Notes on Reinforcement Learning

ERMO WEI

Contents

Activation Function Activation Function are important for neural network to have non-linearity. The commonly used activation function are summarize below. Most of them are typically combined with an affine transformation $\mathbf{a} = \mathbf{b} + \mathbf{W}\mathbf{x}$ and applied element-wise.

$$\mathbf{h} = f(\mathbf{a}) \rightarrow h_i = f(a_i) = f(b_i + \mathbf{W}_{\mathbf{i},:} * \mathbf{x})$$

- Sigmoid

$$f(a) = \frac{1}{1 + e^{-a}}$$

- Hyperbolic tangent

$$f(a) = \tanh(a)$$

- Softmax
- Rectifier or rectified linear unit (ReLU)

$$f(a) = \max(0, a)$$

- Maxout
- Softplus
  A smooth version of the rectifier, basically the same shape. But this function has differentiability and non-zero derivative everywhere.

$$f(a) = \log(1 + e^a)$$

Actor-Critic Method Need more time to work on it ...

Approximate Inference A central task in the application of Bayesian probabilistic models is the evaluation of the posterior distribution $p(\mathbf{Z}|\mathbf{X})$ of the latent variables $\mathbf{Z}$ given the observed data variables $\mathbf{X}$, and the evaluation of expectations computed with respect to this distribution. However, for many models of practical interest, it will be infeasible to evaluate the posterior distribution or indeed to compute expectations with respect to this distribution. There are several reasons for this:

- The dimensionality of the latent space is too high to work with directly
- The posterior distribution has a highly complex form for which expectations are not analytically tractable

For example, in the case of continuous variables, the required integrations may not have closed-form analytical solutions, while the dimensionality of the space and the complexity of the integrand may prohibit numerical integration. For discrete variables, the marginalizations involve summing over all possible configurations of the hidden variables, and though this is always possible in principle, we often find in practice that there may be exponentially many hidden states so that exact calculation is prohibitively expensive.

In such situations, we need to resort to approximation schemes, and these fall broadly into two classes, according to whether they rely on deterministic or stochastic approximations. For the deterministic approximation schemes, variational inference methods which use global criteria are widely used, some of which scale well to large applications. These methods are basing on analytical approximations to the posterior distribution, for example by assuming that it factorizes in a particular way or that it has a specific parametric form such as a Gaussian. Due to this reasons, they can never generate exact results.

On the other hand, stochastic techniques, which based on numerical sampling, such as Markov chain Monte Carlo, generally have the property that given infinite computational resource, they can generate

exact results, and the approximation arises from the use of a finite amount of processor time. In practice, sampling methods can be computationally demanding, often limiting their use to small-scale problems. Also, it can be difficult to know whether a sampling scheme is generating independent samples from the required distribution.

Back Propagation Let's start with linear neurons (also called linear filters). Here we first make two assumptions:

- The neuron has a real- valued output which is a weighted sum of its inputs
- The aim of learning is to minimize the square error summed over all training cases.

The model of the linear neuron is like this :

$$y = \sum w_i x_i = \mathbf{w}^T \mathbf{x}$$

Since we are dealing with a simple linear neuron, we first derive the special case of back propagation algorithm, called *delta rule*, which is a gradient descent learning rule for updating the weights of single layer neural network.

1. We first define the error of our training cases:

$$E = \frac{1}{2} \sum_{n \in \text{training set}} (t^n - y^n)^2$$

2. Now we take the derivative of the error with respect to the weights

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{1}{2} \sum_n \frac{dE^n}{dy^n} \frac{\partial y^n}{\partial w_i} \\ &= -\sum_n x_i^n (t^n - y^n) \end{aligned}$$

3. The **batch** delta rule changes the weights in proportion to their error derivatives summed over all training cases

$$\Delta w_i = -\epsilon \frac{\partial E}{\partial w_i} = \sum_n \epsilon x_i^n (t^n - y^n)$$

One thing to be notice here, we can get as close as we desire to the best answer by making the learning rate small enough

Ok, since we have the simplest neuron, let's making things little interesting by adding the logistic function. And now, our model become this:

$$\begin{aligned} z &= b + \sum_i w_i x_i \\ y &= \frac{1}{1 + e^{-z}} \end{aligned}$$

their corresponding derivatives are

$$\begin{aligned} \frac{\partial z}{\partial w_i} = x_i \,, \frac{\partial z}{\partial x_i} = w_i \\ \frac{dy}{dz} = y(1 - y) \end{aligned}$$

Thus, take the derivative of y with respect to w_i is

$$\frac{\partial y}{\partial w_i} = \frac{dy}{dz}\frac{\partial z}{\partial w_i} = x_i y(1-y)$$

and

$$\frac{\partial E}{\partial w_i} = \sum \frac{\partial E}{\partial y^n}\frac{\partial y^n}{\partial w_i} = -\sum_n x_i^n y^n (1-y^n)(t^n - y^n)$$

In this equation, the first and last terms come from the delta rule, and the term in the middle is the slope of logistic

Need more time to work on it ...

Basic Sampling In the application of Bayesian probabilistic models, we often need to evaluate the posterior distribution $p(\mathbf{Z}|\mathbf{X})$ of the latent variables $\mathbf{Z}$ given the observed data variables $\mathbf{X}$ (see Approximate Inference).

Although for some applications the posterior distribution over unobserved variables will be of direct interest in itself, for most situations the posterior distribution is required primarily for the purpose of evaluating expectations, for example in order to make predictions. The fundamental problem that we therefore wish to address involves finding the expectation of some function $f(\mathbf{z})$ with respect to a probability distribution $p(\mathbf{z})$. Here, the components of $\mathbf{z}$ might comprise discrete or continuous variables or some combination of the two. Thus in the case of continuous variables, we wish to evaluate the expectation

$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})\mathrm{d}\mathbf{z} \tag{1}$$

where the integral is replaced by summation in the case of discrete variables. We shall suppose that such expectations are too complex to be evaluated exactly using analytical techniques.

The general idea behind sampling methods is to obtain a set of samples $\mathbf{z}^{(l)}$ (where $l = 1, \ldots, L$) drawn independently from the distribution $p(\mathbf{z})$. This allows the expectation (1) to be approximated by a finite sum

$$\hat{f} = \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{(l)})$$

As long as the samples $\mathbf{z}^{(l)}$ are drawn from the distribution $p(\mathbf{z})$, then $\mathbb{E}[\hat{f}] = \mathbb{E}[f]$ and so the estimator $\hat{f}$ has the correct mean. The variance of the estimator is given by

$$\mathrm{Var}[\hat{f}] = \frac{1}{L}\mathbb{E}[(f - \mathbb{E}[f])^2]$$

is the variance of the function $f(\mathbf{z})$ under the distribution $p(\mathbf{z})$. It is worth emphasizing that the accuracy of the estimator therefore does not depend on the dimensionality of $\mathbf{z}$, and that, in principle, high accuracy may be achievable with a relatively small number of samples $\mathbf{z}^{(l)}$. In practice, ten or twenty independent samples may suffice to estimate an expectation to sufficient accuracy.

The problem, however, is that the samples $\{\mathbf{z}^{(l)}\}$ might not be independent, and so the effective sample size might be much smaller than the apparent sample size. Also, we note that if $f(\mathbf{z})$ is small in regions where $p(\mathbf{z})$ is large, and vice versa, then the expectation may be dominated by regions of small probability, implying that relatively large sample sizes will be required to achieve sufficient accuracy.

We first consider how to sample from some common distributions. We assume that we have a pseudo-random generator which can generate numbers distributed uniformly over $(0, 1)$.

Need more time to work on it ...

Bellman Equation Due to the Markov property of MDP, we can define the value function of a policy recursively to be

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}\{R_t | s_t = s\} \\ &= \mathbb{E}_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} \\ &= \mathbb{E}_{\pi}\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\} \end{aligned}$$

If policy is stochastic, then we have

$$\begin{aligned} V^{\pi}(s) &= \sum_a \pi(s,a) \sum_{s'} \Pr(s'|s,a) \big[R(s,a) + \gamma \, \mathbb{E}_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s'\}\big] \\ &= \sum_a \pi(s,a) \sum_{s'} \Pr(s'|s,a) \big[R(s,a) + \gamma V^{\pi}(s')\big] \\ &= \sum_a \pi(s,a) \{R(s,a) + \sum_{s'} \Pr(s'|s,a) \gamma V^{\pi}(s')\} \end{aligned}$$

If policy is deterministic, then the Value function of policy π at state s is

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} \Pr(s'|s, \pi(s)) V^{\pi}(s')$$

Thus, the Bellman optimality equation can be defined as

$$V^*(s) = \max_a \{R(s,a) + \gamma \sum_{s'} \Pr(s'|s,a) V^*(s')\}$$

Detail see (Barto, 1998)

Bellman Operator Consider a MDP, Bellman Operator for policy π can be defined as a map from a value function to another value function

$$T^{\pi} : \mathbb{R}^N \to \mathbb{R}^N$$

where N is the cardinality of state space, i.e. $N = |S|$.

Need more time to work on it ...

Boltzmann Selection Need more time to work on it ...

Contraction The operator F (function f) is a α-contraction ($0 \leq \alpha < 1$, called Lipschitz constant for F) with respect to some norm $\|\cdot\|$ if

$$\forall X, \overline{X} : \|FX - F\overline{X}\| \leq \alpha \|X - \overline{X}\| \quad \text{or} \quad \|f(X) - f(\overline{X})\| \leq \alpha \|X - \overline{X}\|$$

- Theorem 1. The sequence $X, FX, F^2X, \ldots$ converges for every X. e.g. $X, f(X), f(f(X)), \ldots$ converges for every X

  Proof:

  Useful fact : Cauchy sequences : If for $x_0, x_1, x_2, \ldots$, we have that

  $$\forall \epsilon, \exists K : \|x_M - x_N\| < \epsilon \quad \text{for} \quad M, N > K$$

then we call $x_0, x_1, x_2, \ldots,$ a Cauchy sequence.

If $x_0, x_1, x_2, \ldots,$ is a Cauchy sequence, and $x_i \in \mathbb{R}^n$, then there exists $x^* \in \mathbb{R}^n$ such that $\lim_{i\to\infty} x_i = x^*$.

Proof: Assume $N > M$.

$$\begin{aligned}
\|F^M X - F^N X\| &= \|\sum_{i=M}^{N-1}(F^i X - F^{i+1}X)\| \\
&\leq \sum_{i=M}^{N-1}\|F^i X - F^{i+1}X\| && \text{by triangle inequality} \\
&\leq \sum_{i=M}^{N-1}\alpha^i\|X - FX\| && \text{use the condition in Theorem} \\
&= \|X - FX\|\sum_{i=M}^{N-1}\alpha^i \\
&= \|X - FX\|\frac{\alpha^M}{1-\alpha}
\end{aligned}$$

As $\|X - FX\|\frac{\alpha^M}{1-\alpha}$ goes to zero for M going to infinity, we have that for any $\epsilon > 0$ for $\|F^M X - F^N X\| \leq \epsilon$ to hold for all $M, N > K$, it suffices to pick K large enough. Hence, $X, FX, \ldots$ is Cauchy sequence and converges.

