MANAGEMENT TRAINING
WHAT WOULD YOU DO IF YOU MADE A HUGE, INCREDIBLY STUPID MISTAKE?
I WOULD TRY TO LEARN FROM IT.
DID YOU LEARN ANYTHING FROM YOUR ANSWER?

DILBERT: © Scott Adams/Dist. by United Feature Syndicate, Inc

Source of cartoon: https://www.sciencedirect.com/science/article/pii/S136466130800137X

Is Learning Feasible?

Leandro L. Minku

Overview

- Is learning feasible?
    - Hoeffding Inequality
    - Generalisation Bound
- Examples of kernel functions
- Kernel machines

Is Learning Feasible?

Supervised Learning Problem

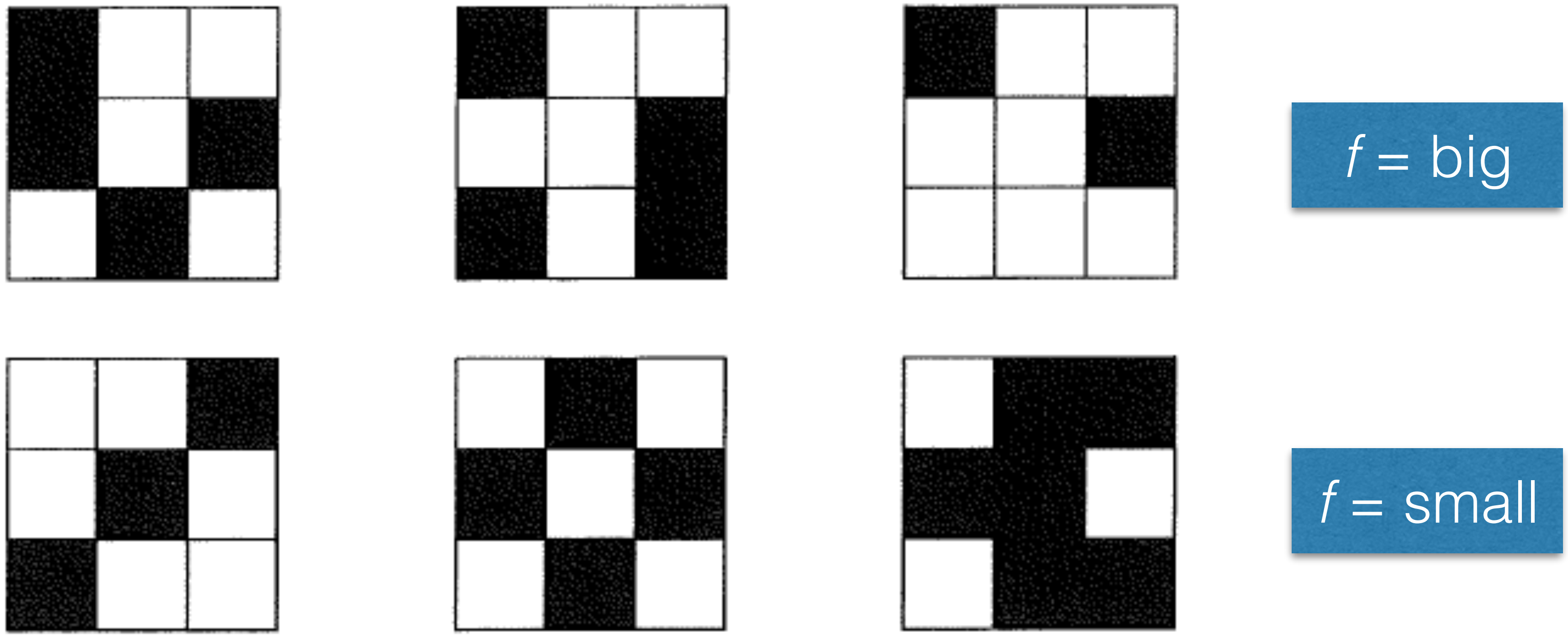
- Given a set of training examples

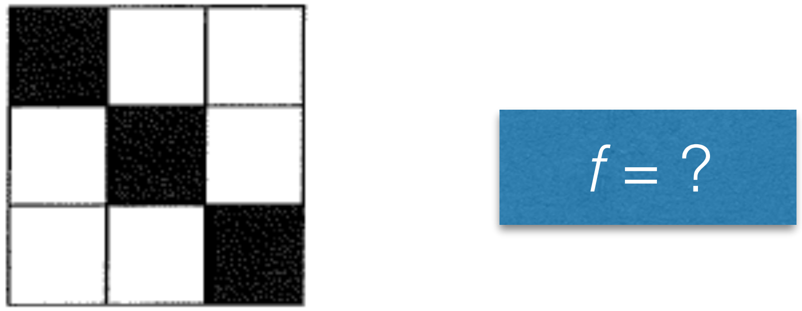
  $$\mathcal{T} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$$

  where $(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}$ are drawn i.i.d. (independently and identically distributed) from a fixed albeit unknown joint probability distribution $p(\mathbf{x}, y) = p(y \mid \mathbf{x})p(\mathbf{x})$.

- Goal: to learn a function g able to generalise to unseen (test) examples of the same probability distribution $p(\mathbf{x}, y)$.
  - $g : \mathcal{X} \to \mathcal{Y}$, mapping input space to output space.
  - g as a probability distribution approximating $p(y \mid \mathbf{x})$.

Exercise





Is Learning Feasible?

Can limited training data reveal something about unseen test examples?

Answer: we can infer something outside $\mathscr{T}$ using only $\mathscr{T}$, but in a probabilistic way.

