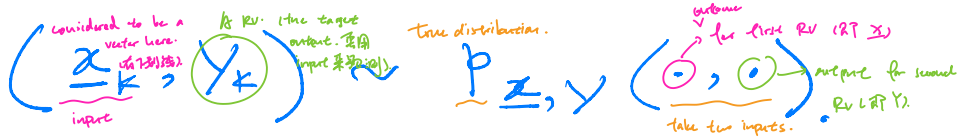
FA Prediction

Mon

[Supervised Learning]

* Samples arrive in the form of tuples of input and target output (the one we decide to predict base on the input) drawn i.i.d.:



Here $y \in \mathbb{R}$ is a scalar and $\underline{x}_k \in \mathbb{R}^n$ is an n-dimensional column vector denoted by an underline (boldface in the textbook).

n dimensional column vector.

Also, $p_{\underline{x},y}(\underline{a}, c) = P(\underline{x} = \underline{a}, y = c)$

($\underline{x}$ takes value $\underline{a}$, y takes value c)

the joint distribution of $\underline{x}, y$.

Dynamic Programming: all off-line. Although can do it online.

Reinforcement learning: agent interact with world, continue over time. (storing things in table is not flexible).

(Approximation)

* The learner wants to predict ^ output Y (the target) accurately based on the observed input X using a parameterized function f:

function f maps input to the output.

the learner observes x, and predicts ŷ

$$f(\underline{x}, \underline{w}) = \hat{y} \approx y$$

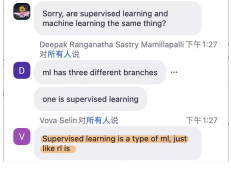
x: can be a table, a linear function, polynomial ...

w: a parameter vector.

ŷ: form by adding weight to input x. ∴ x is a vector. (linear)

⇒ use input to approximate output by this function. (try to predict the reward).

Here, the function f takes $\underline{x}$ as an input, is parameterized by a weight vector $\underline{w}$, and produces an output $\hat{y}$, which is a prediction or approximation of target output y.



value function (V_π, q_π) depend on the policy !! policy change, they change.

(objective)

Translate the goal:

* The learner's prediction accuracy depends on the value of $\underline{w}$.

The learner wants to minimize the prediction error $Y - f(\underline{x}, \underline{w})$ over different samples. This goal can be well formulated as the minimization of the Mean Squared Error (MSE) objective:

find a parameter $\in$ vector so that the whole wise will be minimized.

$$MSE(\underline{w}) = E_{p_{\underline{x},Y}}\left[(Y - f(\underline{x}, \underline{w}))^2\right]$$

- $MSE(\underline{w})$: what the equation have; a function of the weight vector, gonna give different value to different $\underline{w}$.
- $E_{p_{\underline{x},Y}}$: expectation is the true quantity.
- $(Y - f(\underline{x}, \underline{w}))^2$: composed R.V.
- want to find the $\underline{w}$ that minimize this square error.
- expectation of the square error.

It is the expected squared error between the target and predicted outputs.

* Note that

$$MSE(\underline{w}) = \sum p_{\underline{x},Y}(\underline{a}, c)\,(c - f(\underline{a}, \underline{w}))^2$$

takes this variable.

according to (which law?).

(Stochastic gradient descent) or SGD

SGD forms an unbiased estimate of the objective. For example, if the learner observes $(\underline{x}, y)$, then the squared prediction error is an unbiased estimate of MSE: $\boxed{(y - f(\underline{x}, \underline{w}))^2}$. ← the sample objective.

SGD then updates the weight vector $\underline{w}$ incrementally using the gradient of the squared error:

$$\underline{w}_{k+1} \doteq \underline{w}_k - \frac{1}{2}\alpha \nabla_{\underline{w}_k} \left(y_k - f(\underline{x}_k, \underline{w}_k) \right)^2$$

- $\underline{w}_{k+1}$: the new weight
- $\underline{w}_k$: the previous weight
- $\nabla_{\underline{w}_k}(y_k - f(\underline{x}_k, \underline{w}_k))^2$: the gradient of $(y_k - f(\underline{x}_k, \underline{w}_k))^2$
- $(y_k - f(\underline{x}_k, \underline{w}_k))^2$: unbiased estimate of the MSE.