- Theorem 2 (Banach fixed-point theorem). F has a unique fixed point X^* which satisfies $FX^* = X^*$ and all sequences X, FX, F^2X, $\ldots$ converge this unique fixed point X.

  Proof:

  Suppose F has two fixed points. Let's say

  $$\begin{aligned} FX_1 &= X_1 \\ FX_2 &= X_2 \end{aligned}$$

  this implies, $\|FX_1 - FX_2\| = \|X_1 - X_2\|$. At the same time we have from the contractive property of F

  $$\|FX_1 - FX_2\| \leq \alpha\|X_1 - X_2\|$$

  Combining both gives us

  $$\|X_1 - X_2\| \leq \alpha\|X_1 - X_2\|$$

  Hence, $X_1 = X_2$.

Detail see (Conrad, 2014)

Compatible Feature If we are using Softmax Policy, which is

$$\pi(s,a) = \frac{e^{\phi(s,a)^\top\theta}}{\sum_b e^{\phi(s,b)^\top\theta}}$$

if we have $p = e^{\phi(s,a)^\top \theta}$ and $q = \sum_b e^{\phi(s,b)^\top \theta}$, then $\pi(s,a) = p/q$

$$\begin{aligned}
\nabla_\theta \ln \pi(s,a) &= \frac{1}{\pi(s,a)} \frac{\partial \pi(s,a)}{\partial \theta} \\
&= \frac{q}{p} \frac{\frac{\partial p}{\partial \theta} q - \frac{\partial q}{\partial \theta} p}{q^2} \\
&= \frac{\frac{\partial p}{\partial \theta}}{p} - \frac{\frac{\partial q}{\partial \theta}}{q} \\
&= \frac{e^{\phi(s,a)^\top \theta} \phi(s,a)}{e^{\phi(s,a)^\top \theta}} - \frac{\sum_b (e^{\phi(s,b)^\top \theta} \phi(s,b))}{\sum_b e^{\phi(s,b)^\top \theta}} \\
&= \phi(s,a) - \sum_b \frac{e^{\phi(s,b)^\top \theta} \phi(s,b)}{\sum_b e^{\phi(s,b)^\top \theta}} \\
&= \phi(s,a) - \sum_b \Big[\frac{e^{\phi(s,b)^\top \theta}}{\sum_b e^{\phi(s,b)^\top \theta}} \phi(s,b) \Big] \\
&= \phi(s,a) - \sum_b \pi(s,b) \phi(s,b)
\end{aligned}$$

if we are using d-dimensional Gaussian Policy, suppose we fix the covariant matrix to $\mathbf{\Sigma} = c\mathbf{I}$, then our policy is

$$\pi(s,a) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\Big\{ -\frac{1}{2} (a - \phi(s)^\top \theta)^\top \Sigma^{-1} (a - \phi(s)^\top \theta) \Big\}$$

the score function is

$$\begin{aligned}
\nabla_\theta \ln \pi(s,a) &= \frac{1}{\pi(s,a)} \frac{\partial \pi(s,a)}{\partial \theta} \\
&= \frac{\sqrt{(2\pi)^d |\Sigma|}}{\exp\{\dots\}} \frac{\exp\{\dots\}}{\sqrt{(2\pi)^d |\Sigma|}} (-\frac{1}{2}) (\frac{\partial (a - \phi(s)^\top \theta)^\top \Sigma^{-1} (a - \phi(s)^\top \theta)}{\partial \theta}) \\
&= -\Sigma^{-1} \big[a - \phi(s)^\top \theta \big] \phi(s) =
\end{aligned}$$

Conditional Entropy Suppose we have a joint distribution $p(x,y)$ from which we draw pairs of values of x and y. If a value of x is already known, then the additional information needed to specify the corresponding value of y is given by (use the $h(\cdot)$ function in entropy)

$$\begin{aligned}
h(x,y) - h(x) &= -\ln p(x,y) - (-\ln p(x)) \\
&= -\ln p(y|x) p(x) + \ln p(x) \\
&= -\ln p(y|x) - \ln p(x) + \ln p(x) \\
&= -\ln p(y|x)
\end{aligned}$$

Thus the average additional information needed to specify y can be written as

$$H[y|x] = -\iint p(y,x) \ln p(y|x) \mathrm{d}y \mathrm{d}x$$

which is called the *conditional entropy* of y given x. It is easily seen, using the product rule, that the conditional entropy satisfies the relation

$$H[x,y] = H[y|x] + H[x]$$

where $H[x,y]$ is the differential entropy of $p(x,y)$ and $H[x]$ is the differential entropy of the marginal distribution $p(x)$. Thus the information needed to describe x and y is given by the sum of the information needed to describe x alone plus the additional information required to specify y given x.

Differential Entropy Differential Entropy generalize the concept of entropy to continuous cases. We first divide x into bins of width Δ. Then, assuming $p(x)$ is continuous, the *mean value theorem* tells us that, for each such bin, there must exist a value x_i such that

$$\int_{i\Delta}^{(i+1)\Delta} p(x)\mathrm{d}x = p(x_i)\Delta$$

We can now quantize the continuous variable x by assigning any value x to the value x_i whenever x falls in the ith bin. The probability of observing the value x_i is then $p(x_i)\Delta$. This gives a discrete distribution for which the entropy takes the form

$$\begin{aligned} H_\Delta &= -\sum_i p(x_i)\Delta \ln(p(x_i)\Delta) \\ &= -\sum_i p(x_i)\Delta \ln p(x_i) - \ln \Delta \end{aligned}$$

If we consider the limit $\Delta \to 0$. The first term will approach the integral of $p(x)\ln p(x)$ in this limit so that

$$\lim_{\Delta\to 0}\left\{\sum_i p(x_i)\Delta \ln p(x_i)\right\} = -\int p(x)\ln p(x)\mathrm{d}x$$

The term on the very right-hand side is called the *differential entropy*. We can see that the discrete and continuous forms of the entropy differ by a quantity $\ln\Delta$, which diverges in the limit $\Delta \to 0$. This reflects the fact that to specify a continuous variable very precisely requires a large number of bits.

Let us now consider the maximum entropy configuration for a continuous variable. In order for this maximum to be well defined, it will be necessary to constrain the first and second moments of $p(x)$ as well as preserving the normalization constraint. We therefore maximize the differential entropy with the three constraints

$$\begin{aligned} \int_{-\infty}^{\infty} p(x)\mathrm{d}x &= 1 \\ \int_{-\infty}^{\infty} xp(x)\mathrm{d}x &= \mu \\ \int_{-\infty}^{\infty} (x-\mu)^2 p(x)\mathrm{d}x &= \sigma^2 \end{aligned}$$

The constrained maximization can be performed using Lagrange multipliers so that we maximize the following functional with respect to $p(x)$

$$-\int_{-\infty}^{\infty} p(x)\ln p(x)\mathrm{d}x + \lambda_1\left(\int_{-\infty}^{\infty} p(x)\mathrm{d}x - 1\right) + \lambda_2\left(\int_{-\infty}^{\infty} xp(x)\mathrm{d}x - \mu\right) + \lambda_3\left(\int_{-\infty}^{\infty}(x-\mu)^2 p(x)\mathrm{d}x - \sigma^2\right)$$

We set the derivative of this functional to zero giving

$$p(x) = \exp\{-1 + \lambda_1 + \lambda_2 x + \lambda_3(x-\mu)^2\}$$

The Lagrange multipliers can be found by back substitution of this result into the three constraint equations, leading finally to the result

$$p(x) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}}\exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$

and so the distribution that maximizes the differential entropy is the Gaussian. If we evaluate the differential entropy of the Gaussian, we obtain

$$H[x] = \frac{1}{2}\{1 + \ln(2\pi\sigma^2)\}$$

Thus we see again that the entropy increases as the distribution becomes broader, i.e., as σ^2 increases. This result also shows that the differential entropy, unlike the discrete entropy, can be negative, because $H(x) < 0$ for $\sigma^2 < \frac{1}{2\pi e}$.

Entropy Consider a discrete random variable x, if we want to use a quantity $h(x)$ to measure how much information is received when we observe a specific value for this variable. The amount of information can be viewed as the "degree of surprise" or " degree of information" on learning the value of x.

Apparently, if we observed that a highly improbable event has just occurred, we will have received more information than if we observed some very likely event has just occurred, and if we knew that the event was certain to happen we would receive no information. Our measure of information content will therefore depend on the probability distribution $p(x)$, and the quantity $h(x)$ we are looking for is a monotonic function of the probability $p(x)$ that low $p(x)$ indicates high $h(x)$.

The form of $h(\cdot)$ can be found by noting that if we have two events x and y that are unrelated, then the information gain from observing both of them should be the sum of the information gained from each of them separately, so that $h(x, y) = h(x) + h(y)$. Two unrelated events will be statistically independent and so $p(x, y) = p(x)p(y)$. Therefore, it is easily shown that $h(x)$ must be given by the logarithm of $p(x)$ and so we have

$$h(x) = -\log_2 p(x)$$

The negative sign ensures that information is positive or zero and also the low probability events x correspond to high information content. By adopting the convention in information theory, we use the base of 2, and in this case, the units of $h(x)$ are bits ('binary digits'). If we use the natural logarithms in defining entropy, the units is 'nats' instead of bits, which differ simply by a factor of $\ln 2$.

Now suppose that a sender wishes to transmit the value of a random variable to a receiver. The average amount of information that they transmit in the process is obtained by taking the expectation of $h(x)$ with respect to the distribution p(x) and is given by

$$H[x] = -\sum_x p(x) \log_2 p(x)$$

or

$$H[x] = -\sum_x p(x) \ln p(x)$$

This important quantity is called the *entropy* of the random variable x. Note that $\lim_{x\to 0} p \ln p = 0$ and so we shall take $p(x) \ln p(x) = 0$ whenever we encounter a value for x such that $p(x) = 0$. Entropy can be thought as the average amount of information needed to specify the state of a random variable and a sharply peaked distribution $p(x_i)$ will have a relatively low entropy.

ϵ-greedy ϵ-greedy is a common used exploration strategy for Model-free reinforcement learning. This strategy can be seen as a combination of greedy strategy and random strategy. Every time when agent choose an action to perform, it choose greedily with probability 1-ϵ (exploitation), and randomly with probability ϵ (exploration). However, based on the changes of ϵ value, this method have two versions. First, the ϵ value can decrease as the learning process goes on, this is the *decay exploration* method. Another one which are more commonly used is that we fix the value of ϵ.

The difference between those methods is that, the first one can be GLIE if the ϵ-value goes 0 eventually but the second one cannot. Suppose we have a counter of how many times a state have been visited, $n_t(s)$ and a constant c. As long as the ϵ value for the a state $\epsilon_t(s) = \frac{c}{n_t(s)}$ where $0 < c < 1$, this method can be consider GLIE. However, in practice, we usually use fixed value for ϵ and after the convergence of the estimation value of action, we switch to greedy selection.

