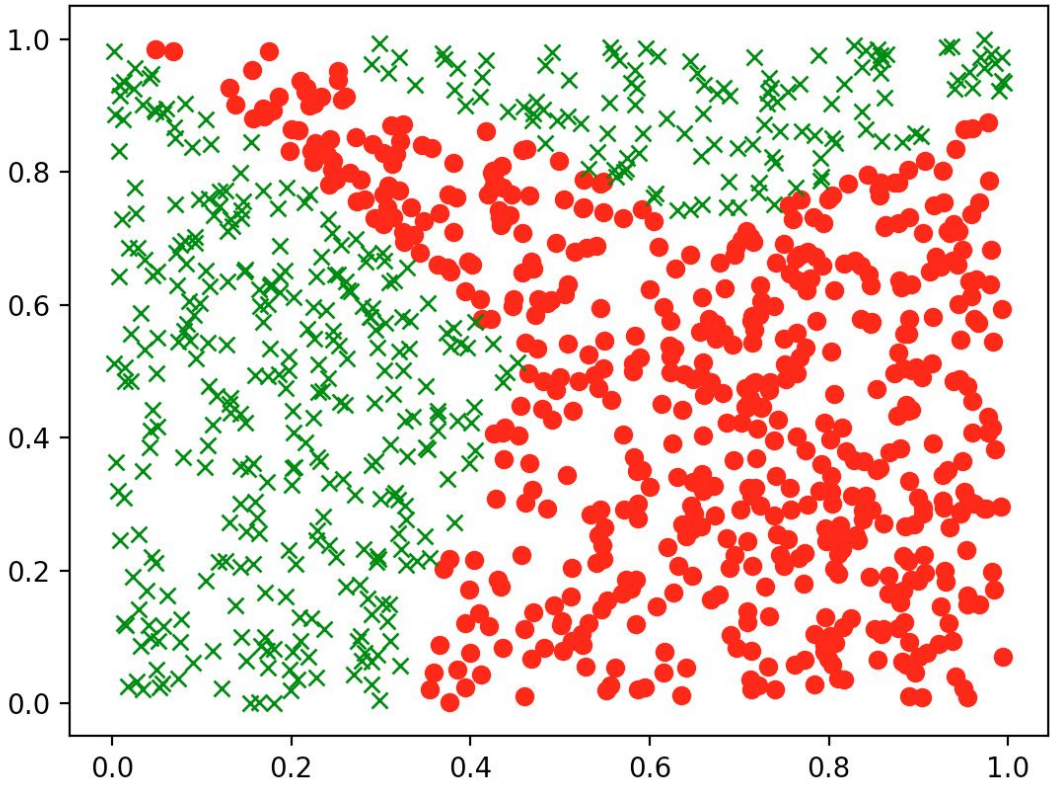
New Idea: Nearest Neighbors

$h_S(x) = \text{argmin}_y \min_{(xl,y)\in S} ||x\text{-}x_l||$

Find the closest labeled point and return its label.

k-NN: Find the k closest points and have them “vote” on a label.

Non-parametric: No fixed set of values to be set.



