CSC311
Lecture Notes

Yuchen Wang

April 1, 2020

Contents

1 Lecture 9 - k-Means and EM Algorithm **2**

1.1 Latent variable models . . . 2

1.2 Clustering . . . 2

1.3 Clustering Problem . . . 2

1.4 K-means . . . 2

1.4.1 K-means Algorithm . . . 2

1.5 Soft K-means . . . 3

1.5.1 Soft K-means Algorithm . . . 3

1.6 A Generative View of Clustering . . . 3

1.7 Expectation-Maximization algorithm . . . 4

2 Reinforcement Learning **5**

2.1 Formalizing Reinforcement Learning Problems . . . 5

2.2 Bellman Equation . . . 7

2.3 Finding the Value Function (Policy Evaluation Problem) . . . 8

2.4 Q-Learning . . . 9

2.5 Approaches to RL . . . 10

1 Lecture 9 - k-Means and EM Algorithm

1.1 Latent variable models

Assumption: data depends on some latent variables that are never observed.
Example: PCA (Even though data is very high dimensional, it can be well represented in low dimensions.)

1.2 Clustering

Grouping data points into clusters, with no observed labels, is called clustering.
Samples within a cluster are similar to each other, and samples in different clusters are dissimilar. It is an unsupervised technique. Such a distribution is multimodal, since it has multiple modes, or regions of high probability mass.

Example Clustering machine learning papers based on topic (deep learning, Bayesian models, etc)

1.3 Clustering Problem

Assumptions

1. The data $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$ lives in a Euclidean space, $\mathbf{x}^{(n)} \in \mathbb{R}^D$
2. Each data point belongs to one of K clusters
3. The data points from the same cluster are similar (i.e. close in Euclidean distance)

1.4 K-means

Objective Find cluster centers $\{\mathbf{m}_k\}_{k=1}^K$ and assignments $\{\mathbf{r}^{(n)}\}_{n=1}^N$ to minimize the sum of squared distances of data points $\{\mathbf{x}^{(n)}\}$ to their assigned cluster centers.

1. Data sample $n = 1, \dots, N : \mathbf{x}^{(n)} \in \mathbb{R}^D$ (observed)
2. Cluster center $k = 1, \dots, K : \mathbf{m}_k \in \mathbb{R}^D$ (not observed)
3. Responsibilities: Cluster assignment for sample n: $\mathbf{r}^{(n)} \in \mathbb{R}^K$ 1-of-K encoding (not observed)

Objective Mathematically

$$\min_{\{\mathbf{m}_k\},\{\mathbf{r}^{(n)}\}} J\left(\{\mathbf{m}_k\}, \{\mathbf{r}^{(n)}\}\right) = \min_{\{\mathbf{m}_k\},\{\mathbf{r}^{(n)}\}} \sum_{n=1}^N \sum_{k=1}^K r_k^{(n)} \left\|\mathbf{m}_k - \mathbf{x}^{(n)}\right\|^2$$
$$\text{where } r_k^{(n)} = \mathbb{I}\left[\mathbf{x}^{(n)} \text{ is assigned to cluster } k\right], \text{ i.e., } \mathbf{r}^{(n)} = [0, \dots, 1, \dots, 0]^\top$$

Notes Finding an optimal solution is an NP-hard problem.

1.4.1 K-means Algorithm

Initialization: Set K cluster means $\mathbf{m}_1, \dots, \mathbf{m}_K$ to random values
Repeat until convergence (until assignments do not change):
Assignment: Optimize J w.r.t. $\{\mathbf{r}\}$: Each data point $\mathbf{x}^{(n)}$ assigned to nearest center

$$\hat{k}^{(n)} = \underset{k}{argmin} ||\mathbf{m}_k - \mathbf{x}^{(n)}||^2$$

and **Responsibilities** (1-hot encoding)

$$r_k^{(n)} = \mathbb{I}[\hat{k}^{(n)} = k]$$

for $k = 1, \dots, K$
Refitting: Optimize J w.r.t. $\{\mathbf{m}\}$: Each center is set to mean of data assigned to it

$$\mathbf{m}_k = \frac{\sum_k r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}$$

Performance

1. K-means algorithm reduces the cost at each iteration.

2. The objective J is non-convex, so coordinate descent on J is not guaranteed to converge to the global minimum (solution: we could try many random starting points)

1.5 Soft K-means

One cluster may have a responsibility of .7 for a datapoint and another may have a responsibility of .3. This allows a cluster to use more information about the data in the refitting step.