Ergodic MDP An MDP is said to be ergodic if for each policy π the Markov chain induced by π is ergodic. We are giving the definition of Ergodicity below. But first, we will give some auxiliary definitions. Several definition of Markov chain:

- Reducibility
  A Markov chain is said to be irreducible if it is possible to get to any state from any state.
- Aperiodicity
  A state i has period k if any return to state i must occur in multiples of k time steps. For example, suppose it is possible to return to a state in $\{6, 8, 10, 12, \ldots\}$ time steps, then k would be 2, even though 2 does not appear in this list. If $k = 1$, then the state is said to be aperiodic: returns to state i can occur at irregular times.
- Recurrence
  A state i is said to be transient if, given that we start in state i, there is a non-zero probability that we will never return to i. State i is recurrent (or persistent) if it is not transient. Recurrent states are guaranteed to have a finite hitting time.

Here we have the definition of Ergodicity.

A state i is said to be ergodic if it is aperiodic and positive recurrent. In other words, a state i is ergodic if it is recurrent, has a period of 1 and it has finite mean recurrence time. If all states in an irreducible Markov chain are ergodic, then the chain is said to be ergodic.

It can be shown that a finite state irreducible Markov chain is ergodic if it has an aperiodic state. **A model has the ergodic property if there's a finite number N such that any state can be reached from any other state in exactly N steps.** For example, we will have $N = 1$ if we have a fully connected transition matrix where all transitions have a non-zero probability. **Additionally, an stationary distribution d^π of states exists and is independent start state s_0.**

Detail see (Ortner, 2007).

Expectation Maximization The *expectation maximization* algorithm, or EM algorithm, is a general technique for finding maximum likelihood solutions for probabilistic models having latent variables. Consider a probabilistic model in which we collectively denote all of the observed variables by $\mathbf{X}$ and all of the hidden variables by $\mathbf{Z}$. The joint distribution $p(\mathbf{X}, \mathbf{Z}|\theta)$ is governed by a set of parameters denoted θ. Our goal is to maximize the likelihood function that is given by

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta)$$

The discussion is identical if $\mathbf{Z}$ comprises continuous variables or a combination of discrete and continuous variables, with summation replaced by integration as appropriate.

The EM algorithm can be used in situation where direct optimization of $p(\mathbf{X}|\theta)$ is difficult, but that optimization of the joint likelihood function $p(\mathbf{X}, \mathbf{Z}|\theta)$ is significantly easier. Now, we introduce a distribution $q(\mathbf{Z})$ defined over the latent variables, and we observe that, for any choice of $q(\mathbf{Z})$, we can have the following derivation

$$\begin{aligned} \ln p(\mathbf{X}, \mathbf{Z}|\theta) &= \ln p(\mathbf{Z}|\mathbf{X}, \theta) p(\mathbf{X}, \theta) \\ &= \ln p(\mathbf{Z}|\mathbf{X}, \theta) + \ln p(\mathbf{X}, \theta) \\ \ln p(\mathbf{X}, \theta) &= \ln p(\mathbf{X}, \mathbf{Z}|\theta) - \ln p(\mathbf{Z}|\mathbf{X}, \theta) \end{aligned}$$

Then we times both side $q(\mathbf{Z})$, and sum over all the $\mathbf{Z}$.

$$\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X}, \theta) = \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X}, \mathbf{Z}|\theta) - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{Z}|\mathbf{X}, \theta)$$

Since the term $\ln p(\mathbf{X}, \theta)$ does not depend on $\mathbf{Z}$, we take it out of the summation, and we notice that $\sum_{\mathbf{Z}} p(\mathbf{Z}) = 1$. So,

$$\begin{aligned}
\ln p(\mathbf{X}, \theta) \sum_{\mathbf{Z}} q(\mathbf{Z}) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X}, \mathbf{Z}|\theta) - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{Z}|\mathbf{X}, \theta) \\
\ln p(\mathbf{X}, \theta) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X}, \mathbf{Z}|\theta) - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{Z}|\mathbf{X}, \theta) \\
\ln p(\mathbf{X}, \theta) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X}, \mathbf{Z}|\theta) - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{Z}|\mathbf{X}, \theta) - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln q(\mathbf{Z}) + \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln q(\mathbf{Z}) \\
\ln p(\mathbf{X}, \theta) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{X}, \mathbf{Z}|\theta) - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln q(\mathbf{Z}) - \left(\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln p(\mathbf{Z}|\mathbf{X}, \theta) - \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln q(\mathbf{Z}) \right) \\
\ln p(\mathbf{X}, \theta) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z}|\theta)}{q(\mathbf{Z})} \right\} - \left(\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X}, \theta)}{q(\mathbf{Z})} \right\} \right)
\end{aligned}$$

If we define

$$\begin{aligned}
\mathcal{L}(q, \theta) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z}|\theta)}{q(\mathbf{Z})} \right\} \\
\mathrm{KL}(q||p) &= -\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X}, \theta)}{q(\mathbf{Z})} \right\}
\end{aligned}$$

Then, we can have the following equation

$$\ln p(\mathbf{X}|\theta) = \mathcal{L}(q, \theta) + \mathrm{KL}(q||p) \tag{2}$$

From 2, we see that $\mathrm{KL}(q||p)$ is the Kullback-Leibler divergence between $q(\mathbf{Z})$ and the posterior distribution $p(\mathbf{Z}|\mathbf{X}, \theta)$. Since Kullback-Leibler divergence satisfies $\mathrm{KL}(q||p) \geq 0$, with equality if, and only if, $q(\mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \theta)$. It therefore follows from 2 that $\mathcal{L}(q, \theta) \leq \ln p(\mathbf{X}|\theta)$, in other words that $\mathcal{L}(q, \theta)$ is a lower bound on $\ln p(\mathbf{X}|\theta)$.

The EM algorithm is a two-stage iterative optimization technique for finding maximum likelihood solutions. We can use the decomposition 2 to define the EM algorithm and to demonstrate that it does indeed maximize the log likelihood. Suppose that the current value of the parameter vector is $\theta^{(\text{old})}$. In the E step, the lower bound $\mathcal{L}(q, \theta^{(\text{old})})$ is maximized with respect to $q(\mathbf{Z})$ while holding θ^{old} fixed. The solution to this maximization problem is easily seen by noting that the value of $\ln p(\mathbf{X}|\theta^{(\text{old})})$ does not depend on $q(\mathbf{Z})$ and so the largest value of $\mathcal{L}(q, \theta^{(\text{old})})$ will occur when the Kullback-Leibler divergence vanishes, in other words when $q(\mathbf{Z})$ is equal to the posterior distribution $p(\mathbf{Z}|\mathbf{X}, \theta^{(\text{old})})$. In this case, the lower bound will equal the log likelihood.

In the subsequent M step, the distribution $q(\mathbf{Z})$ is held fixed and the lower bound $\mathcal{L}(q, \theta)$ is maximized with respect to θ to give some new value $\theta^{(\text{new})}$. This will cause the lower bound $\mathcal{L}$ to increase (unless it is already at a maximum), which will necessarily cause the corresponding log likelihood function to increase. Because the distribution q is determined using the old parameter values rather than the new values and is held fixed during the M step, it will not equal the new posterior distribution $p(\mathbf{Z}|\mathbf{X}, \theta^{(\text{new})})$, and hence there will be a nonzero KL divergence. The increase in the log likelihood function is therefore greater than the increase in the lower bound.

Experience Replay (Vanseijen and Sutton, 2015) (Riedmiller, 2005)

Function Approximation Function approximation is a way of combining supervised learning algorithms with learning procedures in Reinforcement Learning to be able to generalize value function into large scale MDP. There are three different objective function for function approximation.

1. Direct method
   In direct method, we are trying to minimize the square error of our estimation and the "true" value function which we temporarily assume given by an oracle.

$$J(\theta) = \frac{1}{2}(v(s) - V_\theta(s))^2$$

   The term $\frac{1}{2}$ is just for neat math like regular supervised learning, and $v(s)$ is the "true" value for state s which we want to approximate. To use the gradient descent method, we take the gradient of the $J(\theta)$, which gives us

$$\begin{aligned}\nabla_\theta J(\theta) &= \frac{1}{2} * 2 * (v(s) - V_\theta(s))\nabla_\theta V_\theta(s) \\ &= (v(s) - V_\theta(s))\nabla_\theta V_\theta(s)\end{aligned}$$

   Note, in the equation above, $v(s)$ is just a constant number. Since we don't really have a oracle in learning, we need to find a way to estimate the "true" value function. Various method can be used here. For example, we can use the temporal difference learning rule, thus the gradient become $(r + \gamma V_\theta(s') - V_\theta(s))\nabla_\theta V_\theta(s)$, or we can use the monte carlo estimation, then the gradient become $(G_t - V_\theta(s))\nabla_\theta V_\theta(s)$ or eligibility trace.

   However, this direct method has some issue with convergence. detail here

2. Fixed point method
   Another kind of objective function is based on the "fixed point" notion in contraction. With the contraction property, we know the value function will converge to a fixed point

$$V = TV$$

   where T is the bellman operator. Thus, after convergence, our value function should fulfill the Bellman equation which gives the second objective function

$$\begin{aligned}J(\theta) &= \frac{1}{2}(V_\theta(s) - TV_\theta(s))^2 \\ &= \frac{1}{2}(V_\theta(s) - (r + \gamma V_\theta(s')))^2\end{aligned}$$

   when we take the gradient of this cost function $J(\theta)$, we have

$$\begin{aligned}\nabla_\theta J(\theta) &= \frac{1}{2} * 2 * (r + \gamma V_\theta(s') - V_\theta(s))\nabla_\theta(r + \gamma V_\theta(s') - V_\theta(s)) \\ &= (r + \gamma V_\theta(s') - V_\theta(s))(\nabla_\theta \gamma V_\theta(s') - \nabla_\theta V_\theta(s)) \\ &= (r + \gamma V_\theta(s') - V_\theta(s))(\gamma \nabla_\theta V_\theta(s') - \nabla_\theta V_\theta(s))\end{aligned}$$

   need more stuff here

3. Projected fixed point method

Gaussian Mixture Model By taking linear combinations of more basic distributions such as Gaussians, an probabilistic models known as mixture distributions can be formulated. This linear combination of Gaussians can give rise to very complex densities. By using a sufficient number of Gaussians, and by adjusting their means and covariances as well as the coefficients in the linear combination, almost any continuous density can be approximated to arbitrary accuracy. We therefore consider a superposition of K Gaussian densities of the form

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\, u_k, \Sigma_k) \tag{3}$$

which is called a *Gaussian Mixture Model*. Each Gaussian density $\mathcal{N}(x|\mu_k, \Sigma_k)$ is called a *component* of the mixture and has its own mean μ_k and covariance Σ_k.

