PAC learning, Neural Networks and Deep Learning

Abhishek Sinha
Laboratory for Information and Decision Systems

MIT

Talk at: CNRG Meeting

October 5, 2016

Outline

Outline

The formal setting : PAC Learning (Valiant, '84)

- Consider the *Binary Classification Problem* : We have m pairs of labeled training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathcal{X}$ are called *features* and $y_i \in \mathcal{Y} = \{0, 1\}$ are their labels. Examples:
  - $\mathbf{x}_i$'s are pixels (encoding of an image) and y_i's are labels $\{$*cats, dogs* $\}$.
  - $\mathbf{x}_i$'s are ASCII encoding of emails and $y_i's$ are the labels $\{$*spam, not spam* $\}$.
  - $\mathbf{x}_i$'s are medical data (ECG,EEG, CT Scan etc) and y_i's are whether a patient has a certain disease or not.
- Assumption 1: The training data is sampled i.i.d. from an unknown distribution $p_X(x)$.
- Assumption 2: The input $\mathbf{x}$ and the output y are related by an unknown deterministic function g^*, i.e., $y = g^*(x), \forall x$.
- Assumption 3: Although we don't know g^*, it is known that g^* lies in a given function class $\mathcal{C}$ (Concept Class).

For any function (hypothesis) $\psi : \mathcal{X} \to \mathcal{Y}$ in the class $\mathcal{C}$, define its **error-rate**

$$\epsilon_\psi = \mathbb{P}(\psi(X) \neq g^*(X))$$

Problem (The Learning Problem)

For any given $\epsilon, \delta > 0$, upon observing m training samples, **select** *a hypothesis $\psi \in \mathcal{C}$ such that,*

$$\mathbb{P}(\epsilon_\psi \geq \epsilon) \leq \delta$$

Sample Complexity of Learning : Finite Function Class

Algorithm (Empirical Risk Minimization (ERM)): Simply output a function $\psi \in \mathcal{C}$ which agrees on the training data, i.e., $\psi(x_i) = y_i, i = 1, 2, \ldots, m$.

Theorem (Finite Concept Classes are Learnable)

If $|\mathcal{C}| < \infty$, then ERM requires $m = \frac{1}{\epsilon} \ln \frac{|\mathcal{C}|}{\delta}$ samples to learn, irrespective of the underlying distribution $p_X(\cdot)$ and the optimal hypothesis g^*.

The above theorem tells that by minimizing the empirical risk, irrespective of the underlying unknown distribution, we can bound the true risk *w.h.p*.

Proof.

For any hypothesis $f \in \mathcal{C}$, define its error-region $\mathcal{E}_f$, i.e., the set of inputs where it disagrees with the true function g^*

$$\mathcal{E}_f = \{\boldsymbol{x} \in \mathcal{X} : f(x) \neq g^*(x)\} \tag{1}$$

Each error-region has an error-rate ϵ_f associated with it

$$\epsilon_f = \mathbb{P} E_f \tag{2}$$

Note that error-rate is implictly computed using the unknown distribution $p_X()$ of the samples.

□

Proof *contd.*

Now define the set of *Bad* hypotheses $\mathcal{B}$: hypotheses which have error-rate at least ϵ

$$\mathcal{B} = \{f \in \mathcal{C} : \epsilon_f \geq \epsilon\}$$

Hence, for any $f \in \mathcal{B}$, we have $\mathbb{P}(f(x) \neq g^*(x)) \geq \epsilon$.

Now let us compute the probability that a bad hypothesis $f \in \mathcal{B}$ is chosen by ERM. Note that, ERM will choose the function f **only if** the hypothesis agrees with g^* on the training data.
Thus, probability that $f \in \mathcal{B}$ is chosen

$$\mathbb{P}(\mathrm{ERM} = f) \leq (1-\epsilon)^m$$

Using union-bound, probability that *any* bad-hypothesis is chosen:

$$\mathbb{P}(\mathrm{ERM} \in \mathcal{B}) \leq \sum_{f \in \mathcal{B}} \mathbb{P}(\mathrm{ERM} = f) \leq |\mathcal{B}|(1-\epsilon)^m \leq |\mathcal{C}|(1-\epsilon)^m \leq |\mathcal{C}|e^{-m\epsilon}$$

Thus, if we take m number of training samples such that:

$$|\mathcal{C}|e^{-m\epsilon} \leq \delta, \quad \text{i.e.,} \quad m \geq \frac{1}{\epsilon} \ln \frac{|\mathcal{C}|}{\delta},$$

the chosen hypothesis is Good w.p. at least $1-\delta$. ■

Sample Complexity of Learning : Infinite Function Class

Clearly, the above proof does not extend to the important case when $|\mathcal{C}| = \infty$, (e.g., when $\mathcal{C}$ is set of all linear, polynomial functions etc.).