1.5.1 Soft K-means Algorithm

Initialization: Set K cluster means $\mathbf{m}_1, \dots, \mathbf{m}_K$ to random values
Repeat until convergence (until assignments do not change):
Assignment: Each data point n given soft "degree of assignment" to each cluster mean k, based on responsibilities

$$r_k^{(n)} = \frac{\exp[-\beta||\mathbf{m}_k - \mathbf{x}^{(n)}||^2]}{\sum_j \exp[-\beta||\mathbf{m}_j - \mathbf{x}^{(n)}||^2}]$$

where β is a hyperparameter.

$$\implies \mathbf{r}^{(n)} = softmax(-\beta\{\mathbf{m}_k - \mathbf{x}^{(n)}||^2\}_{k=1}^{K})$$

Refitting: Model parameters, means, are adjusted to match sample means of datapoints they are responsible for:

$$\mathbf{m}_k = \frac{\sum_k r_k^{(n)} \mathbf{x}^{(n)}}{\sum_n r_k^{(n)}}$$

Remarks As $\beta \to \infty$, soft k-Means converges to hard k-Means.

1.6 A Generative View of Clustering

Imagine that the data was produced by a generative model, then adjust the model parameters using MLE, i.e., maximize the probability that it would product exactly the data we observed.
Assume a datapoint $\mathbf{x}$ is generated as follow:

1. Choose a cluster z from $\{1, \dots, K\}$ such that $p(z = k) = \pi_k$

2. Given z, sample $\mathbf{x}$ from a Gaussian distribution $\mathcal{N}(\mathbf{x}|\mu_z, I)$

Can also be written:

$$p(z = k) = \pi_k$$

$$p(\mathbf{x}|z = k) = \mathcal{N}(\mathbf{x}|\mu_k, I)$$

This defines joint distribution $p(z, \mathbf{x}) = p(z)p(\mathbf{x}|z)$ with parameters $\{\pi_k, \mu_k\}_{k=1}^{K}$, and the marginal of $\mathbf{x}$ is given by $p(\mathbf{x}) = \sum_z p(z, x)$.
$p(z = k|\mathbf{x})$ can be computed using Bayes rule.

$$\begin{aligned} p(z=k|\mathbf{x}) &= \frac{p(z=k)p(\mathbf{x}|z=k)}{p(\mathbf{x})} && (1) \\ &= \frac{p(z=k)p(\mathbf{x}|z=k)}{\sum_{j=1}^{K} p(z=j)p(\mathbf{x}|z=j)} && (2) \\ &= \frac{\pi_k \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_k, \mathbf{I}\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_j, \mathbf{I}\right)} && (3) \end{aligned}$$

This tells us the probability $\mathbf{x}$ came from the kth cluster

Choose π_k and μ_k Choose parameters to maximize likelihood of observed data. Given data $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$, choose parameters to maximize:

$$\log p(\mathcal{D}) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)})$$

Find $p(\mathbf{x})$ by marginalizing out z:

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(z=k, \mathbf{x}) = \sum_{k=1}^{K} p(z=k)p(\mathbf{x}|z=k) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, I)$$

which is an example of Gaussian Mixture Model (GMM), and π_k are known as the mixing coefficients.

Note If we allow arbitrary covariance matrices, GMMs are universal approximators of densities (if you have enough Gaussians). Even diagonal GMMs are universal approximators.

Maximum Likelihood objective

$$\log p(\mathcal{D}) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)}) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, I) \right)$$

1.7 Expectation-Maximization algorithm

1. **E-step**: Compute the posterior probabilities $r_k^{(n)} = p(z^{(n)} = k|\mathbf{x}^{(n)})$ given our current model - i.e. How much do we think a cluster is responsible for generating a datapoint.

2. **M-step**: Use the equations on the last slide to update the parameters, assuming $r_k^{(n)}$ are held fixed- change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for.