The parameters π_k in equation are called *mixing coefficients*. If we integrate both sides of former equation with respect to x, and note that both $p(x)$ and the individual Gaussian components are normalized, we obtain

$$\begin{aligned}\int p(x)\mathrm{d}x &= \int \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)\mathrm{d}x \\ 1 &= \sum_{k=1}^{K} \pi_k \int \mathcal{N}(x|\mu_k, \Sigma_k)\mathrm{d}x \\ 1 &= \sum_{k=1}^{K} \pi_k\end{aligned}$$

Also, the requirement that $p(x) \geq 0$, together with $\mathcal{N}(x|\mu_k, \Sigma_k) \geq 0$, implies $\pi_k \geq 0$ for all k. Combining this with the condition $\sum_{k=1}^{K} \pi_k = 1$ we obtain

$$0 \leq \pi_k \leq 1$$

We therefore see that the mixing coefficients satisfy the requirements to be probabilities. From the sum and product rules, the marginal density is given by

$$p(x) = \sum_{k=1}^{K} p(k)p(x|k)$$

which is equivalent to the equation 3 in which we can view $\pi_k = p(k)$ as the prior probability of picking the k^{th} component, and the density $\mathcal{N}(x|\mu_k, \Sigma_k) = p(x|k)$ as the probability of x conditioned on k. an important role is played by the posterior probabilities $p(k|x)$, which are also known as *responsibilities*. From Bayes' theorem these are given by

$$\begin{aligned}\gamma_k(x) &= p(k|x) \\ &= \frac{p(k)p(x|k)}{\sum_l p(l)p(x|l)} \\ &= \frac{\pi_k \mathcal{N}(x|\mu_k, \Sigma_k)}{\sum_l \pi_l \mathcal{N}(x|\mu_l, \Sigma_l)}\end{aligned}$$

GLIE GLIE stands for "Greedy in the Limit of Infinite Exploration" (Singh et al., 2000). The learning policies in RL can be divided into two broad categories: a *decay exploration* strategy which become more and more greedy and *persistent exploration* which always maintain a fix exploration rate. The advantage of the first one is that we can eventually converge to the optimal policy. The second one may have the advantage always be adaptive but may not converge to the optimal. (In here, we talk about convergence in the sense that the behavior will become optimal. It is possible that some of the algorithm converge to the correct Q-value but still behave randomly with some probability by using persistent exploration strategy, Q-learning with fix ϵ-greedy for example). We may want to consider this in the context of on-policy&off-policy.

If a *decay exploration* strategy has the following two characters:

1. each action is executed infinitely often in every state that is visited infinitely often, and
2. in the limit, the learning policy is greedy with respect to the Q-value function with probability 1.

Than we can consider this decay exploration strategy GLIE. Some example of GLIE include Boltzmann Selection, ϵ-greedy.

Kullback-Leibler Divergence Consider some unknown distribution $p(x)$, and suppose that we have modelled this using an approximating distribution $q(x)$. If we use $q(x)$ to construct a coding scheme for the purpose of transmitting values of x to a receiver (see Entropy for justification of example), then the average additional amount of information (in nats) required to specify the value of x (assuming we choose an efficient coding scheme) as a result of using $q(x)$ instead of the true distribution $p(x)$ is given by

$$\begin{aligned}\mathrm{KL}(p||q) &= -\int p(x)\ln q(x)\mathrm{d}x - \left(-\int p(x)\ln p(x)\mathrm{d}x\right) \\ &= -\int p(x)\ln\left\{\frac{q(x)}{p(x)}\right\}\mathrm{d}x \\ &= \int p(x)\ln\left\{\frac{p(x)}{q(x)}\right\}\mathrm{d}x\end{aligned}$$

This is known as *Kullback-Leibler divergence* or *relative entropy* between the distribution $p(x)$ and $q(x)$.

Kullback-Leibler divergence satisfies $\mathrm{KL}(p||q) \geq 0$ with equality if, and only if, $p(x) = q(x)$. Although it is often intuited as a way of measuring the distance between probability distributions, the Kullback-Leibler divergence is not a true metric. It does not obey the triangle inequality, and in general $\mathrm{KL}(P||Q)$ does not equal $\mathrm{KL}(Q||P)$.

KL-divergence can be used to task of density estimation. Suppose that data is being generated from an unknown distribution $p(x)$ that we wish to model. We can try to approximate this distribution using some parametric distribution $q(x|\theta)$, governed by a set of adjustable parameters θ, for example a multivariate Gaussian. One way to determine θ is to minimize the Kullback-Leibler divergence between $p(x)$ and $q(x|\theta)$ with respect to θ. We cannot do this directly because we don't know $p(x)$. Suppose, however, that we have observed a finite set of training points x_n, for $n = 1, \ldots, N$, drawn from $p(x)$. Then the expectation with respect to $p(x)$ can be approximated by a finite sum over these points, so that

$$\begin{aligned}\mathrm{KL}(p||q) &= \int p(x)\ln\left\{\frac{p(x)}{q(x|\theta)}\right\}\mathrm{d}x \\ &= \int p(x)\{-\ln q(x|\theta) + \ln p(x)\}\mathrm{d}x \\ &= \sum_{n=1}^{N}\{-\ln q(x_n|\theta) + \ln p(x_n)\}\end{aligned}$$

The second term on the right-hand side of the equation is independent of θ, and the first term is the negative log likelihood function for θ under the distribution $q(x|\theta)$ evaluated using the training set. Thus we see that minimizing this Kullback-Leibler divergence is equivalent to maximizing the likelihood function.

Importance Sampling Unlike rejection sampling or other sampling method for standard distribution. The technique of *importance sampling* provides a framework for approximating expectations directly but does not itself provide a mechanism for drawing samples from distribution $p(\mathbf{z})$. Suppose, that it is impractical to sample directly from $p(\mathbf{z})$ but that we can evaluate $p(\mathbf{z})$ easily for any given value of $\mathbf{z}$. Like rejection sampling, we have a proposal distribution $q(\mathbf{z})$. Given the sample set $\{\mathbf{z}^{(l)}\}$, then we can use the following

derivation

$$\begin{aligned}\mathbb{E}_p[f] &= \int p(\mathbf{z}) f(\mathbf{z}) \mathrm{d}\mathbf{z} \\ &= \int p(\mathbf{z}) f(\mathbf{z}) \frac{q(\mathbf{z})}{q(\mathbf{z})} \mathrm{d}\mathbf{z} \\ &= \int q(\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} f(\mathbf{z}) \mathrm{d}\mathbf{z} \\ &= \mathbb{E}_q \left[\frac{p(\mathbf{z})}{q(\mathbf{z})} f \right] \\ &= \frac{1}{L} \sum_{l=1}^{L} \frac{p(\mathbf{z}^{(l)})}{q(\mathbf{z}^{(l)})} f(\mathbf{z}^{(l)})\end{aligned}$$

The quantities $r_l = p(\mathbf{z}^{(l)})/q(\mathbf{z}^{(l)})$ is the *importance weights*, and they correct the bias introduced by sampling from the wrong distribution. Unlike rejection sampling, all of the samples generated in importance sampling are retained.

Now consider the case same as in rejection sampling

$$p(\mathbf{z}) = \frac{1}{Z_p} \widetilde{p}(\mathbf{z})$$

We make the further assumption that the proposal distribution has the same property

$$q(\mathbf{z}) = \frac{1}{Z_q} \widetilde{q}(\mathbf{z})$$

We then have

$$\begin{aligned}\mathbb{E}[f] &= \int p(\mathbf{z}) f(\mathbf{z}) \mathrm{d}\mathbf{z} \\ &= \frac{Z_q}{Z_p} \int f(\mathbf{z}) \frac{\widetilde{p}(\mathbf{z})}{\widetilde{q}(\mathbf{z})} q(\mathbf{z}) \mathrm{d}\mathbf{z} \\ &= \frac{Z_q}{Z_p} \frac{1}{L} \sum_{l=1}^{L} \widetilde{r}_l f(\mathbf{z}^{(l)})\end{aligned}$$

where $\widetilde{r}_l = \widetilde{p}(\mathbf{z})/\widetilde{q}(\mathbf{z})$. The ratio of Z_p/Z_q can be evaluate using the same sample set.

$$\begin{aligned}\frac{Z_p}{Z_q} &= \frac{\int \widetilde{p}(\mathbf{z}) \mathrm{d}\mathbf{z}}{Z_p} = \int \frac{\widetilde{p}(\mathbf{z})}{Z_q} \mathrm{d}\mathbf{z} \\ &= \int \frac{\widetilde{p}(\mathbf{z})}{\widetilde{q}(\mathbf{z})/q(\mathbf{z})} \mathrm{d}\mathbf{z} \\ &= \int \frac{\widetilde{p}(\mathbf{z})}{\widetilde{q}(\mathbf{z})} q(\mathbf{z}) \mathrm{d}\mathbf{z} \\ &= \frac{1}{L} \sum_{l=1}^{L} \widetilde{r}_l\end{aligned}$$

With this result, we can obtain the value of $\mathbb{E}[f]$.

$$\begin{aligned}\mathbb{E}[f] &= \frac{Z_q}{Z_p}\frac{1}{L}\sum_{l=1}^{L}\widetilde{r}_l f(\mathbf{z}^{(l)}) \\ &= \frac{L}{\sum_{l=1}^{L}\widetilde{r}_l}\frac{1}{L}\sum_{l=1}^{L}\widetilde{r}_l f(\mathbf{z}^{(l)}) \\ &= \sum_{l=1}^{L}\frac{\widetilde{r}_l}{\sum_{l=1}^{L}\widetilde{r}_l} f(\mathbf{z}^{(l)}) \\ &= \sum_{l=1}^{L} w_l f(\mathbf{z}^{(l)})\end{aligned}$$

where we define

$$w_l = \frac{\widetilde{r}_l}{\sum_{l=1}^{L}\widetilde{r}_l} = \frac{\widetilde{p}(\mathbf{z}^{(l)})/\widetilde{q}(\mathbf{z}^{(l)})}{\sum_l \widetilde{p}(\mathbf{z}^{(l)})/\widetilde{q}(\mathbf{z}^{(l)})}$$

As with rejection sampling, the success of the importance sampling approach depends crucially on how well the sampling distribution $q(\mathbf{z})$ matches the desired distribution $p(\mathbf{z})$. If, as is often the case, $p(\mathbf{z})f(\mathbf{z})$ is strongly varying and has a significant proportion of its mass concentrated over relatively small regions of $\mathbf{z}$ space, then the set of importance weights $\{r_l\}$ may be dominated by a few weights having large values, with the remaining weights being relatively insignificant. Thus the effective sample size can be much smaller than the apparent sample size L. The problem is even more severe if none of the samples falls in the regions where $p(\mathbf{z})f(\mathbf{z})$ is large. In that case, the apparent variances of r_l and $r_l f(\mathbf{z}^{(l)})$ may be small even though the estimate of the expectation may be severely wrong. Hence a major drawback of the importance sampling method is the potential to produce results that are arbitrarily in error and with no diagnostic indication. This also highlights a key requirement for the sampling distribution $q(\mathbf{z})$, namely that it should not be small or zero in regions where $p(\mathbf{z})$ may be significant.