Hoeffding Inequality

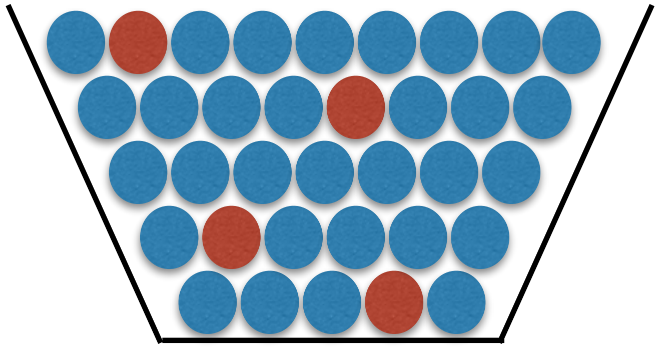
Let $z_1, \ldots, z_N$ be random independent, identically distributed random variables, such that $0 \leq z_i \leq 1$.

$$P\left(\left|\frac{1}{N}\sum_{i=1}^{N} z_i - E_{z\sim p(z)}[z]\right| > \epsilon\right) \leq 2e^{-2\epsilon^2 N} \qquad \text{for any } \epsilon > 0.$$

Hoeffding Inequality

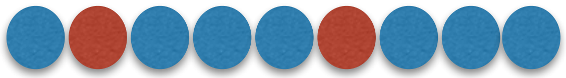
Let $z_1, \ldots, z_N$ be random independent, identically distributed Bernoulli random variables.

$$P\left(\left|\quad \nu \quad - \quad \mu \quad\right| > \epsilon\right) \le 2e^{-2\epsilon^2 N} \qquad \text{for any } \epsilon > 0.$$



μ = probability of red marbles

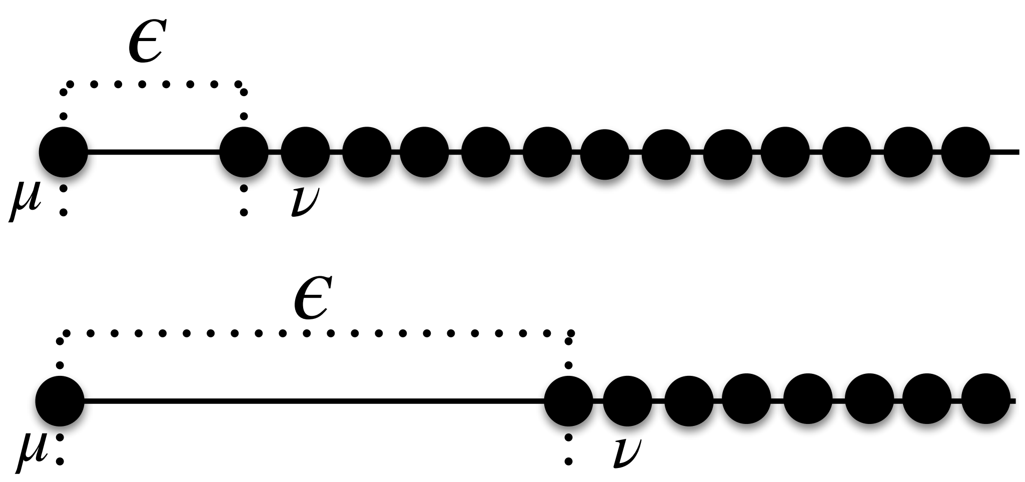
("actual value")



ν = fraction of red marbles in the sample

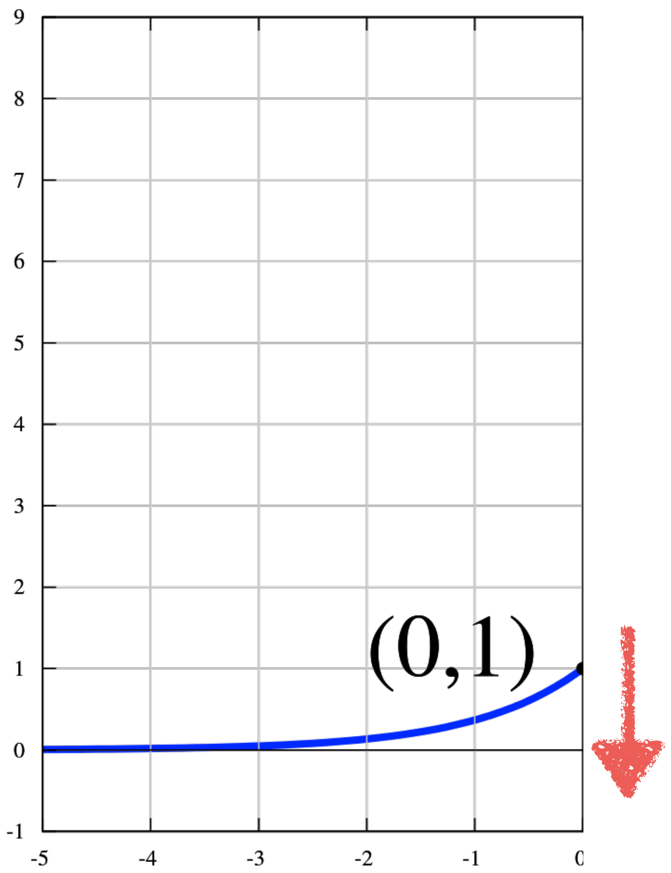
("estimated value")

Hoeffding Inequality



$$P\left(\left|\nu - \mu\right| > \epsilon\uparrow\right) \leq 2e^{\boxed{-2\epsilon^2 N}}\downarrow$$