New Idea: Nearest Neighbors

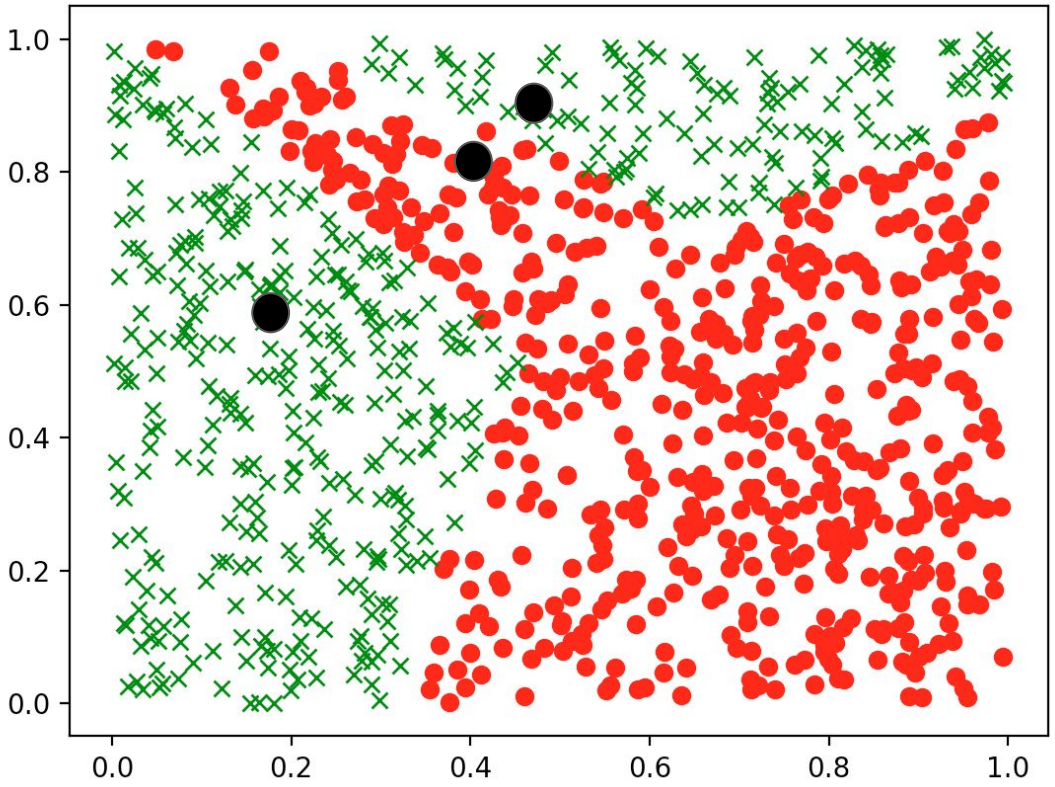
$h_S(x) = \text{argmin}_y \min_{(xl,y)\in S} ||x\text{-}x_l||$

Find the closest labeled point and return its label.

k-NN: Find the k closest points and have them “vote” on a label.

Non-parametric: No fixed set of values to be set.

Where’s the other point?



New Idea: Nearest Neighbors

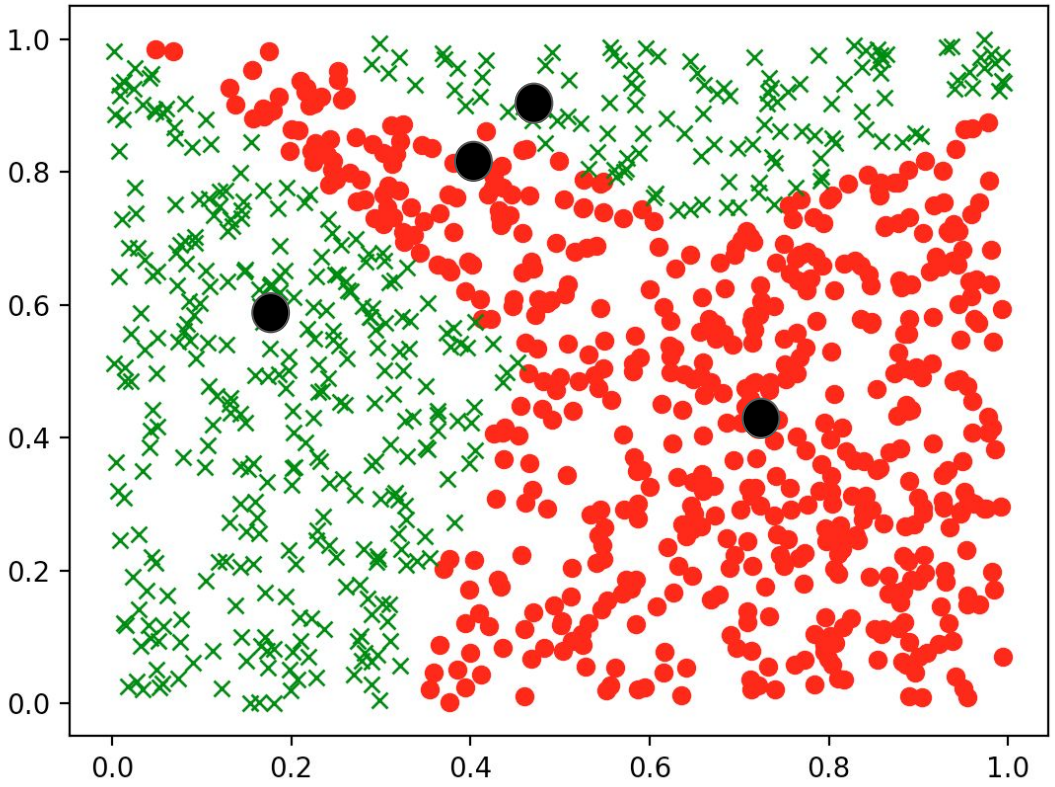
$h_S(x) = \text{argmin}_y \min_{(xl,y)\in S} ||x\text{-}x_l||$