In a breakthrough paper in '95, Vapnik and Chervonenkis introduced the concept of VC-dimension associated with an arbitrary function class $\mathcal{C}$.

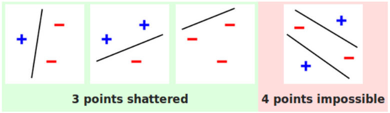
Definition (Shattering)

Suppose that there exists some set S of k points $S = \{\mathbf{x}_i \in \mathcal{X}, i = 1, 2, \ldots, k\}$ such that we can select a hypothesis $f \in \mathcal{C}$ which evaluates to *any* given binary label on this set of points. Then the set S is said to be *shattered* by $\mathcal{C}$.

VC dimension of the function class $\mathcal{C}$ is defined as the maximum cardinality of the set S which can be shattered by $\mathcal{C}$.

VC-dimension : Examples

- VC dimension of the class $\mathcal{C}$ of $2D$ halfspaces is 3.



- In general, VC dimension of n-D hyperplanes is $n+1$.
- Consider the class $\mathcal{C}$ of axis-aligned rectangles. *Claim:* VC dimension is ≥ 4.

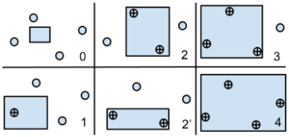


Figure 1: Proving that rectangle concept space shatters at least 4 points

Exercise: Show that VC dimension < 5

Sample Complexity

Theorem (Learning Theorem)

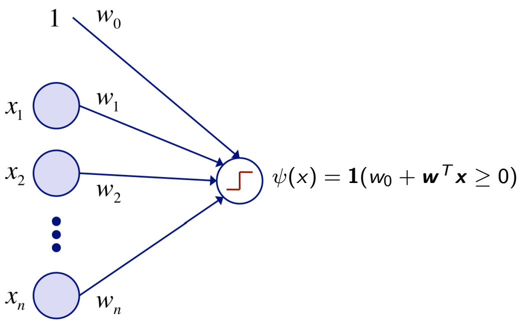
To learn a function class $\mathcal{C}$ of VC-dimension d with the usual parameters (ϵ, δ), it is necessary and sufficient to sample m data points, where $m = \Theta(\frac{1}{\epsilon}(d + \log(\frac{1}{\delta}))$

Compare with finite function class result that we proved : $\texttt{VC}_{\texttt{dim}} \sim \log(|\mathcal{C}|)$.

Outline

Linearly Separable Function Class: Perceptron Algorithm

In this case, we have $\mathcal{C} = \{1(\boldsymbol{w}^T x \geq 0), \boldsymbol{w} \in \mathbb{R}^n\}$.


1
w_0
x_1
w_1
x_2
w_2
x_n
w_n
$\psi(x) = \mathbf{1}(w_0 + \boldsymbol{w}^T \boldsymbol{x} \geq 0)$

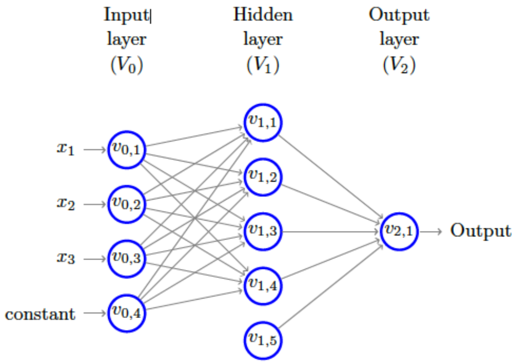
Training Algorithm:

$$\Delta \boldsymbol{w}_i^{k+1} \leftarrow \eta(\boldsymbol{y}_i^k - \boldsymbol{w}^k \boldsymbol{x}^k) x_i, \quad \boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k + \Delta \boldsymbol{w}^{k+1} \tag{3}$$

Outline

Neural Networks

- A neural network is a *layered* DAG $G(V, E)$ with one input layer, one output layer and at least one hidden layer.
- Each edge (i, j) has a tunable real valued weight w_{ij}.
- The vertices linearly combines the input and returns the *sign* (± 1) of the input.



A neural net of depth 2

Power of Neural Nets

Theorem (Universality of Neural Nets)

For any n, there exists a neural network of depth 2 such that it can implement any function $f : \{\pm 1\}^n \to \{\pm 1\}$.

