CMPUT 365:
TD Update Stability

Rupam Mahmood

Mar 28, 2022

Convergence of linear TD update relies on its *stability*: which is defined as the convergence of the expected update

TD update: $$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\left(R_{t+1} + \gamma \mathbf{w}_t^\top \mathbf{x}_{t+1} - \mathbf{w}_t^\top \mathbf{x}_t\right)\mathbf{x}_t$$
$$= \mathbf{w}_t + \alpha\left(R_{t+1}\mathbf{x}_t - \mathbf{x}_t\left(\mathbf{x}_t - \gamma \mathbf{x}_{t+1}\right)^\top \mathbf{w}_t\right)$$

Expected TD update: $$\bar{\mathbf{w}}_{t+1} \doteq \bar{\mathbf{w}}_t + \alpha(\mathbf{b} - \mathbf{A}\bar{\mathbf{w}}_t)$$
$$= (\mathbf{I} - \alpha\mathbf{A})\bar{\mathbf{w}}_t + \alpha\mathbf{b}$$

where

$$\mathbf{b} \doteq \mathbb{E}[R_{t+1}\mathbf{x}_t] \in \mathbb{R}^d \quad \text{and} \quad \mathbf{A} \doteq \mathbb{E}\left[\mathbf{x}_t\left(\mathbf{x}_t - \gamma \mathbf{x}_{t+1}\right)^\top\right] \in \mathbb{R}^{d\times d}$$

When does the expected TD update converge?

Also Check Mahmood 2017 thesis, Chapter 9

Review of fixed-point iteration

$$\text{Expected TD update: } \bar{\mathbf{w}}_{t+1} \doteq \bar{\mathbf{w}}_t + \alpha(\mathbf{b} - \mathbf{A}\bar{\mathbf{w}}_t)$$
$$= (\mathbf{I} - \alpha\mathbf{A})\bar{\mathbf{w}}_t + \alpha\mathbf{b}$$

converge to the solution of TD fixed-point equation, i.e. $\mathbf{A}^{-1}\mathbf{b}$

the vector (take the 0 matrix)

The above is a fixed-point iteration (FPI), e.g., see CMPUT 340 (Heath 2018, Ch 5)

When an FPI converges, it converges to the solution of the fixed-point equation

For TD updates, it is known as the *TD fixed point*

$$\text{TD fixed-point equation: } \bar{\mathbf{w}}_{\text{TD}} = (\mathbf{I} - \alpha\mathbf{A})\bar{\mathbf{w}}_{\text{TD}} + \alpha\mathbf{b}$$
$$\bar{\mathbf{w}}_{\text{TD}} = \mathbf{A}^{-1}\mathbf{b}$$

Review of fixed-point iteration (cont'd)

An FPI converges if (Heath, pg 238)

$$\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x}).$$

is equation to solve.

The corresponding fixed-point iteration is simply

$$\boldsymbol{x}_{k+1} = \boldsymbol{g}(\boldsymbol{x}_k),$$

given some starting vector $\boldsymbol{x}_0$.

In one dimension, we saw that the convergence (and convergence rate) of fixed-point iteration is determined by $|g'(x^*)|$, where x^* is the solution. In higher dimensions the analogous condition is that the spectral radius

$$\rho(\boldsymbol{G}(\boldsymbol{x}^*)) < 1,$$

where $\boldsymbol{G}(\boldsymbol{x})$ denotes the *Jacobian matrix* of $\boldsymbol{g}$ evaluated at $\boldsymbol{x}$,

$$\{\boldsymbol{G}(\boldsymbol{x})\}_{ij} = \frac{\partial g_i(\boldsymbol{x})}{\partial x_j}.$$

If the foregoing condition is satisfied, then the fixed-point iteration converges if started close enough to the solution. (Note that testing this condition does not

Stability of linear TD update

For linear TD update, $\mathbf{G}(\mathbf{w}) = \mathbf{I} - \alpha\mathbf{A}$

So, the update is stable if $\varrho(\mathbf{I} - \alpha\mathbf{A}) < 1$

Equivalently if $\varrho(\mathbf{A}) > 0$, with a sufficiently small α

Which can be ensured if $\mathbf{A}$ is a *positive definite matrix*

Which happens to be true if the state distribution is on-policy

NN Wed

[Matrices and Vectors]