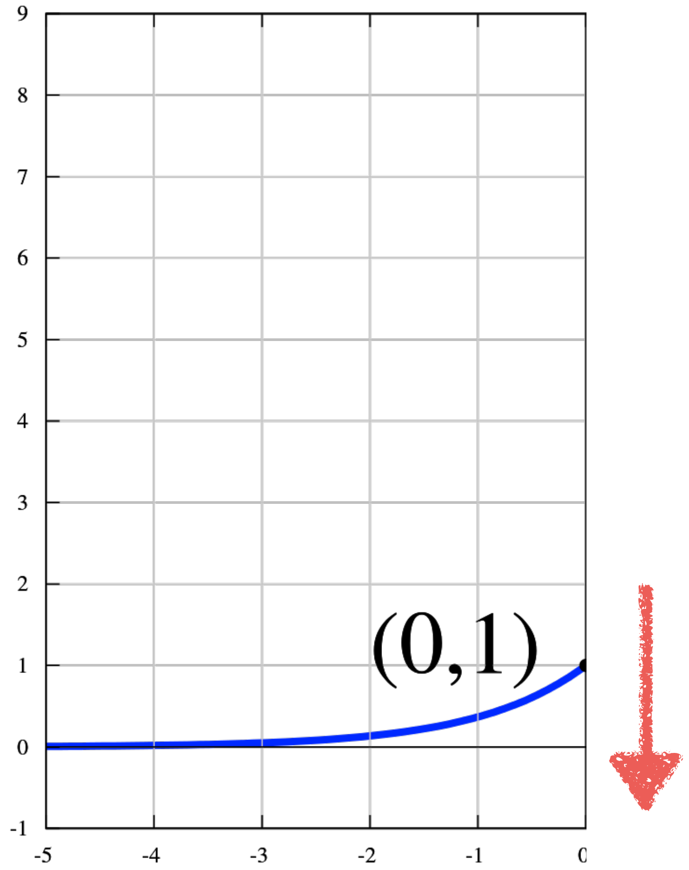
Probability that actual and estimated values are different by more than ϵ



Chances of the differences being larger are bounded by smaller probability values.

Hoeffding Inequality

$$P\left(\left|\nu-\mu\right| > \epsilon\right) \le 2e^{-2\epsilon^2 N \uparrow}\downarrow$$



Probability that actual and estimated values are different by more than ϵ

Larger sample sizes reflect smaller chances that the difference between actual and estimated value is larger than ϵ.

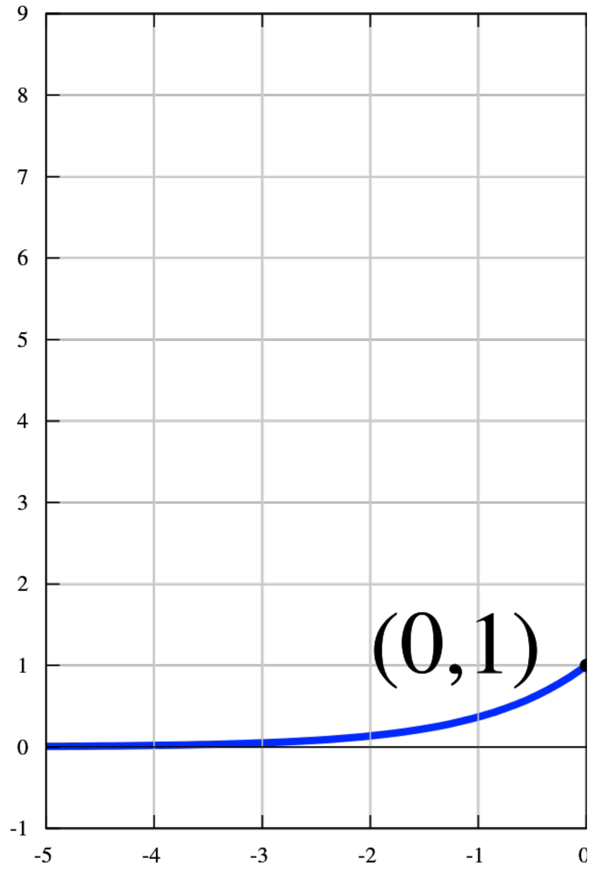
Hoeffding Inequality

This is a bound on the maximum value of the probability, for any underlying distribution.

It is not a tight bound, but useful for understanding machine learning and its feasibility.

$$P\left(\left|\nu-\mu\right| > \epsilon\right) \le \boxed{2}e^{-2\epsilon^2 N}$$

Probability that actual and estimated values are different by more than ϵ



Error is a Random Variable

$$P\left(\left|\nu - \mu\right| > \epsilon\right) \leq 2e^{-2\epsilon^2 N}$$

Error estimated on the training set

Generalisation error across all possible examples

$$E_{in}(h) = \frac{1}{N}\sum_{i=1}^{N} 1(h(\mathbf{x}^{(n)}) \neq f(\mathbf{x}^{(n)}))$$

$$E_{out}(h) = P_{\mathbf{x}\sim p(\mathbf{x})}(h(\mathbf{x}) \neq f(\mathbf{x}))$$

Consider that each individual error is a Bernoulli random variable.

$$P\left(\left|E_{in}(h) - E_{out}(h)\right| > \epsilon\right) \leq 2e^{-2\epsilon^2 N}$$

h has to be fixed beforehand to calculate $E_{in}(h)$ based on training examples drawn i.i.d., so that $1(h(\mathbf{x}^{(n)}) \neq f(\mathbf{x}^{(n)}))$ is i.i.d.. We can't choose g based on these training examples and apply this inequality.

Error is a Random Variable

$$P\left(\left|\nu - \mu\right| > \epsilon\right) \le 2e^{-2\epsilon^2 N}$$

Error estimated on the training set

$$E_{in}(h) = \frac{1}{N}\sum_{i=1}^{N} 1(h(\mathbf{x}^{(n)}) \neq f(\mathbf{x}^{(n)}))$$

Generalisation error across all possible examples

$$E_{out}(h) = P_{\mathbf{x}\sim p(\mathbf{x})}(h(\mathbf{x}) \neq f(\mathbf{x}))$$

Consider that each individual error is a Bernoulli random variable.