Inverse Reinforcement Learning Inverse Reinforcement Learning, also known as Inverse Optimal Control or Inverse Optimal Planning, can be characterized informally as follows (Ng and Russell, 2000).

Given 1) measurements of an agent's behavior over time, in a variety of circumstances, 2) if needed, measurements of the sensory inputs to that agent; 3) if available, a model of the environment.

Determine the reward function being optimized.

A closely related topic to IRL is behavior cloning, where we directly learn the teacher's policy using supervised learning. However, the reason we want to use IRL instead of behavior cloning is that the reward function is often much more succinct than the optimal policy. Reward functions can generalize the expert's behavior to the state space regions not covered by the demonstration. For example, the transition model can change and thus, the optimal policy change as well. In addition to that, behavioral cloning cannot be applied whenever the expert demonstrates actions that cannot be executed by the learner (think at a humanoid robot that learns by observing a human expert).

Inverse Reinforcement Learning was first proposed as machine learning problem in (Ng and Russell, 2000). The basic principle is to find R^* (optimal reward function) given the demonstration from expert's policy π^* such that

$$\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R^*(s_t)|\pi^*\right] \geq \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R^*(s_t)|\pi\right] \quad \forall\pi$$

However, this is an ill-posed problem with several challenges.

- Demonstrated behavior may be optimal for many different reward function, including degeneracies (e.g. $r = 0$ for all states)

- There may be no feasible reward function apart from degeneracies for which demonstrated behavior is optimal
- Usually only the trajectories of expert are observed instead of the policy π^*.
- Expert's policy may be necessary optimal

We start to look the some of the mathematical formulations for this problem.

Tabular Case We first consider the simplest case where we use tabular representations. Since this is tabular case with finite states and actions, we can write everything in matrix form. In here, we assume the reward function is depend on the state instead of state and action, that is, we have $R(s)$ instead of $R(s, a)$. And we can write the reward function as a vector, where ith element represent the reward for ith state.

$$\mathbf{r} = \begin{bmatrix} R(s_1) & R(s_2) & \dots & R(s_n) \end{bmatrix}^T$$

Similarly, the value function can also represent this way.

$$\mathbf{v}^\pi = \begin{bmatrix} V^\pi(s_1) & V^\pi(s_2) & \dots & V^\pi(s_n) \end{bmatrix}^T$$

For each action a, we let $\mathbf{P_a}$ be the $N \times N$ matrix, in which element (i, j) gives the probability of transitioning to state j upon taking action a in state i.

$$\mathbf{P_a} = \begin{bmatrix} P_{11} & \dots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \dots & P_{nn} \end{bmatrix}$$

We further let the symbol $\succ$ and $\succeq$ denote strict and non-strict vectorial inequality, i.e., $x \succ y$ if and only if $\forall i \; x_i > y_i$. Recall the Bellman equations

$$V^\pi(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^\pi(s')$$
$$Q^\pi(s, a) = R(s) + \gamma \sum_{s'} P(s'|s, a) V^\pi(s')$$

And the definition of optimal policy

$$\pi^*(s) \in \arg\max_{a \in A} Q^\pi(s, a) \quad \text{for all } s \in S$$

Suppose the optimal policy is to play a_1 for all states, so that $\pi^*(s_i) \equiv a_1$. We convert the first Bellman equation of state values into matrix form with the notation we defined before. So we have

$$\mathbf{v}^{\pi^*} = \mathbf{r} + \gamma \mathbf{P_{a_1}} \mathbf{v}^{\pi^*}$$

Thus, we can have

$$\mathbf{v}^{\pi^*} = (\mathbf{I} - \gamma \mathbf{P_{a_1}})^{-1} \mathbf{r}$$

Now consider the second Bellman equation

$$\begin{aligned} & a_1 \equiv \pi^*(s) \in \arg\max_{a \in A} \sum_{s'} P(s'|s, a) V^{\pi^*}(s') \quad \forall s \in S \\ \Rightarrow & \sum_{s'} P(s'|s, a_1) V^{\pi^*}(s') \geq \sum_{s'} P(s'|s, a) V^{\pi^*}(s') \quad \forall s \in S, a \in A \\ \Rightarrow & \mathbf{P_{a_1}} \mathbf{v}^{\pi^*} \succeq \mathbf{P_a} \mathbf{v}^{\pi^*} \quad \forall a \in A \backslash a_1 \\ \Rightarrow & \mathbf{P_{a_1}} (\mathbf{I} - \gamma \mathbf{P_{a_1}})^{-1} \mathbf{r} \succeq \mathbf{P_a} (\mathbf{I} - \gamma \mathbf{P_{a_1}})^{-1} \mathbf{r} \\ \Rightarrow & (\mathbf{P_{a_1}} - \mathbf{P_a})(\mathbf{I} - \gamma \mathbf{P_{a_1}})^{-1} \mathbf{r} \succeq 0 \end{aligned} \tag{4}$$

For finite-state MDPs, this result characterizes the set of all reward function that are solutions to the inverse reinforcement learning problems. However, this result have two problem. The first one is the degeneracies problem we mentioned before, where $\mathbf{r} = 0$ is always a solution. Second, for most MDPs, it also seems likely that there are many choices of $\mathbf{r}$ that meet the condition. Ng and Russell solve this problem by formulating it as an linear programming problem and add in the penalty term (Ng and Russell, 2000). They first require that the reward function should make the π^* optimal and also make any other single step deviated from π^* as cost as possible. Thus, for all the reward function $\mathbf{r}$ satisfied the previous equation, they choose to the one so to maximize

$$\sum_{s \in S} \left(Q^{\pi^*}(s, a_1) - \max_{a \in A \backslash a_1} Q^{\pi^*}(s, a) \right)$$

That is, they seek to maximize the sum of the difference between the value of the optimal action and the value of the second-best action. In addition, they pointed out that, all other things being equal, solutions with mainly small rewards are "simpler" and therefore should be preferable. As a consequence, they add in a ℓ_1-penalty term $-\lambda||\mathbf{r}||_1$, so that with sufficiently large λ, $\mathbf{r}$ will often be nonzero in only a few states, consistent with the idea of being "simple". So the final optimization problem is:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{N} \min_{a \in \{a_2, \ldots, a_k\}} \left\{ \left(\mathbf{P}_{\mathbf{a_1}}(i) - \mathbf{P}_{\mathbf{a}}(i)\right) \left(\mathbf{I} - \gamma \mathbf{P}_{\mathbf{a_1}}\right)^{-1} \mathbf{r} \right\} - \lambda ||\mathbf{r}||_1 \\ \text{subject to} \quad & (\mathbf{P}_{\mathbf{a_1}} - \mathbf{P}_{\mathbf{a}})(\mathbf{I} - \gamma \mathbf{P}_{\mathbf{a_1}})^{-1} \mathbf{r} \succeq 0 \quad \forall a \in A \backslash a_1 \\ & |\mathbf{r}_i| \leq R_{\max}, i = 1, \ldots, N \end{aligned}$$

where $\mathbf{P}_{\mathbf{a}}(i)$ denote the ith row of $\mathbf{P}_{\mathbf{a}}$. Several things need to be mentioned for this formulation. The first one is that the constraint on the absolute value of reward function are added in, so the reward we receive cannot larger than $R_{\max}$. The second one is that how do we determine the value of λ. As the value of λ increase, there will be a phase transition at some point λ_0, such that the optimal $\mathbf{r}$ is not all zero for $\lambda < \lambda_0$, and $\mathbf{r} = 0$ for $\lambda \geq \lambda_0$. So we want to find the $\lambda = \lambda_0^-$, a value just before the phase transition, probably through binary search, since it is gives the "simplest" $\mathbf{r}$. The last thing we we want to mention is the maximization goal. In the linear programming formulation, we simply write it in the matrix form.

$$\begin{aligned} & \sum_{s \in S} \left(Q^{\pi^*}(s, a_1) - \max_{a \in A \backslash a_1} Q^{\pi^*}(s, a) \right) \\ \Rightarrow & \sum_{s \in S} \left(R(s) + \gamma \sum_{s'} P(s'|s, a_1) V^{\pi^*}(s') - \max_{a \in A \backslash a_1} \left(R(s) + \gamma \sum_{s'} P(s'|s, a) V^{\pi^*}(s') \right) \right) \\ \Rightarrow & \sum_{s \in S} \left(\gamma \sum_{s'} P(s'|s, a_1) V^{\pi^*}(s') - \max_{a \in A \backslash a_1} \left(\gamma \sum_{s'} P(s'|s, a) V^{\pi^*}(s') \right) \right) \\ \Rightarrow & \sum_{s \in S} \min_{a \in A \backslash a_1} \left(\gamma \sum_{s'} P(s'|s, a_1) V^{\pi^*}(s') - \gamma \sum_{s'} P(s'|s, a) V^{\pi^*}(s') \right) \\ \Rightarrow & \gamma \sum_{s \in S} \min_{a \in A \backslash a_1} \left(\sum_{s'} P(s'|s, a_1) V^{\pi^*}(s') - \sum_{s'} P(s'|s, a) V^{\pi^*}(s') \right) \\ \Rightarrow & \gamma \sum_{i=0}^{N} \min_{a \in A \backslash a_1} \left(\mathbf{P}_{\mathbf{a_1}}(i) \mathbf{v}^{\pi^*} - \mathbf{P}_{\mathbf{a}}(i) \mathbf{v}^{\pi^*} \right) \\ \Rightarrow & \gamma \sum_{i=0}^{N} \min_{a \in A \backslash a_1} \left(\left(\mathbf{P}_{\mathbf{a_1}}(i) - \mathbf{P}_{\mathbf{a}}(i)\right) \mathbf{v}^{\pi^*} \right) \\ \Rightarrow & \gamma \sum_{i=0}^{N} \min_{a \in A \backslash a_1} \left(\left(\mathbf{P}_{\mathbf{a_1}}(i) - \mathbf{P}_{\mathbf{a}}(i)\right) (\mathbf{I} - \gamma \mathbf{P}_{\mathbf{a_1}})^{-1} \mathbf{r} \right) \end{aligned}$$

However, since the γ is a constant, we can drop it in the maximization goal without effect the result.

