Deep Q Learning

Prof. Alfio Ferrara

Reinforcement Learning

Introduction

We are going to explore other (eventually non-linear) methods for Value Function Approximation (VFA) by Deep Learning. Our approach is based on the same definition of state value function and state-action value function that we have seen so far, that is:

Approximate value function

$$\hat{V}(s, \mathbf{w}) \rightarrow \mathbf{x(s)}^T\mathbf{w} \tag{1}$$

Approximate state-action value function

$$\hat{Q}(s, a, \mathbf{w}) \rightarrow \mathbf{x(s, a)}^T\mathbf{w} \tag{2}$$

Still, we are going to focus on differentiable approximation functions and we are going to use stochastic gradient descent to update the values of the two functions by updating the parameters $\mathbf{w}$. The goal of this formulation is to *generalize* the notion of state in order to deal with problems with a large state space, where the agent's behavior can be based on previous experience even if a specific state has never been encountered but only sufficiently similar states.

In this setting, state value and state-action value functions are no more represented by tables and the new information needed to update the parameters is given in form of a tuple (s, a, r, s') (we will use also the notation (s_t, a_t, r_t, s_{t+1})).

The objective function for learning is:

$$L(\mathbf{w}) = \mathbb{E}_\pi \left[\left(V^\pi(s) - \hat{V}(s, \mathbf{w}) \right)^2 \right] \tag{3}$$

and the update value of $\mathbf{w}$ by stochastic gradient descent is:

$$\Delta_\mathbf{w} = -\frac{1}{2}\alpha\nabla_\mathbf{w}L(\mathbf{w}) \tag{4}$$

Let's remember that this is not a supervised setting, because the ground-truth real value $V^\pi(s)$ is not known but must be estimated somehow.

We have two main methods for estimating $V^\pi(s)$:

- MC approach for policy evaluation: we use the sum of rewards G_t for a whole episode, so that $\Delta \mathbf{w} = \alpha \left(G_t - \mathbf{x}(s)^T \mathbf{w}\right) \mathbf{x}(\mathbf{s})$
- TD approach for policy evaluation: we use the (discounted) temporal difference of our estimations wuch that $\Delta_{\mathbf{w}} = \alpha \left(r_t + \gamma \mathbf{x}(s_{t+1})^T \mathbf{w} - \mathbf{x}(s_t)^T \mathbf{w}\right) \mathbf{x}(s_t)$

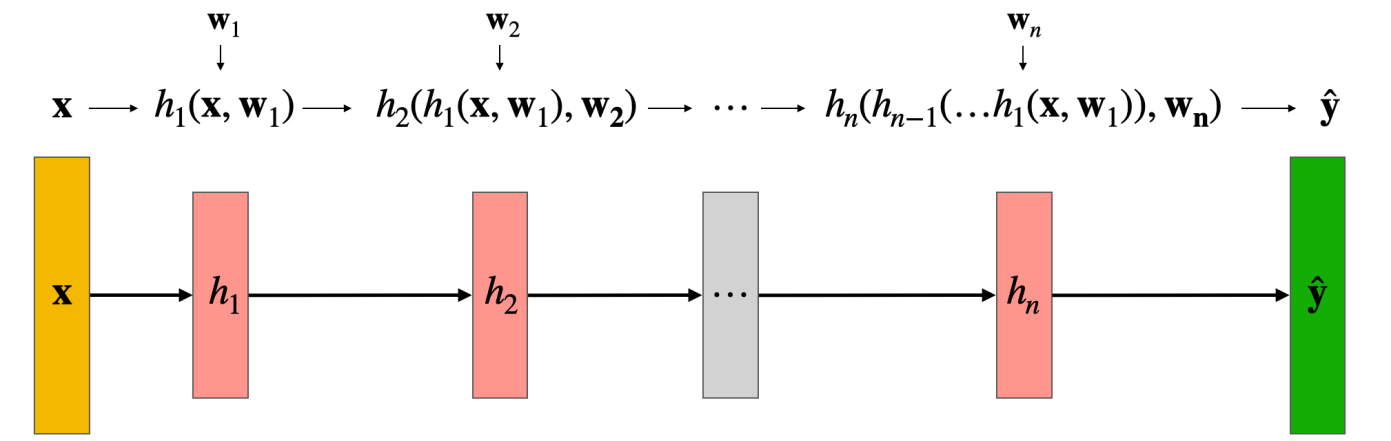
The same applies for $Q^\pi(s, a)$.