~~$P\left(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right) \le 2e^{-2\epsilon^2 N}$~~

h has to be fixed beforehand to calculate $E_{in}(h)$ based on training examples drawn i.i.d., so that $1(h(\mathbf{x}^{(n)}) \neq f(\mathbf{x}^{(n)}))$ is i.i.d.. We can't choose g based on these training examples and apply this inequality.

Considering the Hypothesis Set

$$P\left(\left|E_{in}(h) - E_{out}(h)\right| > \epsilon\right) \leq 2e^{-2\epsilon^2 N}$$

Consider a finite hypothesis set $\mathscr{H} = \{h_1, h_2, \ldots, h_M\}$

$$P(\ |E_{in}(g) - E_{out}(g)|\ > \epsilon\ ) \quad \overset{\leq}{\longrightarrow} \quad P(\ \ |E_{in}(h_1) - E_{out}(h_1)|\ > \epsilon$$

or $|E_{in}(h_2) - E_{out}(h_2)|\ > \epsilon$

$\ldots$

or $|E_{in}(h_M) - E_{out}(h_M)|\ > \epsilon\ \ )$

Rule of probabilities:
if $z_1 \rightarrow z_2$, then $P(z_1) \leq P(z_2)$

Considering the Hypothesis Set

$$P\left(\left|E_{in}(h) - E_{out}(h)\right| > \epsilon\right) \leq 2e^{-2\epsilon^2 N}$$

Consider a finite hypothesis set $\mathscr{H} = \{h_1, h_2, \ldots, h_M\}$

$$\begin{aligned} P(\ |E_{in}(g) - E_{out}(g)| > \epsilon\ ) \quad &\leq \quad P(\ \ |E_{in}(h_1) - E_{out}(h_1)| > \epsilon \\ &\quad \text{or } |E_{in}(h_2) - E_{out}(h_2)| > \epsilon \\ &\quad \ldots \\ &\quad \text{or } |E_{in}(h_M) - E_{out}(h_M)| > \epsilon\ \ ) \end{aligned}$$

Rule of probabilities (union bound):
$P(z_1 \text{ or } z_2 \text{ or } \ldots z_M) \leq P(z_1) + P(z_2) + \ldots + P(z_M)$

Considering the Hypothesis Set

$$P\left(\left|E_{in}(h) - E_{out}(h)\right| > \epsilon\right) \leq 2e^{-2\epsilon^2 N}$$

Consider a finite hypothesis set $\mathscr{H} = \{h_1, h_2, \ldots, h_M\}$

$$P(\,|E_{in}(g) - E_{out}(g)| > \epsilon\,) \quad \leq \quad P(|E_{in}(h_1) - E_{out}(h_1)| > \epsilon)$$

$$+P(|E_{in}(h_2) - E_{out}(h_2)| > \epsilon)$$

$$\ldots$$

$$+P(|E_{in}(h_M) - E_{out}(h_M)| > \epsilon)$$

Rule of probabilities (union bound):
$P(z_1 \text{ or } z_2 \text{ or } \ldots z_M) \leq P(z_1) + P(z_2) + \ldots + P(z_M)$

Considering the Hypothesis Set

$$P\left(\left|E_{in}(h) - E_{out}(h)\right| > \epsilon\right) \leq 2e^{-2\epsilon^2 N}$$

Consider a finite hypothesis set $\mathscr{H} = \{h_1, h_2, \ldots, h_M\}$

$$P(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon) \quad \leq \quad \sum_{i=1}^{M} P(|E_{in}(h_i) - E_{out}(h_i)| > \epsilon)$$

$$P\left(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right) \leq 2Me^{-2\epsilon^2 N}$$

The bound above can be used with a hypothesis g chosen based on the training set.