Find the closest labeled point and return its label.

k-NN: Find the k closest points and have them “vote” on a label.

Non-parametric: No fixed set of values to be set.

Where’s the other point?



Nearest Neighbors

Can also be used when input is bit vector. What's a good metric?

Can use any of a number of distance metrics.

Can also adapt the distance metric.

Training is super cheap!

Classification is expensive.

Voronoi tessellation.

Can use distance-weighted average vote.

Generalization Bound for 1-NN

$h_S(x) = y_{\pi 1(x)}$, where $\pi_i(x)$ = id of ith closest point to x.

$\mathcal{D}$ is distribution over $\mathcal{X}$ x $\mathcal{Y}$. $\mathcal{D}_{\mathcal{X}}$ is the marginal distribution over $\mathcal{X}$.

$\eta(x) = \mathrm{P}[y = 1|x]$. Give me an x and I'll tell you how likely it is to have a label of $y = 1$.

Bayes optimal rule (minimizes 0-1 loss aka error): $h^*(x) = (\eta(x) > .5)$.

Assumption: η is c-Lipschitz, for all $x, x' \in \mathcal{X}$, $|\eta(x) - \eta(x')| \leq c\, ||x - x'||$.

Theorem: $\mathrm{E}_{S \sim \mathcal{D}m}[L_{\mathcal{D}}(h_S)] \leq 2L_{\mathcal{D}}(h^*) + 4\, c\, \sqrt{d}\, m^{-1/(d+1)}$.

Lemma: $\mathrm{E}_{S \sim \mathcal{D}m}[L_{\mathcal{D}}(h_S)] \leq 2L_{\mathcal{D}}(h^*) + c\, \mathrm{E}_{S \sim \mathcal{D}m, x \sim \mathcal{D}}\, [||x - x_{\pi 1(x)}||]$.

Proof Sketch

The probability that the labels assigned to two points x and x' are different:

$$\mathrm{P}_{y\sim\eta(x),y'\sim\eta(x')}[y\neq y'] = \eta(x)\,(1-\eta(x')) + \eta(x')\,(1-\eta(x))$$
$$= \eta(x)\,(1 - \eta(x') + \eta(x) - \eta(x)) + (\eta(x') - \eta(x) + \eta(x))\,(1-\eta(x))$$
$$= \eta(x) - \eta(x)\,\eta(x') + \eta(x)\eta(x) - \eta(x)\eta(x) - \eta(x')\eta(x) + \eta(x)\eta(x) - \eta(x)\eta(x) + \eta(x') - \eta(x) + \eta(x)$$
$$= 2\eta(x) - 2\eta(x)\eta(x) + 2\eta(x)\eta(x) - \eta(x) - 2\eta(x)\eta(x') + \eta(x')$$
$$= 2\eta(x)(1 - \eta(x)) + (\eta(x) - \eta(x'))(2\eta(x) - 1)$$
$$\leq 2\eta(x)(1 - \eta(x)) + c\,\|x-x'\| \qquad \text{[via Lipschitz and } |2\eta(x) - 1| \leq 1\text{]}$$

$$\mathrm{E}_{S\sim\mathcal{D}m}[L_{\mathcal{D}}(h_S)] \leq 2L_{\mathcal{D}}(h^*) + c\,\mathrm{E}_{S\sim\mathcal{D}m,x\sim\mathcal{D}}[\|x - x_{\pi 1(x)}\|].$$

Observations

1. As m goes to infinity, 1-NN loss goes to twice the optimal loss.

Probability matching! (Instead of maximization.)