Mathematically,

1. **E-step**: Evaluate the responsibilities $r_k^{(n)}$ given current parameters

$$r_k^{(n)} = p(z^{(n)} = k|\mathbf{x}^{(n)}) = \frac{\hat{\pi}_k \mathcal{N}(\mathbf{x}^{(n)}|\hat{\mu}_k, I)}{\sum_{j=1}^{K} \hat{\pi}_j \mathcal{N}(\mathbf{x}^{(n)}|\hat{\mu}_j, I)}$$

2. **M-step**: Re-estimate the parameters given current responsibilities

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} r_k^{(n)} \mathbf{x}^{(n)}$$

$$\hat{\pi}_k = \frac{N_k}{N}$$

with

$$N_k = \sum_{n=1}^{N} r_k^{(n)}$$

3. Evaluate log likelihood and check for convergence

$$\log p(\mathcal{D}) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)}) = \sum_{n=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(n)} | \mu_k, I) \right)$$

In step 3, replace $\log p(\mathcal{D})$ with the expected complete data log-likelihood:

$$\sum_{n=1}^{N} \mathbb{E}_{p(z^{(n)}|\mathbf{x}^{(n)})}[\log p(z^{(n)}, \mathbf{x}^{(n)}] = \sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)} \left(\log \mathcal{N}(\mathbf{x}^{(n)} | \mu_k, I) + \log \pi_k \right)$$

which we maximized over parameters $\{\pi_k, \mu_k\}_k$

Notes This method can suffer from bad local optima (non-convex), as well as singularities: Arbitrarily large likelihood when a Gaussian explains a single point with variance shrinking to zero.

Remarks Suppose we sample $\mathbf{x}$ from a Gaussian distribution $\mathcal{N}(\mathbf{x}|\mu_z, \sigma^2 I)$. If $\sigma^2 \to 0$, then EM converges to soft k-Means.

2 Reinforcement Learning

Problem An agent continually interacts with the environment. How should it choose its actions so that its long-term rewards are maximized?

2.1 Formalizing Reinforcement Learning Problems

Markov Decision Process (MDP) A discounted MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$.

1. $\mathcal{S}$: State space, can be either discrete or continuous
2. $\mathcal{A} = \{a_1, \ldots, a_M\}$: Action space (finite)
3. $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{M}(\mathcal{S})$: Transition probability kernel
4. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathcal{M}(\mathbb{R})$: Immediate reward distribution
5. γ: Discount factor $(0 \leq \gamma < 1)$

The agent has a **state** $s \in \mathcal{S}$ in the environment.
At every time step $t = 0, 1, \ldots,$, the agent is at state S_t. It takes an **action** A_t, moves into a new state S_{t+1} based on the dynamics of the environment and the selected action:

$$S_{t+1} \sim \mathcal{P}(\cdot | S_t, A_t)$$

It receives some **reward**:

$$R_t \sim \mathcal{R}(\cdot | S_t, A_t)$$

Policy The action selection mechanism is described by a **policy** π: A mapping from states to actions

1. **Deterministic policy**: $A_t = \pi(S_t)$
2. **Stochastic policy**: $A_t \sim \pi(\cdot|S_t)$

Reward The goal is to find a policy π s.t. **long-term rewards** of the agent is maximized.
Different notions of the long-term reward:

1. **Cumulative / total reward**: $R_0 + R_1 + R_2 + \dots$
2. **Discounted (cumulative) reward**: $R_0 + \gamma R_1 + \gamma^2 R_2 + \dots$

The cumulation reward is problematic for continuing tasks (the agent-environment interactions does not break naturally into identifiable episodes, but goes on continually without limit) because the final time step would be $T = \infty$, and the return could easily be infinite.

Discount Factor The **discount rate** determines the present value of future rewards: a reward received k time steps in the future is worth only γ^{k-1} times what it would be worth if it were received immediately. If $\gamma < 1$, the infinite sum has a finite value as long as the reward sequence $\{R_k\}$ is bounded.

1. When γ is closer to 0, the agent prefers to obtain reward as soon as possible. (myopic)
2. When γ is closer to 1, the agent is willing to receive rewards in the farther future. (farsighted)

Transition Probability (or Dynamics) The **transition probability** describes the changes in the state of the agent when it chooses actions

$$\mathcal{P}(S_{t+1} = s' | S_t = s, A_t = a)$$

This model has **Markov property**: the future depends on the past only through the current state.

Value Function **Value function** V^π is the expected future reward, and is used to evaluate the desirability of states.
<u>State value function</u> V^π for policy π is a function defined as

$$V^\pi(s) := \mathbb{E}_\pi\left[\sum_{t\geq 0} \gamma^t R_t | S_0 = s\right]$$

where $R_t \sim \mathcal{R}(\cdot|S_t, A_t, S_{t+1})$.
It describes the expected discounted reward if the agent starts from state s and follows policy π.
Problem: we don't have direct access to the environment's dynamics, i.e. we don't know the transition probabilities, so this function is intractable.
<u>Action-value</u> function Q^π for policy π is

$$Q^\pi(s, a) := \mathbb{E}_\pi\left[\sum_{t\geq 0} \gamma^t R_t | S_0 = s, A_0 = a\right]$$

It describes the expected discounted reward if the agent starts from state s, takes action a, and afterwards follows policy π.
The goal is to find a policy π that maximizes the value function.
Optimal value function:

$$Q^*(s, a) = \sup_\pi Q^\pi(s, a)$$

Given Q^*, the optimal policy can be obtained as

$$\pi^*(s) = \underset{a}{argmax} Q^*(s, a)$$

The goal of an RL agent is to find a policy π that is close to optimal: $Q^\pi \approx Q^*$.