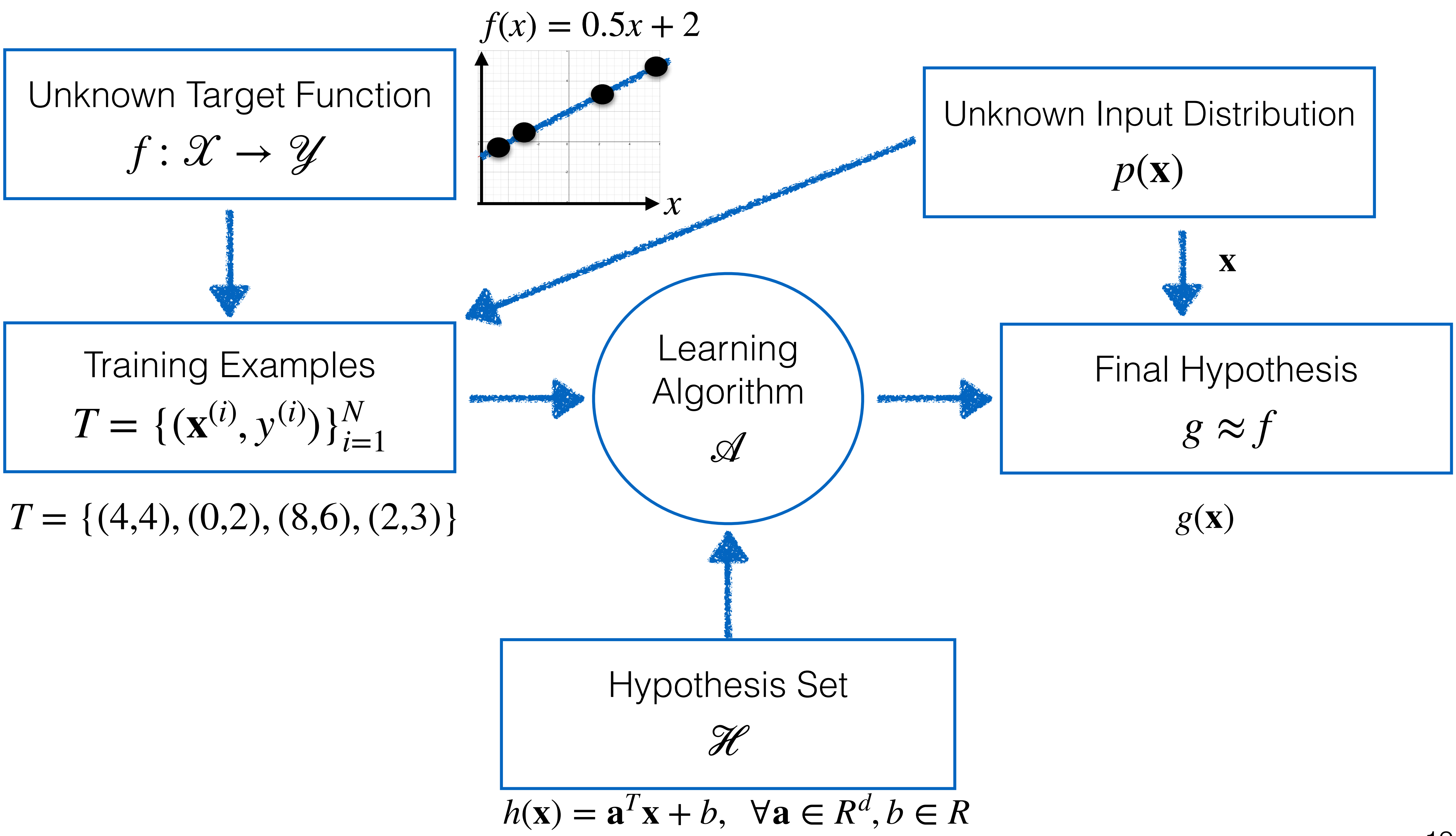
Considering the Hypothesis Set

$$P\left(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right) \leq 2Me^{-2\epsilon^2 N}$$

The bound above can be used with a hypothesis g chosen based on the training set.

Assumption: examples in $\mathcal{T}$ are drawn i.i.d. from $p(\mathbf{x})$, and so do any test examples.
They are all coming from the same underlying f.

Components of the Supervised Learning Process in View of Noise



Unknown Target Function

$f : \mathscr{X} \rightarrow \mathscr{Y}$

$f(x) = 0.5x + 2$

Unknown Input Distribution

$p(\mathbf{x})$

$\mathbf{x}$

Training Examples

$T = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$

$T = \{(4,4), (0,2), (8,6), (2,3)\}$

Learning Algorithm

$\mathscr{A}$

Final Hypothesis

$g \approx f$

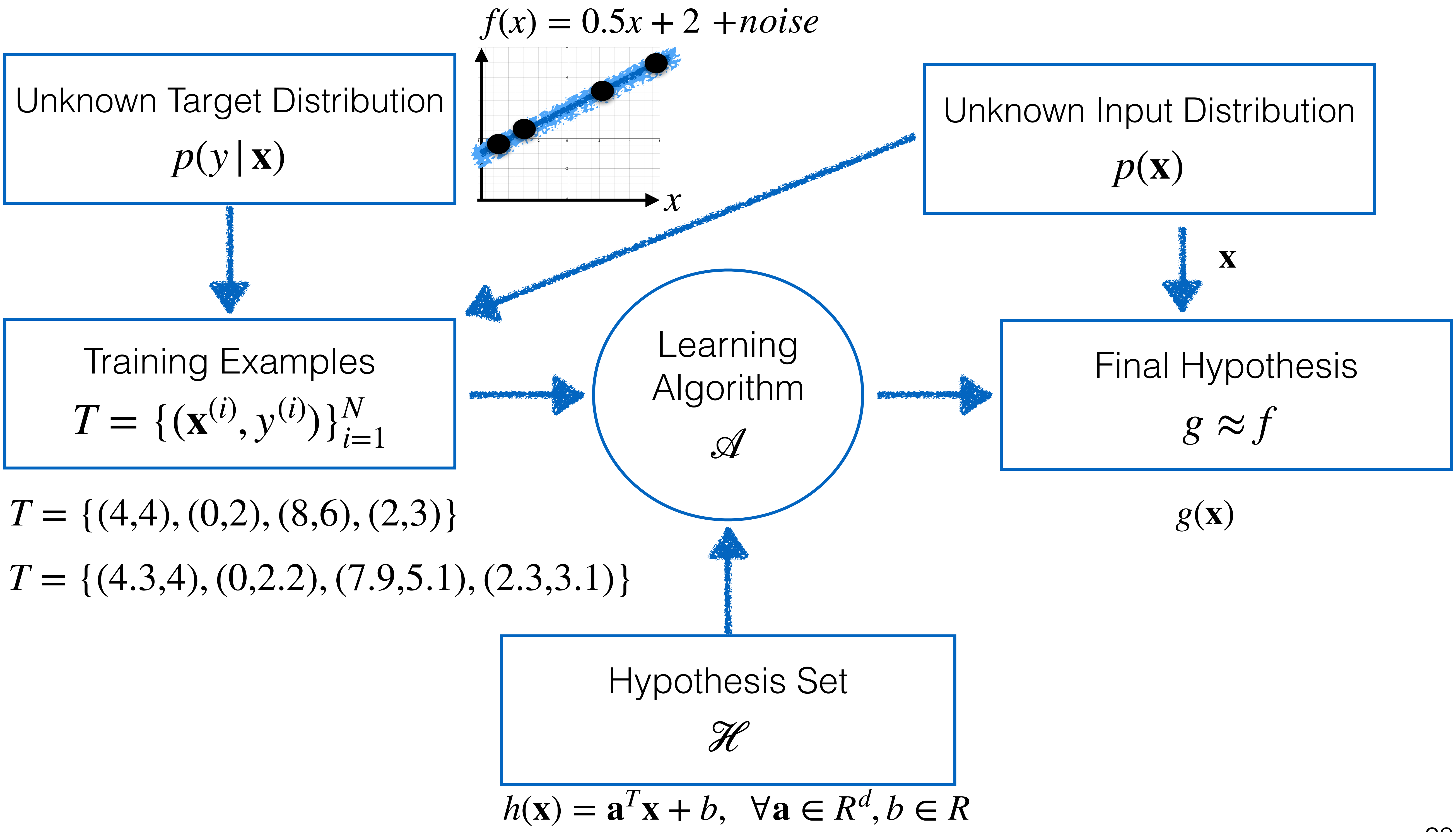
$g(\mathbf{x})$

Hypothesis Set

$\mathscr{H}$

$h(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b, \quad \forall \mathbf{a} \in R^d, b \in R$