Stochastic gradient ⇒ update. (do not take the min possible $\mathbb{E}_{\underline{x},y}[(y - f(\underline{x}, \underline{w}))^2]$)

$$= \underline{w}_k + \alpha \left(y_k - f(\underline{x}_k, \underline{w}_k) \right) \nabla_{\underline{w}_k} f(\underline{x}_k, \underline{w}_k).$$

- α: the stepsize.
- $\nabla_{\underline{w}_k} f(\underline{x}_k, \underline{w}_k)$: the gradient of the function itself.

For linear function approximation,

$\underline{w} \in \mathbb{R}^n$ is a column vector of the same size as $\underline{x}$:

$$f(\underline{x}, \underline{w}) = \underline{x}^T \underline{w} = \sum_{i=1}^{n} x_i w_i,$$

where x_i and w_i are ith elements of vectors $\underline{x}$ and $\underline{w}$, respectively.

Then the SGD update reduces to

$$\underline{w}_{k+1} \doteq \underline{w}_k + \alpha \left(Y_k - f(\underline{x}_k, \underline{w}_k) \right) \underline{x}_k,$$

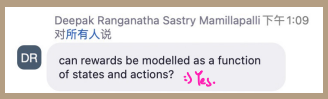
because

$$\nabla_{\underline{w}} f(\underline{x}, \underline{w}) = \nabla_{\underline{w}} \left(\underline{x}^T \underline{w} \right) = \underline{x}.$$

the input vector $\underline{x}$.

Prediction with Approximation

Wed



[Mean Squared Value Error]

What can be an objective for policy evaluation? Let's consider function approximation as described in section 9.1:

$\hat{v}(s, \underline{w}) \approx v_\pi(s)$. (wanna predict $v_\pi(s)$ by parameter $\underline{w}$) Then the policy evaluation / value prediction problem can be seen as a supervised learning problem with input, target output pairs as: $(S_t, v_\pi(S_t))$.

Then we can use the MSE objective we introduced before, which we call Mean Squared Value Error ($\overline{VE}$):

$$\overline{VE}(\underline{w}) = \mathbb{E}_{S_t \sim \mu}\left[\left(v_\pi(S_t) - \hat{v}(S_t, \underline{w})\right)^2\right],$$

where μ is some state distribution. i.i.d

[Mean Squared Return Error]

$\overline{VE}$ can be re-written as:

$$\overline{VE}(\underline{w}) = \sum_s \mu(s) \left(v_\pi(s) - \hat{v}(s, \underline{w}) \right)^2.$$

Interestingly, minimizing $\overline{VE}$ is equivalent to minimizing the following objective we call Mean Squared Return Error (MSRE):

$$MSRE(\underline{w}) = E_{S_t \sim \mu, A \sim \pi} \left[\left(G_t - \hat{v}(S_t, \underline{w}) \right)^2 \right].$$

agent trys to use S_t as input and predict expected return G_t as output. the return.

$S_t \sim \mu$ 表明不满足iid. S_t is not iid.

It is a different objective with the same solution / minimum.

[Deriving Monte Carlo Method as SGD]

Then an unbiased estimate can be easily obtained as $\left(G_t - \hat{v}(S_t, \underline{w})\right)^2$ if we can sample states from μ.

If μ is the on-policy distribution, then we can just use the sequential experience of S_t and make SGD updates:

In tabular case: $$\underline{w} \leftarrow \underline{w} + \alpha \left[G_t - \hat{v}(S_t, \underline{w}) \right] \nabla_{\underline{w}} \hat{v}(S_t, \underline{w})$$

the Monte Carlo error.

gradient of the approximation.

which is the gradient MC pseudocode from Section 9.3.

∵ for tabular case, we don't have gradient!!