Linear Function Approximation Case Now we consider the case of infinite state spaces. The most basic assumption is that the reward function R is now a function from $S = \mathbb{R}^k$ into the reals and takes linear form.

$$R(s) = w_1\phi_1(s) + w_2\phi_2(s) + \cdots + w_k\phi_k(s)$$

where $\phi_1, \ldots, \phi_k$ are fixed, known, bounded basis functions mapping from S into $\mathbb{R}$, and w_is are the unknown parameters that we want to "fit". The constraint that a valid reward function should satisfy is

$$\mathbb{E}_{s' \sim P(s'|s,a_1)}\left[V^\pi(s')\right] \geq \mathbb{E}_{s' \sim P(s'|s,a)}\left[V^\pi(s')\right] \quad \forall s \in S, a \in A \tag{5}$$

which can be thought as the generalization of (4). This constraint have two problems. The first one is that, for infinite state spaces, there are infinitely many constraints of the form in equation (5). Second, we constraint ourselves to use linear function to represent the reward function, but the "true" reward function may not exist in this class of functions. That is, for all the reward function in linear form, the policy π^* may not be optimal for any of them. We solve the first issue by only consider a finite subset of the entire state space . To be more specific, we only consider those initial states $s_0 \sim D$, where D is the initial-state distribution. For the second problem, we are willing to relax some of the constraints by paying a penalty when they are violated. So the final linear programming formulation is

$$\begin{aligned} \text{maximize} \quad & \sum_{s_\in S_0} \min_{a \in \{a_2, \ldots, a_k\}} \left\{ p\big(\mathbb{E}_{s' \sim P(s'|s,a_1)}\left[V^\pi(s')\right] \geq \mathbb{E}_{s' \sim P(s'|s,a)}\left[V^\pi(s')\right]\big) \right\} \\ \text{subject to} \quad & |w_i| \leq 1, i = 1, \ldots, k \end{aligned}$$

where p is given by $p(x) = x$ if $x \geq 0$, $p(x) = 2x$ otherwise, and 2 is penalty weight that was heuristically chosen in the original paper.

Apprenticeship Learning via Feature Matching The ultimate goal of IRL is to learn a control policy. Thus, after recover the reward function, we need to learn the policy given that reward function. We first consider the value of a policy π

$$\begin{aligned} \mathbb{E}_{s_0 \sim D}\left[V^\pi(s_0)\right] &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t w \cdot \phi(s_t)|\pi\right] \\ &= w \cdot \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi\right] \end{aligned}$$

The last term of the right hand side is the expecte discounted accumulated feature value vector, or more succinctly the feature expectations $\mu(\pi)$ (Abbeel and Ng, 2004).

$$\mu(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi\right] \in \mathbb{R}^k$$

(Abbeel and Ng, 2004) (Ziebart et al., 2008) (Syed and Schapire, 2007) (Finn et al., 2016) (Levine et al., 2011)

Lipschitz Continuity Suppose we are given two metric space (X, d_X) and (Y, d_Y), where d_X denotes the metric on the set X and d_Y is the metric on set Y, we have the following definition.

- A function $f : X \to Y$ is called *Lipschitz continuous* at $x \in X$ if there exists a real constant $C \geq 0$ such that, for all x' in X,

$$\mathrm{d}_Y(f(x), f(x')) \leq C\mathrm{d}_X(x, x')$$

(Eriksson et al., 2013)

Linear Quadratic Regulator When we consider a special system with linear dynamic and quadratic cost, we have some very nice algorithm so solve this questions.

$$\begin{aligned} x_{t+1} &= A_t x_t + B_t u_t + w_t \quad \text{for} \quad t = 0, 1, \ldots, N-1 \\ C(x_t, u_t) &= x_t^\top Q_t x_t + u_t^\top R_t u_t \end{aligned}$$

In the equation above, x_t are the state with n-dimension, and u_t are the control (action) with m-dimension. A_t, B_t, Q_t, R_t are given and have appropriate dimension. Q_t are positive semidefinite symmetric and R_t are positive definite symmetric. w_t is the noise term with given probability distributions that do not depend on x_t ad u_t. It also has zero mean and finite variance.

The reason for use this kind of control is that we want to keep the state of the system close to the origin. The quadratic cost function will induce heavy penalty if the system is largely deviate from the it. So now we apply Dynamic Programming algorithm to it. First, we have the cost-to-go function of the last state N and other states.

$$\begin{aligned} J_N(x_N) &= x_N^\top Q_N x_N \\ J_t(x_t) &= \min_{u_k} \mathbb{E}\left\{x_t^\top Q_t x_t + u_t^\top R_t u_t + J_{t+1}(A_t x_t + B_t u_t + w_t)\right\} \end{aligned}$$

We can replace the t with $N-1$, then we have

$$\begin{aligned} J_{N-1}(x_{N-1}) = \min_{u_{N-1}} \mathbb{E}\big\{ & x_{N-1}^\top Q_{N-1} x_{N-1} + u_{N-1}^\top R_{N-1} u_{N-1} \\ & + (A_{N-1}x_{N-1} + B_{N-1}u_{N-1} + w_{N-1})^\top Q_N (A_{N-1}x_{N-1} + B_{N-1}u_{N-1} + w_{N-1})\big\} \end{aligned}$$

Next, we expand the term inside square brackets.

$$\begin{aligned} J_{N-1}(x_{N-1}) = x_{N-1}^\top Q_{N-1} x_{N-1} + \min_{u_{N-1}} \big[& u_{N-1}^\top R_{N-1} u_{N-1} + u_{N-1}^\top B_{N-1}^\top Q_N B_{N-1} u_{N-1} \\ & + 2 x_{N-1}^\top A_{N-1}^\top Q_N B_{N-1} u_{N-1}\big] + x_{N-1}^\top A_{N-1}^\top Q_N A_{N-1} x_{N-1} + \mathbb{E}\left\{w_{N-1}^\top Q_N w_{N-1}\right\} \end{aligned}$$

Several things to note here. First, some of the terms does not depend on u_{N-1}, thus we move them out of the minimization operator. Second, $\mathbb{E}\{w_{N-1}\} = 0$ suggests that $\mathbb{E}\{w_{N-1}^\top Q_N (A_{N-1}x_{N-1} + B_{N-1}u_{N-1})\}$ is also zero. Third, we reverse the position of some terms due to the transpose. If we differentiate the equation with respect to u_{N-1} and set the derivative to zero, we get

$$(R_{N-1} + B_{N-1}^\top Q_N B_{N-1}) u_{N-1} = -B_{N-1}^\top Q_N A_{N-1} x_{N-1}$$

Since R_{N-1} is positive definite and $B_{N-1}^\top Q_{N-1} B_{N-1}$ is positive semidefinite, matrix multiplying u_{N-1} on the left is positive definite (and hence invertible). Thus, the minimizing control vector is given by

$$u_{N-1}^* = -(R_{N-1} + B_{N-1}^\top Q_N B_{N-1})^{-1} B_{N-1}^\top Q_N A_{N-1} x_{N-1}$$

By substitution into the expression for J_{N-1}, we have

$$J_{N-1}(x_{N-1}) = x_{N-1}^\top K_{N-1} x_{N-1} + \mathbb{E}\left\{w_{N-1}^\top Q_N w_{N-1}\right\}$$

and matrix K_{N-1} should be

$$K_{N-1} = A_{N-1}^\top (Q_N - Q_N B_{N-1}(B_{N-1}^\top Q_N B_{N-1} + R_{N-1})^{-1} B_{N-1}^\top Q_N) A_{N-1} + Q_{N-1}$$

LSPI Need more time to work on it ...

LSTD Need more time to work on it . . . Related paper

Long-Short Term Memory Need more time to work on it . . .

Markov Chain Monte Carlo Need more time to work on it . . .

Markov Decision Process blah blah here

Markov Decision Process can be seen as a extension of Markov Chain with additional action set (allowing selection) and reward function (motivation). It can be reduced to Markov chain if we have only one action per state and same reward for all the state.

Metric Space Need more time to work on it . . .

Definition (Metric Space, metric). A *metric space* is a pair (X, d), where X is a set and d is a metric on X (distance function on X), that is, a function defined[1] on $X \times X$ such that for all $x, y, z \in X$ we have:

1. d is real-valued, finite and nonnegative.
2. $d(x, y) = 0$ if and only if $x = y$
3. $d(x, y)$ = $d(y, x)$ (Symmetry)
4. $d(x, y) \leq d(x, z) + d(z, y)$ (Triangle inequality)

In this definition, X is called the underlying set of (X, d). Its elements are called points. Here are some examples of metric space:

- Real line $\mathbb{R}$
  The set of all real numbers and $d(x, y) = |x - y|$
- Euclidean plane $\mathbb{R}^2$
  Suppose we have $x = (\epsilon_1, \epsilon_2)$ and $y = (\upsilon_1, \upsilon_2)$. The euclidean metric defined by
  $$d(x, y) = \sqrt{(\epsilon_1 - \upsilon_1)^2 + (\epsilon_2 - \upsilon_2)^2}$$
- Euclidean space $\mathbb{R}^n$
  If $x = (\epsilon_1, \ldots, \epsilon_n)$ and $y = (\upsilon_1, \ldots, \upsilon_n)$, then euclidean metric defined by
  $$d(x, y) = \sqrt{(\epsilon_1 - \upsilon_1)^2 + \ldots + (\epsilon_n - \upsilon_n)^2}$$
- Unitary space $\mathbb{C}^n$
  a n-dimensional unitary space $\mathbb{C}^n$ is the space of all ordered n-tuples of complex numbers with distance function
  $$d(x, y) = \sqrt{|\epsilon_1 - \upsilon_1|^2 + \ldots + |\epsilon_n - \upsilon_n|^2}$$
- Sequence space l^∞
  This example and the next one give a first impression of how surprisingly the concept of a metric space is. As a set X we take the set of all bounded sequences of complex numbers; that is, every element of X is a complex sequence
  $$x = (\epsilon_1, \epsilon_2, \ldots) \quad \text{briefly} \quad x = (\epsilon_j)$$
  such that for all $j = 1, 2, \ldots$ we have
  $$|\epsilon_j| \leq c_x$$

[1] The symbol $\times$ denotes the Cartesian product of sets: $A \times B$ is the set of all ordered pairs (a, b), where $a \in A$, and $b \in B$. Hence, $X \times X$ is the set of all ordered pairs of elements of X

where c_x is a real number which may depend on x, but does not depend on j. We choose the metric defined by

$$d(x, y) = \sup_{j \in \mathbb{N}} |\epsilon_j - \upsilon_j|$$

where $y = (\upsilon_j) \in X$ and $\mathbb{N} = 1, 2, \ldots$, and sup denotes the supremum (least upper bound).

- Function space $C[a, b]$
  As a set X we take the set of all real-valued functions $x, y, \ldots$ which are functions of an independent real variable t and are defined and continuous on a given closed interval $J = [a.b]$. Choosing the metric defined by

$$d(x, y) = \max_{t \in J} |x(t) - y(t)|$$

  where max denotes the maximum, we obtain a metric space which is denoted by $C[a, b]$. (The letter C suggests "continuous.") This is a function space because every point of $C[a, b]$ is a function.

(Kreyszig, 1989).

Norm a function that assigns a strictly positive length or size to each vector in a vector space)

Monte Carlo Method Monte Carlo method is a way of making the prediction in model-free environment. The question it wants to solve is that suppose we have a policy π known, how good is this policy? In this case, we evaluate the policy by giving the method episodes of experience $\{s_1, a_1, r_2, \ldots, s_T\}$ generated by following policy π and wants the value function V^π as output.