Linear approximation may be accurate but depends very much on the selection of features. To overcome this limitation, we are going to use other methods for approximating $V(s)$ and $Q(s, a)$, based on *deep neural networks*.

Summary on Deep Neural Networks

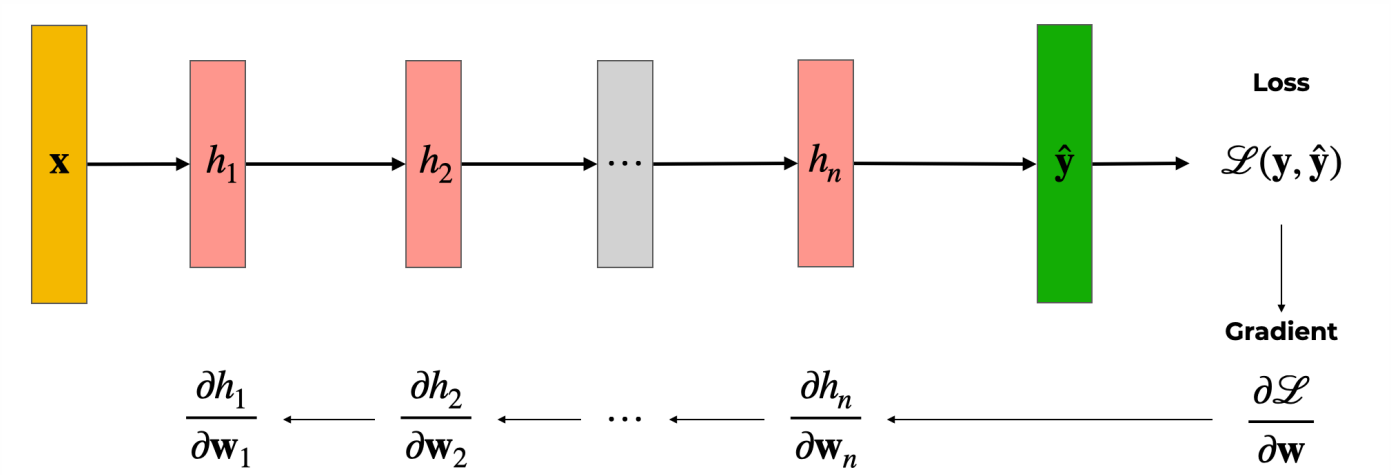
The main idea of DNNs is to produce and estimation by a composition of multiple functions.

Forward propagation



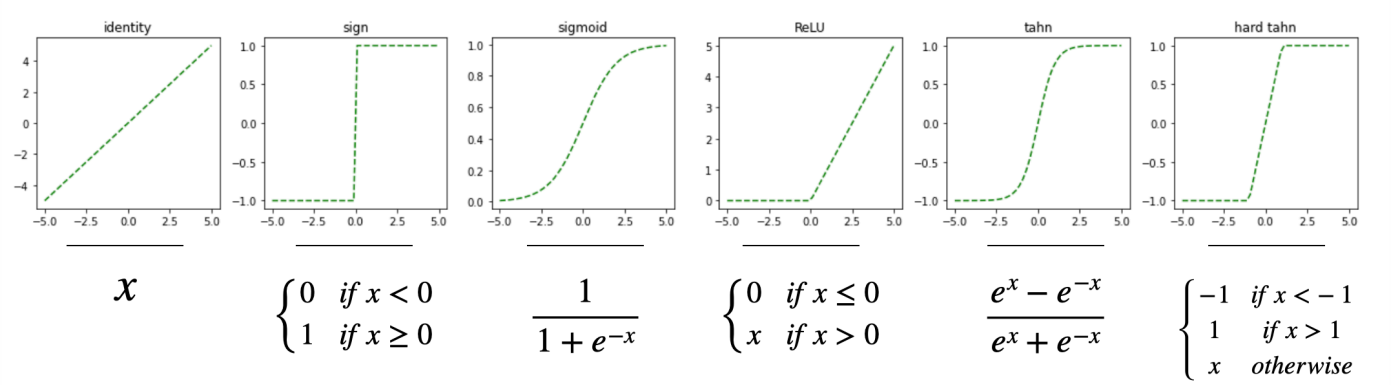
By decomposing a complex and wide spece of functions in a set of subsequent transformation, we ban propagate the gradient of the error back along the composition chain.