2. This factor is reduced for larger k to $1 + \sqrt{(8/k)}$.

3. Data requirements increase exponentially with d.

4. Classification time is: $O(dm)$. Can be sped up in some cases via clever indexing.

Between Naive Bayes and Nearest Neighbors

Let’s return to binary vector inputs for a moment.

Naive Bayes: Kind of like collapsing all $y=0$ examples into a single vector, all $y=1$ examples into a second vector, and comparing them to new xs to see which matches better.

Nearest Neighbor: Keeps examples separate checks which matches better.

When is each best?

When are both problematic?

K Means Clustering

Group xs that are close to each other into a single cluster represented by their means.

But, how do we know which x belongs in each cluster?

Loss function defined on assignment of data to clusters. Group data to minimize loss.

Take k as input.

Partition xs into $C_1,...,C_k$. Each cluster has a centroid $\mu_i = \text{argmin}_\mu \sum_{x \in Ci} ||\mu_i - x||^2$. Find clustering such that $\sum_i \sum_{x \in Ci} ||\mu_i - x||^2$ is minimized.

Optimizing K Means Criterion

NP hard to minimize (or even approximate).

But, a powerful local search technique can be used.

Randomly choose initial centroids: $\mu_i = x \sim S$.
Repeat until convergence:

$\forall i,\ C_i = \{x \in S: i = \text{argmin}_j \|x - \mu_j\|^2\}$ Use centers to define clusters

$\forall i,\ \mu_i = 1/|C_i| \sum_{x \in Ci}\ x$ Use clusters to define centers

Each iteration does not increase the k means criterion: $\sum_i \sum_{x \in Ci} \|\mu\text{-}x\|^2$.

Eventually, no improvements are made.

How Many Clusterings are possible?

We have n points and k possible clusters to stick them in.

How many ways can we do this assignment of points to clusters?

A. n choose k
B. n*k
C. k^n
D. n^k
E. 2^(kn)

Use Centers to Define Clusters

$\forall i,\ C_i = \{x \in S: i = \text{argmin}_j \|x - \mu_j\|^2\}$

Keeping centers fixed and (possibly) reassigning cluster for x:

Let C' be the old clustering (with x in cluster j').

Let C be the new clustering (with x in cluster j).

Improvement = old loss - new loss

$= \sum_i \sum_{x \in C'i} \|\mu_i\text{-}x\|^2 - \sum_i \sum_{x \in Ci} \|\mu_i\text{-}x\|^2$

$= \|\mu_{j'}\text{-}x\|^2 - \|\mu_j\text{-}x\|^2 = \|\mu_{j'}\text{-}x\|^2 - \min_j \|\mu_j\text{-}x\|^2 \geq 0$

Use Clusters to Define Centers

$\forall i, \mu_i = 1/|C_i| \sum_{x \in Ci} x$

Keeping clusters fixed and (possibly) recomputing center for C_i:

Let μ'_i be the old center.

Let μ_i be the new center.

Improvement = old loss - new loss

$= \sum_i \sum_{x \in Ci} ||\mu'_i - x||^2 - \sum_i \sum_{x \in Ci} ||\mu_i - x||^2$

$= \sum_{x \in Ci} ||\mu'_i - x||^2 - \sum_{x \in Ci} ||\mu_i - x||^2$

$= \sum_{x \in Ci} ||\mu'_i - x||^2 - \min_\mu \sum_{x \in Ci} ||\mu - x||^2 \geq 0$ (since mean minimizes distance)

K Means Updates Converge in Finite Time

Infinite number of possible centers, but finite set of possible clusterings: $k^{|S|}$.

Stop iterating when criterion fails to improve.

Since each iteration has an associated clustering and each iteration improves the criterion, once we leave a clustering, we can never return to it.

Eventually must exhaust all clusterings.

But, could be many iterations and could get stuck before reaching global minimum.