Notations:		hand-written	textbook
Scalar random variables	Big Letter	A	A
Scalar constants	Small	a	a
vectors	Small	$\underline{a}$ (underlined)	$\mathbf{a}$ (boldfaced)
Matrices	Big	$\underline{A}$ (underlined)	$\mathbf{A}$ (boldfaced)
ith element of a vector	Small	a_i	a_i
ith row & jth column of a matrix	Big	$A_{i,j}$	$A_{i,j}$

Vector inner product: $\underline{a}, \underline{b} \in \mathbb{R}^n$

$$\underline{a}^T \underline{b} = \sum_{i=1}^{n} a_i b_i$$

Matrix-vector product: $\underline{b} \in \mathbb{R}^n$, $\underline{A} \in \mathbb{R}^{m \times n}$

$$\underline{c} = \underline{A}\,\underline{b}. \quad c_i = \sum_{j=1}^{n} A_{ij} b_j$$

what is the dimensionality of $\underline{c}$? $\Rightarrow m \times 1$

$(m \times n) \times (n \times 1) = m \times 1$ vector.

Approximator is the linear function of the features.

Matrix-vector multiplication

If $\mathbf{x}$ is vector of size $p \times 1$, then $\mathbf{Ax}$ can be written in the following two ways:

$$\mathbf{Ax} = [\mathbf{A}]_{:,1}\ [\mathbf{A}]_{:,2}\ \cdots\ [\mathbf{A}]_{:,p} \begin{bmatrix} [\mathbf{x}]_1 \\ [\mathbf{x}]_2 \\ \cdots \\ [\mathbf{x}]_p \end{bmatrix} = \sum_{b=1}^{p} [\mathbf{A}]_{:,b} [\mathbf{x}]_b,$$

matrix is written in a row, each element is a column vector.

$$\mathbf{Ax} = \begin{bmatrix} [\mathbf{A}]_{1,:} \\ [\mathbf{A}]_{2,:} \\ \cdots \\ [\mathbf{A}]_{n,:} \end{bmatrix} \mathbf{x} = \begin{bmatrix} [\mathbf{A}]_{1,:}\mathbf{x} \\ [\mathbf{A}]_{2,:}\mathbf{x} \\ \cdots \\ [\mathbf{A}]_{n,:}\mathbf{x} \end{bmatrix}.$$

An element-wise vector product $\mathbf{x} \circ \mathbf{y}$ of two $p \times 1$ vectors $\mathbf{x}$, and $\mathbf{y}$ can be written as a matrix-vector multiplication in the following way:

$$\mathbf{x} \circ \mathbf{y} = \mathbf{y} \circ \mathbf{x} = \text{Diag}(\mathbf{x})\mathbf{y} = \text{Diag}(\mathbf{y})\mathbf{x}.$$

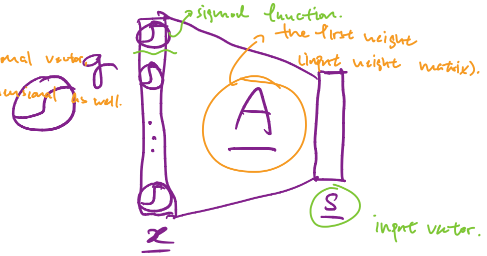
Consider worksheet Q2.

Then feature vector $\underline{x}$ is

$$\underline{x} \doteq g(\underline{\psi})$$

function (g)

If $\underline{\psi}$ is a n-dimensional vector then $\underline{x}$ will be n-dimensional as well.



$$\underline{\psi} = \underline{A}\,\underline{s}$$

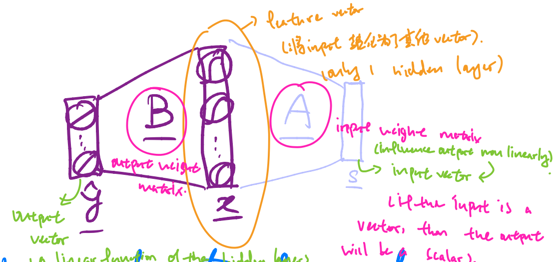
input vector.

the vector after linear transformation A.

And the output of the network $\hat{\underline{y}}$ is

$$\hat{y} = \underline{B}\,\underline{x}$$

the function of input $\underline{s}$.
(与$\hat{y}$相似).