Notes If the state space is finite, then we can use a table to represent the value function.

2.2 Bellman Equation

Write $R_t = R(S_t, A_t)$ and $r(s,a) := \mathbb{E}_\pi[R(s,a)]$.
The value function satisfies the following recursive relationship:

$$\begin{aligned}
Q^\pi(s,a) &= \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t | S_0 = s, A_0 = a\right] && (4)\\
&= \mathbb{E}_\pi\left[R(S_0, A_0) + \sum_{t=1}^{\infty} \gamma^t R_t | S_0 = s, A_0 = a\right] && (5)\\
&= \mathbb{E}_\pi\left[R(s, a) + \sum_{t=1}^{\infty} \gamma^t R_t | S_0 = s, A_0 = a\right] && (6)\\
&= \mathbb{E}_\pi\left[R(s,a) + \gamma R(S_1, A_1) + \gamma^2 R(S_2, A_2) + \ldots | S_0 = s, A_0 = a\right] && (7)\\
&= \mathbb{E}_\pi\left[R(s,a) + \gamma\{R(S_1, \pi(S_1)) + \gamma R(S_2, \pi(S_2)) + \ldots\} | S_0 = s, A_0 = a\right] && (8)\\
&= \mathbb{E}_\pi\left[R(s,a)\right] + \gamma \mathbb{E}[\sum_{t=1}^{\infty} \gamma^t R_t | S_1 = S_1, A_1 = \pi(S_1)] && (9)\\
&= r(s,a) + \gamma Q^\pi(S_1, \pi(S_1)) && (10)\\
&= r(s,a) + \gamma \int_S Q^\pi(s', \pi(s')) \mathcal{P}(s'|s,a)\, ds' && (11)
\end{aligned}$$

Or for countable state spaces:

$$Q^\pi(s,a) = r(s,a) + \gamma \sum_S Q^\pi(s', \pi(s')) \mathcal{P}(s'|s,a)\, ds'$$

We define the **Bellman operator** $T^\theta : B(\mathcal{S} \times \mathcal{A}) \to B(\mathcal{S} \times \mathcal{A})$

$$(T^\theta Q)(s,a) := r(s,a) + \gamma \int_S Q(s', \theta(s')) \mathcal{P}(s'|s,a)\, ds'$$

Similarly, define the **Bellman optimality operator** T^*

$$(T^* Q)(s,a) := r(s,a) + \gamma \int_S \max_{a'} Q(s', a') \mathcal{P}(s'|s,a)\, ds'$$

Key observation

1. $Q^\pi \equiv T^\pi Q^\pi$
2. $Q^* \equiv T^* Q^*$

The solutions of these fixed-point equations are unique.

Proof. (for (2))

$$\begin{aligned}
(T^* Q^*)(s,a) &= r(s,a) + \gamma \int_S \max_{a'} Q^*(s', a') \mathcal{P}(s'|s,a)\, ds' && (12)\\
&= r(s,a) + \gamma \int_S Q^*(s', \pi^*(s)) \mathcal{P}(s'|s,a)\, ds' && (13)\\
&= \sup_\pi r(s,a) + \gamma \int_S Q^\pi(s', \pi(s')) \mathcal{P}(s'|s,a)\, ds' && (14)\\
&= \sup_\pi Q^\pi(s,a) && (15)\\
&= Q^*(s,a) && (16)
\end{aligned}$$

■ ■

Notes Value-based approaches try to find a $\hat{Q}$ s.t.