Backward propagation



The only constraint we have on the choice of h_i is that this must be a **differentiable** function in order to compute gradient descent. Actually, h_i can be either linear or not-linear. Tipicall, h_i is designed to be a function $f(h_{i-1})$ of the previous h_{i-1} layer. In this setting f if called **activation function** and h are said to be the network **layers**.

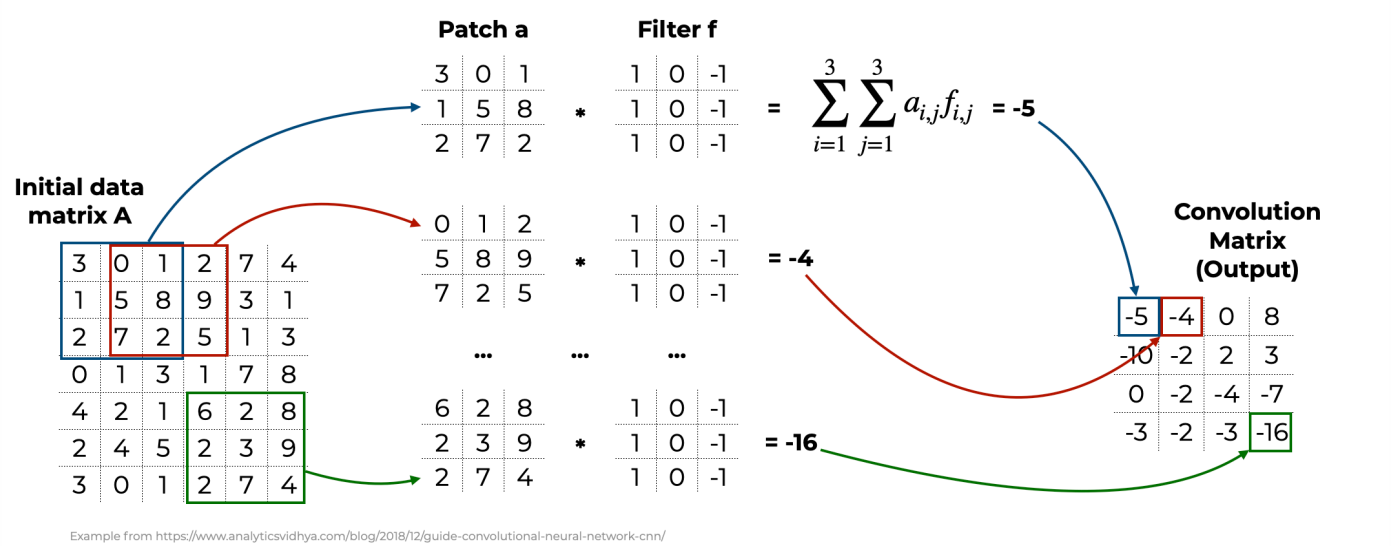
Popular activation functions



Convolutional Neural Networks (CNNs)

In order to use DNN to efficiently handle large space MDPs, we need a NN architecture that efficiently reduces the number of parameters required to compute all the functions that compose the solution.

The main idea of CNN is to use **filters** (or masks) on top of the input in order to reduce the input to smaller portions. But the point is to use the same weights for all the patches selected by the filter in the input, in order to produce a reduced version of the original input data.



Example from https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/

Through this mechanism, CNNs are particularly suited in their different layers for representing different features of the problem space.

In addition to one or more convolutinal layers, CNNs are typically equipped also with pooling layers, to compress information and a final fully connected set of layers.