Components of the Supervised Learning Process in View of Noise



$f(x) = 0.5x + 2 + noise$

Unknown Target Distribution

$p(y \mid \mathbf{x})$

Unknown Input Distribution

$p(\mathbf{x})$

$\mathbf{x}$

Training Examples

$T = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$

$T = \{(4,4), (0,2), (8,6), (2,3)\}$

$T = \{(4.3,4), (0,2.2), (7.9,5.1), (2.3,3.1)\}$

Learning Algorithm

$\mathscr{A}$

Final Hypothesis

$g \approx f$

$g(\mathbf{x})$

Hypothesis Set

$\mathscr{H}$

$h(\mathbf{x}) = \mathbf{a}^T\mathbf{x} + b, \quad \forall \mathbf{a} \in R^d, b \in R$

Considering the Hypothesis Set

$$P\left(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right) \leq 2Me^{-2\epsilon^2 N}$$

The bound above can be used with a hypothesis g chosen based on the training set.

Assumption: examples in $\mathcal{T}$ are drawn i.i.d. from $p(\mathbf{x}, y)$, and so do any test examples.

Note: our individual errors need to be in [0,1] and E_{out} computed accordingly. The specific inequality would be a bit different if this is not the case.

Supervised Learning Problem

- Given a set of training examples

  $$\mathcal{T} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \cdots, (\mathbf{x}^{(N)}, y^{(N)})\}$$

  where $(\mathbf{x}^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}$ are drawn i.i.d. (independently and identically distributed) from a fixed albeit unknown joint probability distribution $p(\mathbf{x}, y) = p(y \,|\, \mathbf{x})p(\mathbf{x})$.

- Goal: to learn a function g able to generalise to unseen (test) examples of the same probability distribution $p(\mathbf{x}, y)$.

  - $g : \mathcal{X} \rightarrow \mathcal{Y}$, mapping input space to output space.

  - g as a probability distribution approximating $p(y \,|\, \mathbf{x})$.

Is Learning Feasible?

Does $\mathscr{T}$ tell us something *certain* about f outside of $\mathscr{T}$? No

Does $\mathscr{T}$ tell us something *likely* about f outside of $\mathscr{T}$? Yes

$$P\left(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right) \leq 2Me^{-2\epsilon^2 N}$$

It makes sense to estimate the generalisation error based on the training error.

Larger sample sizes will likely lead to better estimation of generalisation error.

It makes sense to try and minimise the training error, but…

… a smaller training error may require a larger hypothesis set, potentially resulting in the training error being a less good approximation of the generalisation error.

There is a complex relationship between model complexity, training error and generalisation error.

Generalisation Bound

$$P\left(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right) \leq 2Me^{-2\epsilon^2 N}$$

↓

With probability at least $1 - \delta$

$$\downarrow E_{out}(g) \leq \downarrow E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$$

Smaller training error
is limiting the
generalisation error to
smaller values

Generalisation Bound

$$P\left(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right) \leq 2Me^{-2\epsilon^2 N}$$

↓

With probability at least $1 - \delta$

$$\updownarrow E_{out}(g) \leq E_{in}(g) + \updownarrow\sqrt{\frac{1}{2N\uparrow}\ln\frac{2M\uparrow}{\delta}}$$

Large training sets are limiting the generalisation error to smaller values

Large hypothesis sets are limiting the generalisation error to larger values

Generalisation Bound

$$P\left(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right) \leq 2Me^{-2\epsilon^2 N}$$

↓

With probability at least $1 - \delta$

$$E_{out}(g) \leq \downarrow E_{in}(g) + \uparrow \sqrt{\frac{1}{2N} \ln \frac{2M \uparrow}{\delta}}$$

There is a difficult relationship between model complexity, training error and generalisation error.

Obtaining the Generalisation Bound

$$P\left(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right) \leq 2Me^{-2\epsilon^2 N}$$

With probability at least $1 - 2Me^{-2\epsilon^2 N}$, $\left|E_{in}(g) - E_{out}(g)\right| \leq \epsilon$

$$\begin{tikzcd}
& -\epsilon \leq E_{in}(g) - E_{out}(g) \leq \epsilon \arrow[dl] \arrow[dr] & \\
E_{in}(g) - E_{out}(g) \geq -\epsilon & & E_{in}(g) - E_{out}(g) \leq \epsilon
\end{tikzcd}$$

$$\boxed{E_{out}(g) \leq E_{in}(g) + \epsilon}$$

$$E_{out}(g) \geq E_{in}(g) - \epsilon$$

Obtaining the Generalisation Bound

$$P\left(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right) \leq 2Me^{-2\epsilon^2 N}$$

With probability at least $1 - 2Me^{-2\epsilon^2 N}$, $E_{out}(g) \leq E_{in}(g) + \epsilon$

Setting $\delta = 2Me^{-2N\epsilon^2}$ and solving it for ϵ leads to:

With probability at least $1 - \delta$, $E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$

The Other Side of the Inequality

$$P\left(\left|E_{in}(g) - E_{out}(g)\right| > \epsilon\right) \leq 2Me^{-2\epsilon^2 N}$$

With probability at least $1 - 2Me^{-2\epsilon^2 N}$, $\left|E_{in}(g) - E_{out}(g)\right| \leq \epsilon$

$$\begin{tikzcd}
& -\epsilon \leq E_{in}(g) - E_{out}(g) \leq \epsilon \arrow[dl] \arrow[dr] & \\
E_{in}(g) - E_{out}(g) \geq -\epsilon & & E_{in}(g) - E_{out}(g) \leq \epsilon \\
E_{out}(g) \leq E_{in}(g) + \epsilon & & \boxed{E_{out}(g) \geq E_{in}(g) - \epsilon}
\end{tikzcd}$$

The Other Side of the Inequality

$$\uparrow E_{out}(h) \geq \uparrow E_{in}(h) - \sqrt{\frac{1}{2N}\ln\frac{2M}{\delta}}$$

Hypothesis with higher $E_{in}(h)$ would also have a comparatively higher $E_{out}(h)$.