Then the derivatives of the output elements are:

$$\frac{\partial \hat{y}_k}{\partial B_{k,j}} = \frac{\partial \sum_i B_{k,i}\, x_i}{\partial B_{k,j}}$$

each element of the output vector

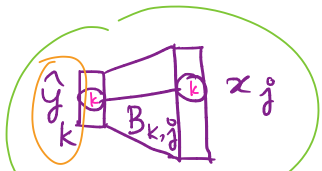
each element of the output weight matrix.

gradient of $\hat{y}$ with respect to B.

use to make an update to $B_{k,j}$

因为此处只对$B_{k,j}$求偏导，∴下列中除$B_{k,j}$所在以外其他都为0.

NN without hidden layer: is a linear function.



choosing element k of the output vector.
∴ B 那边也在 chose k. 不然无法对 j 产生 linear effect.

$$= \frac{\partial B_{k,1} x_1}{\partial B_{k,j}} + \frac{\partial B_{k,2} x_2}{\partial B_{k,j}} + \cdots + \frac{\partial B_{k,j} x_j}{\partial B_{k,j}} + \cdots$$

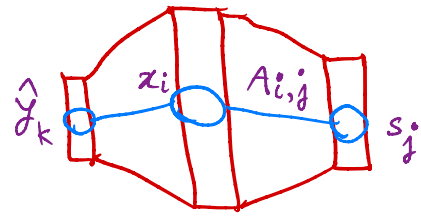
(=0) (=0)

$$= x_j.$$

$$\frac{\partial \hat{y}_k}{\partial A_{i,j}} = \frac{\partial \sum_\ell B_{k,\ell} x_\ell}{\partial x_i} \frac{\partial x_i}{\partial A_{i,j}}$$

use to make an update to $A_{i,j}$

$$= B_{k,i} \frac{\partial g(\psi_i)}{\partial A_{i,j}}$$



$$= B_{k,i} \frac{\partial g(\psi_i)}{\partial \psi_i} \frac{\partial \psi_i}{\partial A_{i,j}}$$ linear.

$$= B_{k,i}\, g'(\psi_i) \frac{\partial \sum_\ell A_{i,\ell} s_\ell}{\partial A_{i,j}}$$

$$= B_{k,i}\, g'(\psi_i)\, s_j .$$

9.5.3 Coarse Coding

Consider a task in which the natural representation of the state set is a continuous two-dimensional space. One kind of representation for this case is made up of features corresponding to *circles* in state space, as shown to the right. If the state is inside a circle, then the corresponding feature has the value 1 and is said to be *present*; otherwise the feature is 0 and is said to be *absent*. This kind of 1–0-valued feature is called a *binary feature*. Given a state, which binary features are present indicate within which circles the state lies, and thus coarsely code for its location. Representing a state with features that overlap in this way (although they need not be circles or binary) is known as *coarse coding*.

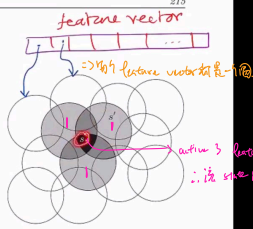


Figure 9.6: Coarse coding. Generalization from state s to state s' depends on the number of their features whose receptive fields (in this case, circles) overlap. These states have one feature in common, so there will be slight generalization between them.

Assuming linear gradient-descent function approximation, consider the effect of the size and density of the circles. Corresponding to each circle is a single weight (a component of $\mathbf{w}$) that is affected by learning. If we train at one state, a point in the space, then the weights of all circles intersecting that state will be affected. Thus, by (9.8), the approximate value function will be affected at all states within the union of the circles, with a greater effect the more circles a point has "in common" with the state, as shown in Figure 9.6. If the circles are small, then the generalization will be over a short distance, as in Figure 9.7 (left), whereas if they are large, it will be over a large distance, as in Figure 9.7 (middle). Moreover,

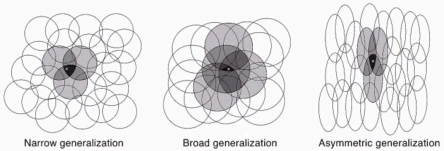


Figure 9.7: Generalization in linear function approximation methods is determined by the sizes and shapes of the features' receptive fields. All three of these cases have roughly the same