As we know, the value of being in a state s is the expectation of the discounted rewards received afterwards.

$$V^\pi(s) = \mathbb{E}_\pi[r_{t+1} + \gamma r_{t+2} + \ldots + \gamma^{T-1} r_T]$$

two methods can be used here : first-visit and every-visit method

Off-Policy Policy Gradient Method In REINFORCE method, when we try to evaluate the expected return of a policy π, we have to run the policy several time to be able to have reasonable estimation of how good the policy is. Let $J(\theta)$ denote the expected return of policy π_θ, and τ denote the trajectory (essentially a state-action sequence : $s_0, a_0, s_1, a_1, \ldots, s_T, a_T$) or rollout or history (these terms are interchangeable) of executing the policy π_θ, then we know

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{H} \gamma^t R(s_t, u_t) | \pi_\theta\right] \\ &= \sum_\tau p(\tau|\theta) R(\tau) \end{aligned}$$

where $P(\tau|\theta)$ is the probability of having a trajectory τ by following policy π_θ and $R(\tau)$ is just the accumulated reward of that trajectory. In policy gradient method, after we evaluate the policy π_θ, we may want to improve it and use a new policy. Thus, the sample we collected during the process of following policy π_θ are discarded. However, it will preferable if we can reuse the data gathered of following one policy to estimate the value of following another policy. The method "likelihood ratio" estimation make this data reuse possible.

In practice, if we want to evaluate $J(\theta)$, we may want to draw the rollout samples from the distribution induced by policy π_θ. After taking N samples $\{\tau_0, \tau_1, \ldots, \tau_N\}$, we have a unbiased estimator:

$$\hat{J(\theta)} = \frac{1}{N} \sum_i R(\tau_i)$$

Imagine, however, instead of π_θ, we only have $\pi_{\theta'}$ available, we can do some trick like this

$$\begin{aligned} J(\theta) &= \sum_\tau p(\tau|\theta) R(\tau) \\ &= \sum_\tau p(\tau|\theta) \frac{p(\tau|\theta')}{p(\tau|\theta')} R(\tau) \\ &= \sum_\tau p(\tau|\theta') \frac{p(\tau|\theta)}{p(\tau|\theta')} R(\tau) \\ &= \mathbb{E}_{\pi_{\theta'}} \left[\frac{p(\tau|\theta)}{p(\tau|\theta')} R(\tau) \right] \end{aligned}$$

Note that, $p(\tau|\theta)$ and $p(\tau|\theta')$ is the probability of same sample τ under different distribution induced by θ and θ'. Now, we can estimate the $J(\theta)$ with respect to the the distribution induced by $\pi_{\theta'}$. This method of estimating one expectation with respect to another distribution is called importance sampling, which is widely used for off-policy learning. So, we can rewrite the $\hat{J(\theta)}$ as

$$\hat{J(\theta)} = \frac{1}{N} \sum_i R(\tau_i) \frac{p(\tau|\theta)}{p(\tau|\theta')}$$

Note, in the equation above, we have a term $\frac{p(\tau|\theta)}{p(\tau|\theta')}$. Since we don't have the model of the world, it's not possible to directly compute $p(\tau|\theta)$ or $p(\tau|\theta')$. But if we expand the fraction term (note that $p(\tau|\theta)$ and $p(\tau|\theta')$ both evaluate on the same trajectory sample τ, therefore the state action sequence is same for both), we can see that

$$\begin{aligned} \frac{p(\tau|\theta)}{p(\tau|\theta')} &= \frac{p(s_0) \prod_{t=0}^T \left[p(s_{t+1}|s_t, a_t) p(a_t|s_t, \theta) \right]}{p(s_0) \prod_{t=0}^T \left[p(s_{t+1}|s_t, a_t) p(a_t|s_t, \theta') \right]} \\ &= \frac{p(s_0) \prod_{t=0}^T p(s_{t+1}|s_t, a_t) \prod_{t=0}^T p(a_t|s_t, \theta)}{p(s_0) \prod_{t=0}^T p(s_{t+1}|s_t, a_t) \prod_{t=0}^T p(a_t|s_t, \theta')} \\ &= \frac{\prod_{t=0}^T p(a_t|s_t, \theta)}{\prod_{t=0}^T p(a_t|s_t, \theta')} \\ &= \frac{\prod_{t=0}^T \pi_\theta(a_t|s_t)}{\prod_{t=0}^T \pi_{\theta'}(a_t|s_t)} \end{aligned}$$

Thus, we should be able to calculate the likelihood $\frac{p(\tau|\theta)}{p(\tau|\theta')}$ for any two policies θ and θ'. Actually, we can have a mixture distributions, where $p(\tau|\theta')$ is replaced by $\frac{1}{N} \sum_j p(\tau_j|\theta_j)$ where τ_j are trajectory of following θ_j, then we can make use of multiple source of distributions.

On-policy and Off-policy An RL algorithm can be essentially divided into two parts, the *learning policy* and *update rule*. The first one is a non-stationary policy that maps experience (state visited, action chosen, reward received) to into a currently choice of action. The second part is how the algorithm uses experience to change its estimation of the optimal value function. In off-policy algorithm, the *update rules* doesn't have relationship with *learning policy*, that is the *update rules* doesn't care the what action agent take. Q-learning can be consider as the off-policy algorithm.

$$Q_{t+1}(s, a) = (1 - \alpha) Q_t(s, a) + \alpha (r_t + \gamma \max_{a'} Q(s', a'))$$

We can see that the Q-value is update based on the $\max_{a'} Q(s', a')$, which doesn't depend on the action the agent was taking.

However, if we take a look of SARSA(0), which is very similar to Q-learning.

$$Q_{t+1}(s,a) = (1-\alpha)Q_t(s,a) + \alpha(r_t + \gamma Q(s',a'))$$

We can see the update is based on the Q-value of the next action of the agent. Thus it is an on-policy algorithm. The convergence condition are heavily depend on the *learning policy*, The Q-value of SARSA(0) can only converge to optimality in the limit only if the learning policy behavior optimal in the limit. The SARSA(0) and Q-learning will be same if we use greedy action selection strategy.

Detail see (Singh et al., 2000).

Policy Gradient Reinforcement Learning Instead of determining the action based on value function, another method of determine the action to take is the direct policy search method. It is well known that value-function combined with function approximation are unable to converge to any policy even for simple MDPs (in mean-squared-error, residual-gradient, temporal-difference, and dynamic-programming sense)

Let θ denote the parameters of the function approximator, neural network for example, and $J(\theta)$ the performance of the corresponding policy (e.g. the average reward per step). Then in the policy gradient approach, we can update the policy parameters proportional to the gradient:

$$\Delta\theta = \alpha\frac{\partial J(\theta)}{\partial\theta}$$

Let's begin with the definition of the two formulations:

- Average reward formulation

$$J(\theta) = \lim_{n\to\infty}\frac{1}{n}\mathbb{E}\{r_1 + r_2 + \ldots + r_n|\pi\} = \sum_s d^\pi(s)\sum_a \pi(s,a)R(s,a)$$

  where $d^\pi(s)$ is the stationary distribution of the states induced by π

where α is a positive-definite step size.

Policy Gradient Theorem Using the same notation in Policy Gradient,

Policy Iteration Need more time to work on it ...

REINFORCE REINFORCE algorithm also finds an unbiased estimate of the gradient, but without the assistance of a learned value function. REINFORCE learns much more slowly than RL methods using value functions and has received relatively little attention. Learning a value function and using it to reduce the variance of the gradient estimate appears to be essential for rapid learning.

Likelihood Ratio Methods

$$\begin{aligned}
J(\theta) &= \mathbb{E}\left[r(\tau)\right] \\
&= \int_{\tau} p_\theta(\tau)\, r(\tau) \\
\nabla_\theta J(\theta) &= \nabla_\theta \int_{\tau} p_\theta(\tau)\, r(\tau) \\
&= \int_{\tau} \nabla_\theta p_\theta(\tau)\, r(\tau) \\
&= \int_{\tau} \nabla_\theta p_\theta(\tau)\, \frac{p_\theta(\tau)}{p_\theta(\tau)}\, r(\tau) \\
&= \int_{\tau} p_\theta(\tau)\, \frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)}\, r(\tau) \\
&= \int_{\tau} p_\theta(\tau)\, \nabla_\theta \ln p_\theta(\tau)\, r(\tau) \\
&= \mathbb{E}\left[\nabla_\theta \ln p_\theta(\tau)\, r(\tau)\right]
\end{aligned}$$

However, we know that

$$p_\theta(\tau) = p(x_0) \prod_{k=0}^{H} p(x_{k+1}|x_k, u_k)\pi_\theta(u_k|x_k)$$

Thus

$$\begin{aligned}
\nabla_\theta \ln p_\theta(\tau) &= \nabla_\theta \big[\ln p(x_0) + \sum_{k=0}^{H} (\ln p(x_{k+1}|x_k, u_k) + \ln \pi_\theta(u_k|x_k))\big] \\
&= \nabla_\theta \big[\ln p(x_0) + \sum_{k=0}^{H} \ln p(x_{k+1}|x_k, u_k) + \sum_{k=0}^{H} \ln \pi_\theta(u_k|x_k)\big] \\
&= \nabla_\theta \ln p(x_0) + \sum_{k=0}^{H} \nabla_\theta \ln p(x_{k+1}|x_k, u_k) + \sum_{k=0}^{H} \nabla_\theta \ln \pi_\theta(u_k|x_k) \\
&= 0 + 0 + \sum_{k=0}^{H} \nabla_\theta \ln \pi_\theta(u_k|x_k) \\
&= \sum_{k=0}^{H} \nabla_\theta \ln \pi_\theta(u_k|x_k)
\end{aligned}$$

Note, if instead of stochastic policy, we are using a deterministic policy, then we have

$$p_\theta(\tau) = p(x_0) \prod_{k=0}^{H} p(x_{k+1}|x_k, \pi_\theta(x_k))$$

In this case,

$$\begin{aligned}
\nabla_\theta \ln p_\theta(\tau) &= \nabla_\theta \big[\ln p(x_0) + \sum_{k=0}^{H} \ln p(x_{k+1}|x_k, \pi_\theta(x_k)) \big] \\
&= \nabla_\theta \ln p(x_0) + \sum_{k=0}^{H} \nabla_\theta \ln p(x_{k+1}|x_k, \pi_\theta(x_k)) \\
&= \nabla_\theta \ln p(x_0) + \sum_{k=0}^{H} \nabla_{u_k} \ln p(x_{k+1}|x_k, u_k) \nabla_\theta \pi_\theta(x_k) \\
&= 0 + \sum_{k=0}^{H} \nabla_{u_k} \ln p(x_{k+1}|x_k, u_k) \nabla_\theta \pi(x_k) \\
&= \sum_{k=0}^{H} \nabla_{u_k} \ln p(x_{k+1}|x_k, u_k) \nabla_\theta \pi(x_k)
\end{aligned}$$

Since we need to compute $\nabla_{u_k} \ln p(x_{k+1}|x_k, u_k)$, thus we need to know the model of the system

Rejection Sampling Due some constraints, the inverse cumulative distribution function (CDF) is intractable (no analytic form is available through the indefinite integral) for some distributions. Therefore, direct sampling from the this kind of posterior distributions in probabilistic model cannot use the technique described in basic sampling. Furthermore, it is often the case that for a given distribution

$$p(\mathbf{z}) = \frac{1}{Z_p} \widetilde{p}(\mathbf{z})$$

evaluating $\widetilde{p}(\mathbf{z})$ for any given value of $\mathbf{z}$ is easy, but to compute the normalizing constant Z_p is hard. In practice, we may only know about the part that is directly associate with z, and we treat the rest part as Z_p. For example, consider a gamma distribution

$$\text{Gam}(z|a, b) = \frac{b^a z^{a-1} \exp(-bz)}{\Gamma(a)}$$

The expression $\frac{b^a}{\Gamma(a)}$ is not relevant to z, so we can treat it as Z_p. For the cases where the vale of Z_p is hard to obtain, we want to use the sampling methods only require the knowledge of $\widetilde{p}(z)$, thus, avoid the computation of Z_p.