Summary

- The Hoeffding Inequality can be used to discuss the feasibility of learning.
- It shows us that larger training sets will likely lead to training errors more similar to the generalisation error.
    - Reducing the training error is likely to reduce the generalisation error.
- However, to reduce the training error, we may need a larger hypothesis set.
    - Larger hypothesis sets are likely to lead to more different training and generalisation errors.
- There is a trade-off between model complexity and generalisation.
- Caveat: our analyses were considering a finite number of hypothesis, but our machine learning algorithms usually have infinite hypothesis sets.
- Next: how to consider infinite hypothesis sets?

All-Subtree Kernel

Assume T_1 and T_2 are trees that can be constructed with a given set of possible nodes, and $\mathcal{T}$ is the set of all possible trees.

$$\phi_S(T) = 1(S \in T)$$

(S is a subtree of T)

$$k(T_1, T_2) = \phi(T_1)^T \phi(T_2) = \sum_{S \in \mathcal{T}} \phi_S(T_1)\phi_S(T_2)$$

This kernel is a similarity metric that counts the number of sub-trees in common between T_1 and T_2.

All-Subtree Kernel

Assume T_1 and T_2 are trees that can be constructed with a given set of possible nodes, and $\mathcal{T}$ is the set of all possible trees.

$\phi_S(T) = 1(S \in T)$ (S is a subtree of T)

$$k(T_1, T_2) = \phi(T_1)^T \phi(T_2) = \sum_{S \in \mathcal{T}} \phi_S(T_1)\phi_S(T_2)$$

Dynamic programming can be used to compute this kernel in $O(|T_1||T_2|d_{max}^2)$, where d_{max} is the maximum number of children a node can have in these trees.

Kernels for Graphs and Other Structures

Kernels for graphs and other structures that can be decomposed into smaller sub-structures can be defined using similar ideas to the ones described for kernels for trees.

Different Terminologies

- The term kernel can also be used with other meanings.

- Some simply define a kernel as a real valued function of two arguments $k(\mathbf{x}, \mathbf{z}) \in \mathbb{R}$.

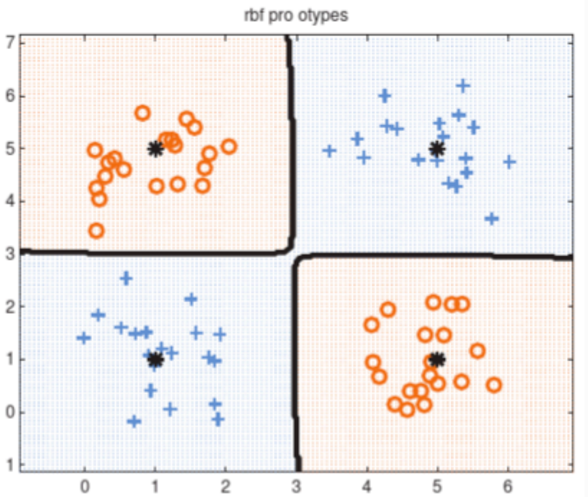
- Some simply define a kernel as a real valued function of two arguments $k(\mathbf{x}, \mathbf{z}) \geq 0$.

- The term kernel is also used to refer to a matrix which is slid across an image and multiplied with the input such that the output is enhanced in a desired manner (used in Convolutional Neural Networks).

Kernel Machines

- One can create basis expansions based on kernels, where μ_i are "centroids".

$$\phi(\mathbf{x}) = (k(\mathbf{x}, \mu_1), k(\mathbf{x}, \mu_2), \ldots, k(\mathbf{x}, \mu_D))$$