$$\hat{Q} \approx T^*\hat{Q}$$

Greedy policy The **greedy policy** of $\hat{Q}$ is close to the optimal policy:

$$Q^{\pi(s;\hat{Q})} \approx Q^{\pi^*} \equiv Q^*$$

where the greedy policy of $\hat{Q}$ is defined as

$$\pi(s;\hat{Q}) = \underset{a \in \mathcal{A}}{argmax}\, \hat{Q}(s,a)$$

2.3 Finding the Value Function (Policy Evaluation Problem)

Given policy π, find Q^π (or V^π).
The uniqueness of the fixed-point of the Bellman operator implies that if we find a Q s.t. $T^\pi Q = Q$, then $Q = Q^\pi$. If we find a Q such that $T^*Q = Q$, then $Q = Q^*$
Assume that $\mathcal{P}$ and $r(s,a) = \mathbb{E}[\mathcal{R}(\cdot|s,a)]$.
Case 1:
If the state-action space $\mathcal{S} \times \mathcal{A}$ is finite and small, we can solve the following Linear System of Equations over Q:

$$Q(s,a) = r(s,a) + \gamma \sum_{S} Q(s',a)\mathcal{P}(s'|s,a)\, ds' \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}$$

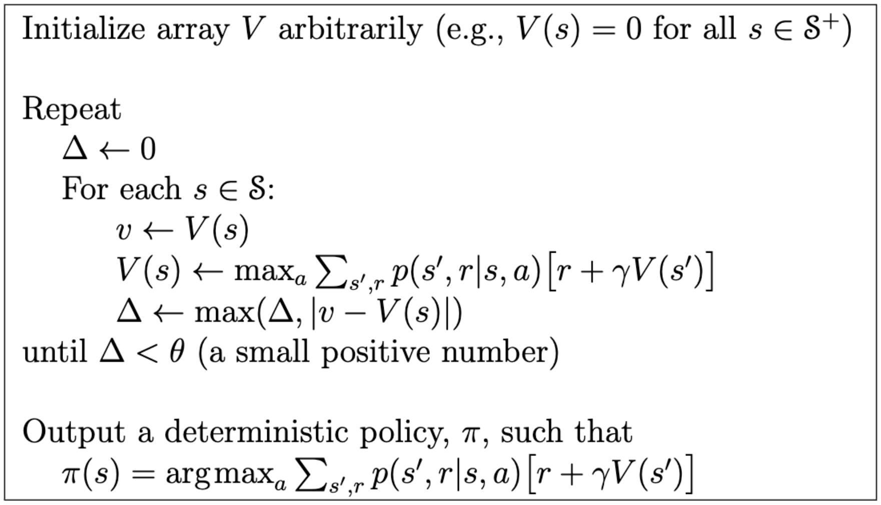
We can also solve the following Nonlinear System of Equations over Q:

$$Q(s,a) = r(s,a) + \gamma \sum_{S} \max_{a' \in \mathcal{A}} Q(s',a')\mathcal{P}(s'|s,a)\, ds' \quad \forall (s,a) \in \mathcal{S} \times \mathcal{A}$$

Case 2: Removing the assumption that the state-action space $\mathcal{S} \times \mathcal{A}$ is finite and small.

the Value Iteration method Assume that we know the model $\mathcal{P}$ and $\mathcal{R}$. Finding the optimal policy/value function when the model is known is sometimes called the **planning** problem.
We use a method called **Value Iteration**: Start from an initial function Q_1. For each $k = 1, 2, \ldots$, apply

$$Q_{k+1} \leftarrow T^* Q_k$$
$$Q_{k+1}(s,a) \leftarrow r(s,a) + \gamma \int_{\mathcal{S}} \max_{a' \in \mathcal{A}} Q_k(s',a')\mathcal{P}(s'|s,a)\, ds'$$
$$Q_{k+1}(s,a) \leftarrow r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \max_{a' \in \mathcal{A}} Q_k(s',a')\mathcal{P}(s'|s,a)$$



```
Initialize array $V$ arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi$, such that
    $\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
```

Contraction property of the Bellman operator

$$
\begin{aligned}
||T^{*}Q_1 - T^{*}Q_2||_\infty &= ||Q_2 - Q_3||_\infty \qquad (17)\\
&\leq \gamma ||Q_1 - Q_2||_\infty \qquad (18)
\end{aligned}
$$

Equivalently,

$$\sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} |(T^{*}Q_1)(s,a) - (T^{*}Q_2)(s,a)| \leq \sup_{(s,a)\in\mathcal{S}\times\mathcal{A}} |Q_1(s,a) - Q_2(s,a)|$$

Theorem By the Contraction property of the Bellman operator, the Value Iteration converges to the optimal value function.
Case 3: If $\mathcal{S} \subset \mathbb{R}^d$ or $|\mathcal{S}\times\mathcal{A}|$ is very large: Exact representation of the value function is infeasible for all $(s,a)\in\mathcal{S}\times\mathcal{A}$. The exact integration in the Bellman operator is challenging:

$$Q_{k+1}(s,a) \leftarrow r(s,a) + \gamma \int_{\mathcal{S}} \max_{a'\in\mathcal{A}} Q_k(s',a')\mathcal{P}(s'|s,a)\,ds'$$

We often do not know the dynamics $\mathcal{P}$ and the reward function $\mathcal{R}$, so we cannot calculate the Bellman operators.