Although the above theorem seems very impressive, the power of neural networks comes at a cost.

Theorem (Complexity of Neural Nets)

Let $s(n)$ denote the size (number of vertices) of a depth 2 neural net which can implement any boolean function of size n. Then $s(n)$ is exponential in n.

Thus, neural nets of limited size has limited power. In particular we have the following result:

Theorem (VC dimension)

The VC dimension of any neural network $G(V, E)$ with m edges is $O(m \log m)$.

The above theorem should not surprise as any neural network has m tunable weights, thus it is expected that "dimension" of the network should grow linearly in m.

Training a Neural Net

By *training* a neural network, we mean adjusting the weight parameters $\boldsymbol{w}$ of edges such that the training error is minimized (ERM).

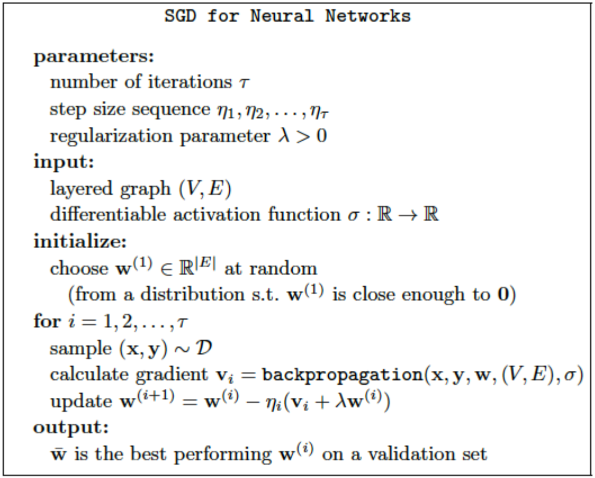
Theorem (Hardness of Training)

Consider a depth 2 neural network with n input nodes and one output node and at most 4 nodes in the hidden layer. Then it is NP-hard to train the network optimally.

Practical considerations:

- In practice, neural networks are trained (sub-optimally) by Stochastic Gradient Descent (SGD) algorithm:
  - GD : uses $\nabla_{\boldsymbol{w}}\left(\sum_{i=1}^{m}|y_i - f_{\boldsymbol{w}}(x_i)|^2\right)$, SGD : uses $\nabla_{\boldsymbol{w}}|y_i - f_{\boldsymbol{w}}(x_i)|^2$.
- Gradient of the overall cost function is calculated efficiently by an algorithm called `backpropagation`.

SGD for Training a Neural Network



```
SGD for Neural Networks

parameters:
  number of iterations $\tau$
  step size sequence $\eta_1, \eta_2, \ldots, \eta_\tau$
  regularization parameter $\lambda > 0$
input:
  layered graph $(V, E)$
  differentiable activation function $\sigma : \mathbb{R} \to \mathbb{R}$
initialize:
  choose $\mathbf{w}^{(1)} \in \mathbb{R}^{|E|}$ at random
    (from a distribution s.t. $\mathbf{w}^{(1)}$ is close enough to $\mathbf{0}$)
for $i = 1, 2, \ldots, \tau$
  sample $(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}$
  calculate gradient $\mathbf{v}_i = \texttt{backpropagation}(\mathbf{x}, \mathbf{y}, \mathbf{w}, (V, E), \sigma)$
  update $\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta_i(\mathbf{v}_i + \lambda \mathbf{w}^{(i)})$
output:
  $\bar{\mathbf{w}}$ is the best performing $\mathbf{w}^{(i)}$ on a validation set
```

Examples: http://bit.ly/2dCZYKw

Outline

Deep Neural Networks

Deep Neural networks are Neural networks with many hidden layers.

- Theoretical advantage for deep learning : Obvious as it increases the learning capacity (increased VC-dimension of the function class $\mathcal{C}$).
- History : Was tried in 90's with limited success, adding more layer yielded marginal performance gain.
  - Reason : *Was* hard to train with `backpropagation` : stuck in local optima.
- Idea 1: Keep many layers $(6-7)$ but make connections sparse (Convolutional Network, LeCun '98)
  - Less number of parameters and hence easier to train by `backpropagation`.
- Idea 2: Change the non-linearity to $\psi(x) = \max\{0, x\}$ (a.k.a. Linear Rectified Units (LRU), ImageNet, Hinton '12).
  - Was observed to be several times faster in training than convolutional network.

Why it works?

Nobody knows exactly. It is likely due to the following reasons:

- Local Minimas are as good as global minimas with proper regularization.
- SGD is able to find a “good solution” quickly [Choromanska, ’15]. Can be understood using concepts from Statistical Physics.