In order to apply rejection sampling, we need some simpler distribution q(z), sometimes called a *proposal distribution*, from which we can readily draw samples. We next introduce a constant k whose value is chosen such that $kq(z) \geq \widetilde{p}(z)$ for all values of z. The function $kq(z)$ is called the *comparison function*. Each step of the rejection sampler involves generating two random numbers. First, we generate a number z_0 from the distribution $q(z)$. Next, we generate a number u_0 from the uniform distribution over $[0, kq(z_0)]$. This pair of random numbers has uniform distribution under the curve of the function $kq(z)$. Finally, if $u_0 > \widetilde{p}(z_0)$ then the sample is rejected, otherwise u_0 is retained. Thus the pair is rejected if it doesn't lie in the region in $\widetilde{p}(z)$. The remaining pairs then have uniform distribution under the curve of $\widetilde{p}(z)$, and hence the correspond z values are distributed according to $p(z)$, as desired.

To see why the obtained sample obeying $p(z)$, we first notice that the probability of accept the sample is given by

$$p(\text{accept}|\mathbf{z}) = \frac{\widetilde{p}(\mathbf{z})}{kq(\mathbf{z})} \tag{6}$$

Thus, the probability of drawing a sample $\mathbf{z}$ following the proposed method is

$$q(\mathbf{z})p(\text{accept}|\mathbf{z}) = q(\mathbf{z})\frac{\widetilde{p}(\mathbf{z})}{kq(\mathbf{z})} = \frac{\widetilde{p}(\mathbf{z})}{k}$$

which is a probability density that integrates to 1

$$\int \frac{\widetilde{p}(\mathbf{z})}{k} \mathrm{d}\mathbf{z} = 1$$

Since $p(\mathbf{z})$ is $\widetilde{p}(\mathbf{z})$ normalized with Z_p,

$$p(\mathbf{z}) = \frac{\widetilde{p}(\mathbf{z})}{Z_p}$$

where

$$\int \frac{\widetilde{p}(\mathbf{z})}{Z_p} \mathrm{d}\mathbf{z} = 1$$

Therefore, $k = Z_p$, and $p(\mathbf{z})$ is $\frac{\widetilde{p}(\mathbf{z})}{k}$ which is normalized.

From equation (6), we see that the fraction of points that are rejected by this method depends on the ratio of the area under the unnormalized distribution, $p(z)$ to the area under the curve $kq(z)$. We therefore see that the constant k should be as small as possible subject to the limitation that $kq(z)$ must be nowhere less than $\widetilde{p}(z)$.

In many instances where we might wish to apply rejection sampling, it proves difficult to determine a suitable analytic form for the envelope distribution $q(z)$. An alternative approach is to construct the envelope function on the fly based on measured values of the distribution $p(z)$. Construction of an envelope function is particularly straightforward for cases in which $p(z)$ is log concave, in other words when $-\ln p(z)$ is a convex function.

The function $\ln p(z)$ and its gradient are evaluated at some initial set of grid points, and the intersections of the resulting tangent lines are used to construct the envelope function. Next a sample value is drawn from the envelope distribution. This is straightforward because the log of the envelope distribution is a succession of linear functions, and hence the envelope distribution itself comprises a piecewise exponential distribution of the form

$$q(z) = k_i \lambda_i \exp\{-\lambda_i (z - z_{i-1})\}$$

Once a sample has been drawn, the usual rejection criterion can be applied. If the sample is accepted, then it will be a draw from the desired distribution. If, however, the sample is rejected, then it is incorporated into the set of grid points, a new tangent line is computed, and the envelope function is thereby refined. As the number of grid points increases, so the envelope function becomes a better approximation of the desired distribution $p(z)$ and the probability of rejection decreases. This is called *adaptive rejection sampling*.

Clearly for rejection sampling to be of practical value, we require that the comparison function be close to the required distribution so that the rate of rejection is kept to a minimum. However, even with a great comparison function, the acceptance rate can still diminishe exponentially with the increasing of dimensionality. Thus, this method is more suitable for problems with one or two dimensions.

Sampling-Importance-Resampling The rejection sampling method depends in part for its success on the determination of a suitable value for the constant k. For many pairs of distributions $p(\mathbf{z})$ and $q(\mathbf{z})$, it will be impractical to determine a suitable value for k in that any value that is sufficiently large to guarantee a bound on the desired distribution will lead to impractically small acceptance rates.

As in the case of rejection sampling, the *sampling-importance-resampling* (SIR) approach also makes use of a sampling distribution $q(\mathbf{z})$ but avoids having to determine the constant k. There are two stages to the scheme. In the first stage, L samples $\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(L)}$ are drawn from $q(\mathbf{z})$. Then in the second stage, weights $w_1, \ldots, w_L$ are constructed using the method from importance sampling. Finally, a second set of L samples is drawn from previous sample set $(\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(L)})$ with probabilities given by the weights $(w_1, \ldots, w_L)$.

The resulting L samples are only approximately distributed according to $p(\mathbf{z})$, but the distribution becomes correct in the limit $L \to \infty$. To see this, consider the univariate case, and note that the cumulative distribution of the resampled values is given by

Need more time to work on it ...

Stationary Distribution Need more time to work on it ...

Stochastic Game Stochastic game can been seen as an extension of MDP.

Universal Approximator Accroding to (**?**), Multilayer feedforward networks are universal approximators. Single hidden layer feedforward networks can approximate any measurable function arbitrarily well regardless of the activation function Ψ, the dimension of the input space r, and the input space environment μ

1. universal approximators: standard multilayer feedforward networks are capable of approximating any measurable function to any desired degree of accuracy
2. there are no theoretical constraints for the success of feedforward networks
3. lack of success is due to inadequate learning, insufficient number of hidden units or the lack of a deterministic relationship between input and target
4. rate of convergence as the number of hidden units grows
5. rate of increase of the number of hidden units as the input dimension increases for a fixed accuracy

Value Iteration Need more time to work on it ...

Variational Inference Instead of using sampling method to approximate the posterior distribution of $p(\mathbf{Z}|\mathbf{X})$, we could use the distribution we come up with to approximate the real posterior. This method of using one distribution to approximate another distribution is *variational inference*, which is the deterministic way of doing approximate inference method.

Suppose we have a fully Bayesian model in which all parameters are given prior distributions. The model may also have latent variables as well as parameters, and we shall denote the set of all latent variables and parameters by $\mathbf{Z}$. Similarly, we denote the set of all observed variables by $\mathbf{X}$. For example, we might have a set of N independent, identically distributed data, for which $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$. Our probabilistic model specifies the joint distribution $p(\mathbf{X}, \mathbf{Z})$, and our goal is to find an approximation for the posterior distribution $p(\mathbf{Z}|\mathbf{X})$ as well as for the model evidence $p(\mathbf{X})$. Similar to EM, we can decompose the log marginal probability using

$$\mathcal{L}(q) = \int_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} \mathrm{d}\mathbf{Z}$$
$$\mathrm{KL}(q||p) = -\int_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} \mathrm{d}\mathbf{Z}$$

Note the difference here is that the parameter vector θ no longer appears, because now the parameters are now stochastic variables which is part of $\mathbf{Z}$. As before, we can maximize the lower bound L(q) by optimization with respect to the distribution q(Z), which is equivalent to minimizing the KL divergence. If we allow any possible choice for q(Z), then the maximum of the lower bound occurs when the KL divergence vanishes, which occurs when q(Z) equals the posterior distribution p(Z—X). However, we shall suppose the model is such that working with the true posterior distribution is intractable.

We therefore consider instead a restricted family of distributions q(Z) and then seek the member of this family for which the KL divergence is minimized. Our goal is to restrict the family sufficiently that they comprise only tractable distributions, while at the same time allowing the family to be sufficiently rich and flexible that it can provide a good approximation to the true posterior distribution. It is important to

emphasize that the restriction is imposed purely to achieve tractability, and that sub- ject to this requirement we should use as rich a family of approximating distributions as possible. In particular, there is no ?over-fitting? associated with highly flexible dis- tributions. Using more flexible approximations simply allows us to approach the true posterior distribution more closely.

Vector Space Need more time to work on it ...

References

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, 2004.

A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

K. Conrad. The contraction mapping theorem. *Expository paper. University of Connecticut, College of Liberal Arts and Sciences, Department of Mathematics*, 2014.

K. Eriksson, D. Estep, and C. Johnson. *Applied Mathematics: Body and Soul: Volume 1: Derivatives and Geometry in IR3*. Springer Science & Business Media, 2013.

C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. *CoRR*, abs/1603.00448, 2016. URL http://arxiv.org/abs/1603.00448.

E. Kreyszig. *Introductory functional analysis with applications*, volume 81. 1989.

S. Levine, Z. Popovic, and V. Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *International Conference on Neural Information Processing Systems*, pages 19–27. 2011.

A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, pages 663–670, 2000.

R. Ortner. Linear dependence of stationary distributions in ergodic markov decision processes. *Operations research letters*, 35(5):619–626, 2007.

M. Riedmiller. Neural fitted q iteration – first experiences with a data efficient neural reinforcement learning method. In *Machine Learning: ECML 2005*, volume 3720 of *Lecture Notes in Computer Science*, pages 317–328. 2005.

S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, 2000.

U. Syed and R. E. Schapire. A game-theoretic approach to apprenticeship learning. In *International Conference on Neural Information Processing Systems*, pages 1449–1456, 2007.

H. Vanseijen and R. Sutton. A deeper look at planning as learning from replay. pages 2314–2322, 2015.

B. D. Ziebart, A. Maas, J. A. D. Bagnell, and A. Dey. Maximum entropy inverse reinforcement learning. In *Proceeding of AAAI 2008*, July 2008.