Solution: Approximate/Fitted Value Iteration Suppose that we are given the following dataset

$$
\begin{aligned}
D_N &= \{(S_i, A_i, R_i, S'_i)\}_{i=1}^{N} \qquad (19)\\
(S_i, A_i) &\sim v \quad v \text{ is a distribution over } \mathcal{S}\times\mathcal{A} \qquad (20)\\
S'_i &\sim \mathcal{P}(\cdot|S_i, A_i) \qquad (21)\\
R_i &\sim \mathcal{R}(\cdot|S_i, A_i) \qquad (22)
\end{aligned}
$$

This is called **Batch RL** problem.
We can approximately perform one step of Value Iteration using these data.
Consider a function $Q : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$
We can define a random variable $t_i = R_i + \gamma \max_{a'\in\mathcal{A}} Q(S'_i, a')$
Notice that

$$
\begin{aligned}
\mathbb{E}[t_i|S_i, A_i] &= \mathbb{E}\left[R_i + \gamma \max_{a'\in\mathcal{A}} Q(S'_i, a')|S_i, A_i\right] \qquad (23)\\
&= r(S_i, A_i) + \gamma \int \max_{a'\in\mathcal{A}} Q(s', a')\mathcal{P}(s'|S_i, A_i)\,ds' \qquad (24)\\
&= (T^{*}Q)(S_i, A_i) \qquad (25)\\
&= Q_{k+1}(S_i, A_i) \qquad (26)
\end{aligned}
$$

So we can treat the problem of estimating Q_{k+1} as a regression problem with noisy data. We minimize the squared error:

$$Q_{k+1} \leftarrow \underset{Q\in\mathcal{F}}{argmin}\, \frac{1}{N}\sum_{i=1}^{N} \left| \underbrace{Q(S_i, A_i)}_{\text{current Q}} - \underbrace{(R_i + \gamma \max_{a'\in\mathcal{A}} Q_k(S'_i, a))}_{\text{true Q}} \right|^2$$

We run this procedure K-times.

2.4 Q-Learning

Online RL The agent continually interacts with the environment, updates its knowledge of the world and its policy, with the goal of achieving as much rewards as possible.

Q-Learning with ϵ-Greedy Policy Update the estimate of the action-value function Q online and only based on (S_t, A_t, R_t, S_{t+1}), using the policy π_t to explore the environment, such that Q converges to the optimal value function Q^*.

- learning rate α, exploration parameter ϵ
- Initialize $Q(s, a)$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$

The agent starts at state S_0. For time step $t = 0, 1, \ldots$, choose A_t according to the ϵ-greedy policy

$$\pi_\epsilon(S_t; Q) = A_t \leftarrow \begin{cases} \underset{a \in \mathcal{A}}{argmax}\, Q(S_t, a) & \text{with probability } 1 - \epsilon \\ \text{Uniformly random action in } \mathcal{A} & \text{with probability } \epsilon \end{cases}$$

Take action A_t in the environment.
The state of the agent changes from S_t to $S_{t+1} \sim \mathcal{P}(\cdot | S_t, A_t)$.
Observe S_{t+1} and R_t.
Update the action-value function at state-action (S_t, A_t):

$$Q(S_t, A_t) \leftarrow \; Q(S_t, A_t) + \alpha \left[R_t + \gamma \max_{a' \in \mathcal{A}} Q(S_{t+1}, a') - Q(S_t, A_t) \right]$$

Remarks The ϵ-greedy policy ensures that most of the time the agent exploits its incomplete knowledge of the world by choose the best action, but occasionally it explores other actions.
One can show that under certain conditions, Q-Learning indeed converges to the optimal action-value function Q^* for finite state-action spaces.

2.5 Approaches to RL

- Discussed are **value-based methods**: Optimize the value function
- **policy search methods**: Optimize the policy
- **Model-based RL**: Estimate the true, but unknown, model of environment $\mathcal{P}$ by an estimate $\hat{\mathcal{P}}$, and use $\hat{\mathcal{P}}$ to plan
- hybrid methods