K Means-style Classifier

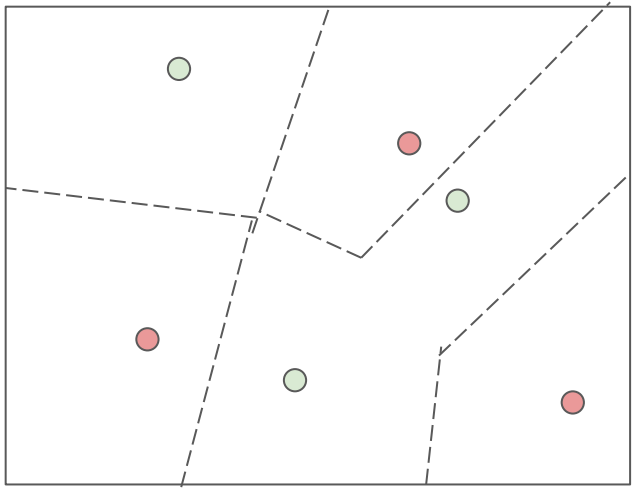
For each class, define k "centers". Call them μ^i_j where i is the label and j is the cluster within that label. We classify an input x via:

$$h(x) = \text{argmin}_i \min_j ||x - \mu^i_j||^2.$$

That is, find the closest center and returns its label.

Optimize the clusters for each label via K means.

More expressive than "Naive Bayes" style, more stable than nearest neighbor.



Sample Complexity of K Means-style Classifier

Given a dataset, there is a finite set of clusterings. But, that doesn't give us a finite hypothesis set since there are an infinite set of values for the cluster means (over all possible datasets).

Can't use PAC bound for finite hypothesis spaces (realizable).
Can't use PAC bound for finite hypothesis spaces (unrealizable).

What can we do when the hypothesis space is infinite?

In general, nothing. But, more often than you might think, something...

The Case of the 1-d Decision Stump

Back to the realizable setting. Input is $x \in [0,1]$, output is a bit.

Assume true function has the form $f(x) = 1$ *if* $x \geq \theta$, *0 otherwise.*

Here, θ is not known to the learner.

Infinite space of hypotheses of the form $h_w(x) = 1$ *if* $x \geq w$, *0 otherwise*, so our PAC bound depending on $|H|$ is hosed.

But, again, we don't have to get $w=\theta$, we just need a w with error below ε.

ERM for 1-d Decision Stump

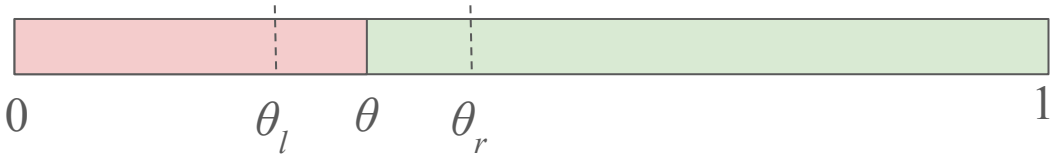
Data:

0.000	0.015	0.047	0.056	0.153	0.208	0.279	0.520	0.711	0.951
0	0	0	0	0	0	0	0	1	1

$h(x) = 1$ *if* $x \geq w$, *0 otherwise*.

Any $w \in (0.520, 0.711]$. For simplicity, let's take $w = \min_{(x,1) \in S} x$.

Epsilon Error



If our learning algorithm picks $w = \theta_r$, what's the error?

A. $\Pr_{x\sim\mathcal{D}}[x \in [0, 1]]$
B. $\Pr_{x\sim\mathcal{D}}[x \in [\theta_l, \theta_r]]$
C. $\Pr_{x\sim\mathcal{D}}[x \in [\theta_l+\varepsilon, \theta]]$
D. $\Pr_{x\sim\mathcal{D}}[x \in [\theta, \theta_r]]$
E. $\Pr_{x\sim\mathcal{D}}[x \in [\theta_r, 1]]$

Building a Bound

Define θ_r such that $\Pr_{x \sim \mathcal{D}}[x \in [\theta, \theta_r]] = \varepsilon$.

If we get a sample that includes an $x \in [\theta, \theta_r]$, we win. Why? We'll choose w to be the smallest such x. Then, we'll get anything bigger than or equal to w or anything below θ correct, and the chance of being asked something between θ and w is less than that of seeing something between θ and θ_r, which is ε.

What's the probability, in a sample of size m, that we'll get something in this range? $1-(1-\varepsilon)^m$. Using the same reasoning used in the finite hypothesis realizable case, we get that it's sufficient to use $m \geq \log(1/\square)/\varepsilon$.

Intuition

What made it so we could pick an ε-accurate hypothesis out of an infinitely large set?

Could we use the same argument for an arbitrary function in 1-d?