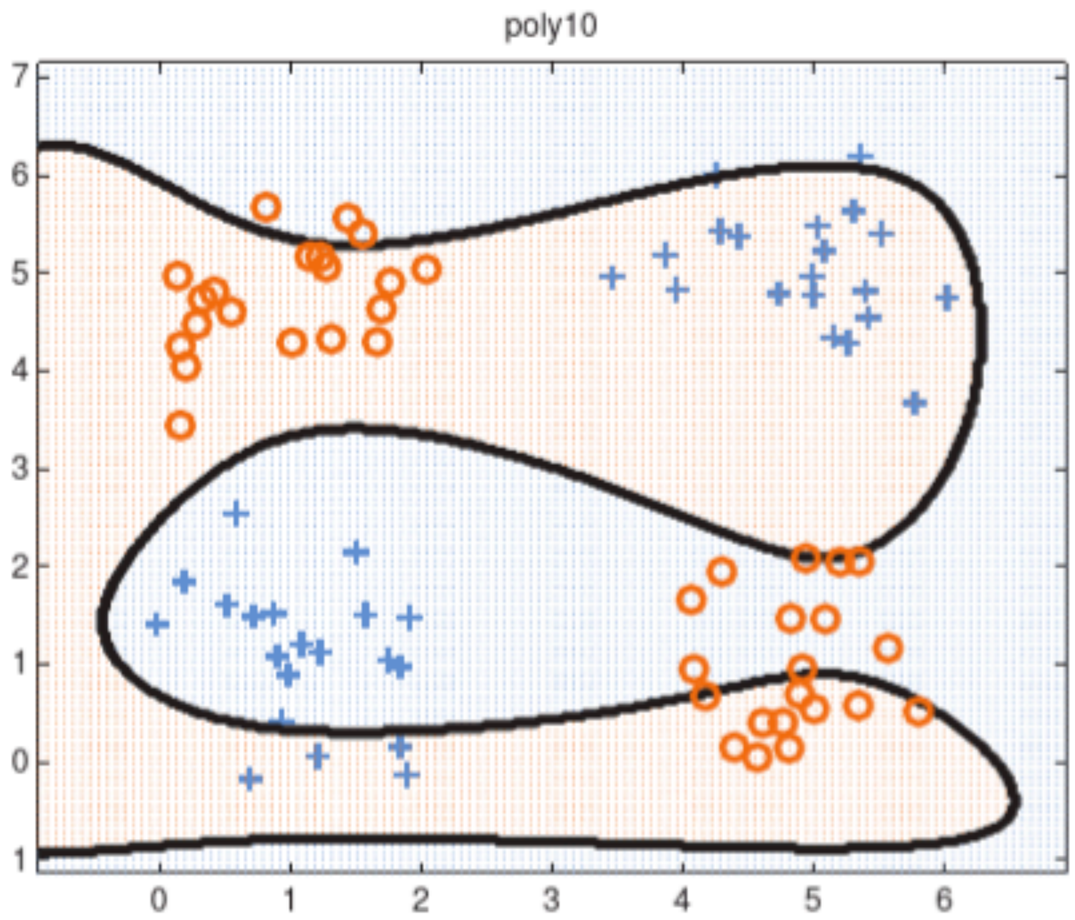


- We call this a kernelised feature vector, where the kernel typically corresponds to a similarity metric but does not need to satisfy the Mercer's condition.

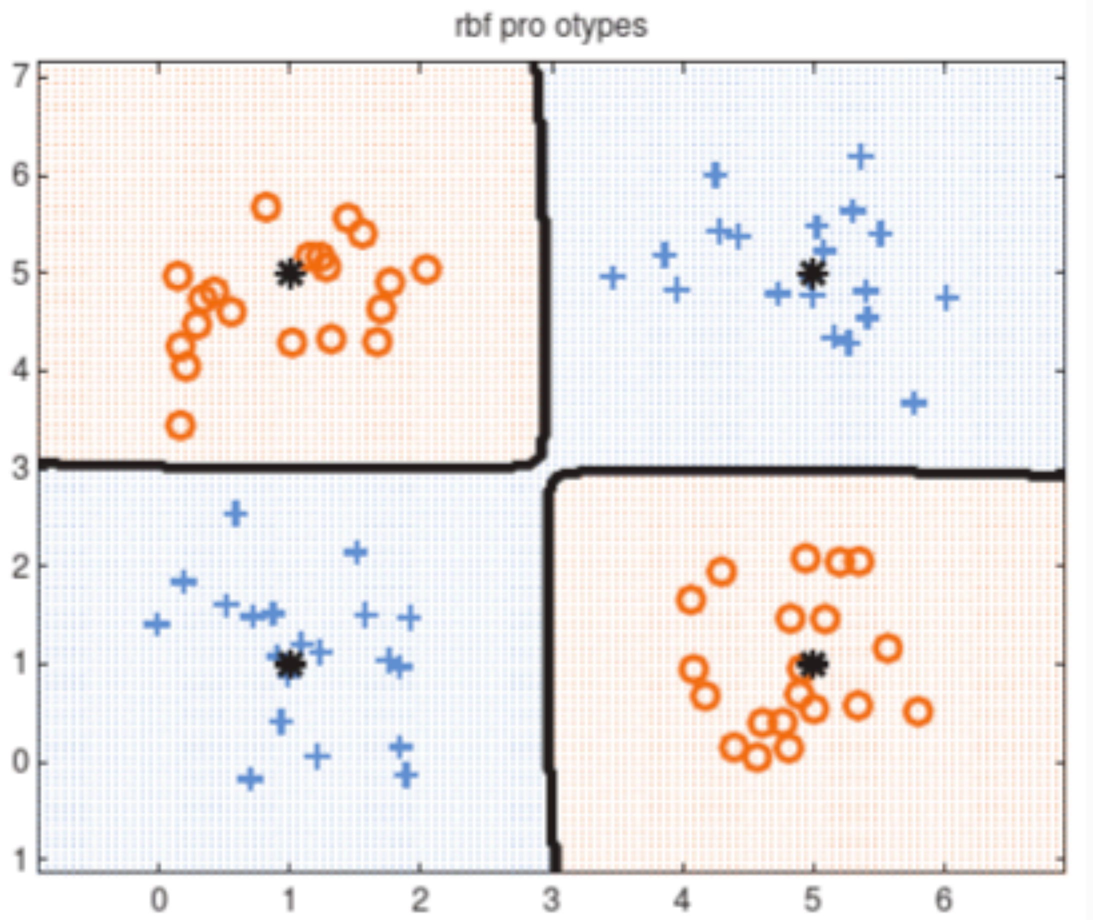
- We can then use a linear model such as linear regression or logistic regression with this basis expansion, making it non-linear.

- If the kernel is the Gaussian kernel, this gives rise to the Radial Basis Function Network.

Example of Decision Boundary for Radial Basis Function Networks



Logistic regression with polynomial embedding of degree 10



Radial Basis Function Network

Source: Murphy's Machine Learning: A Probabilistic Perspective.

Summary

- Kernels are powerful tools.
  - May enable us to better separate training examples through the feature embedding that they represent.
  - May enable us to use such feature embeddings without having to compute them.
  - May enable us to deal with different types of input features.