We're using the fact that the hypothesis class is highly structured. There's really just one degree of freedom (θ) and the data we are getting is helping us narrow down its value.

Intuitively, the *dimension* of the hypothesis space is important.

Shattering

A hypothesis class H *shatters* a finite set $C \subseteq \mathcal{X}$ if, for every possible assignment of outputs to the points in C, there's some $h \in H$ that induces it.

H: $h_c(x) = 1$ *if* $|x\text{-}c| < 0.2$, 0 *otherwise*.

$C_1 = \{ 0.3, 0.5 \}$; $C_2 = \{ 0.4, 0.9 \}$

H shatters C_1: $h_{0.0}(x) = (0, 0)$; $h_{0.6}(x) = (0,1)$; $h_{0.2}(x) = (1,0)$; $h_{0.4}(x) = (1,1)$.

H doesn't shatter C_2: $h_{0.0\text{-}0.2,0.6\text{-}0.7}(x) = (0, 0)$; $h_{0.2\text{-}0.6}(x) = (1, 0)$; $h_{0.7\text{-}1.0}(x) = (0, 1)$.
No hypothesis can achieve $(1,1)$.

Vapnik-Chervonenkis Dimension

VCdim(H): The maximal size set C such that H shatters C.

There exists a C such that |C| = VCdim(H) and C is shattered by H. There is NO C such that |C| = VCdim(H)+1 and C is shattered by H.

H_1: $h_w(x) = 1$ *if* $x \geq w$, *0 otherwise*.

H_2: $h_c(x) = 1$ *if* $|x\text{-}c| < 0.2$, *0 otherwise*.

VCdim(H_1)?

A. No idea. B. At least 1. C. Exactly 1. D. Exactly 2. E. No more than 2.

Vapnik-Chervonenkis Dimension

VCdim(H): The maximal size set C such that H shatters C.

H_1: $h_w(x) = 1$ *if* $x \geq w$, *0 otherwise*.

H_2: $h_c(x) = 1$ *if* $|x\text{-}c| < 0.2$, *0 otherwise*.

VCdim(H_1)?

$C = \{ 0.5 \}$ can be shattered: $h_{0.6}(x) = (0)$; $h_{0.4}(x) = (1)$

No pair of xs can be shattered, since we can't make the larger one 0 and the smaller one 1.

Vapnik-Chervonenkis Dimension

VCdim(H): The maximal size set C such that H shatters C.

H_1: $h_w(x) = 1$ *if* $x \geq w$, *0 otherwise*.

H_2: $h_c(x) = 1$ *if* $|x\text{-}c| < 0.2$, *0 otherwise*.

VCdim(H_2)?

A. No idea.
B. At least 1.
C. Exactly 1.
D. Exactly 2.
E. No more than 2.

Vapnik-Chervonenkis Dimension

VCdim(H): The maximal size set C such that H shatters C.

H_1: $h_w(x) = 1$ *if* $x \geq w$, *0 otherwise*.

H_2: $h_c(x) = 1$ *if* $|x\text{-}c| < 0.2$, 0 *otherwise*.

VCdim(H_2)?

Already showed we can shatter 2.

No triple of xs can be shattered, since we can't make the middle one 0 and the outer ones 1.

VC Theorem

Let H be a hypothesis class of functions from a domain $\mathcal{X}$ to $\{0, 1\}$ and let the loss function be the 0−1 loss (error). Assume that $\mathrm{VCdim}(H) = d < \infty$. Then, there is an absolute constant C such that H is agnostic PAC learnable (ε accurate with probability at least $1\text{-}\delta$) with m samples, where: $m \geq C\,(d + \log(1/\delta))/\varepsilon^2$.

Compare to finite hypothesis case: $m \geq \log(2|H|/\delta)/(2\varepsilon^2)$; $m \geq C'\,(\log(|H|)+\log(1/\delta))/\varepsilon^2$.

Sample Complexity of K Means-style Classifier

Recall: For each class, define k "centers". Call them μ^i_j where i is the label and j is the cluster within that label. We classify an input x via:

$$h(x) = \text{argmin}_i \min_j ||x - \mu^i_j||^2.$$

That is, find the closest center and returns its label.

(Would be good to choose μ^i_j to minimize training error, but perhaps K means clustering is faster?)

What's the VC dimension? At least k. But, also no more than $2k$.