Latex Book

Rahul Bhadani
rahulbhadani@email.arizona.edu

May 31, 2019

Contents

1 First Chapter 1

1.1 Some important questions 1

1.2 Some metrics 1

2 Second chapter 2

2.1 Terminology 2

3 Writing code in latex 3

1

First Chapter

This is first chapter

1.1 Some important questions

It is a section

1.2 Some metrics

Another section

2

Second chapter

Deep neural nets are learning intelligent behavior in a complex dynamic environment but there are cons: we have to train them. At the same time, trained agents can't do better than the training set. Solution: Use a policy-based approach and reward the agent whenever the agents do something good while penalizing them when agents do something undesirable. This is the principle behind the **Reinforcement Learning (RL)**. A general idea is depicted in Figure 2.1.

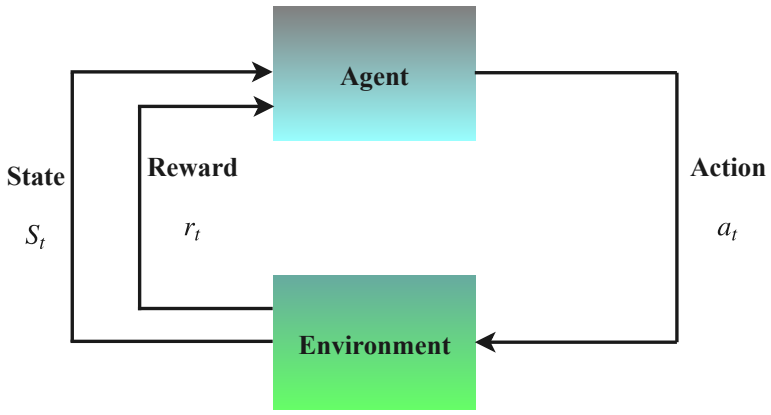

Figure 2.1: Reinforcement learning.

Policy network: network in reinforcement learning to give output.

- Start with completely random network
- Produces random o/p.
- Feedback to (say) game engine
- loop continues
- Use scoreboard for reward/penalty
- Goal is to optimize the policy to receive as much reward as possible.

2.1 Terminology

3

Writing code in latex

Install necessary packages:

```
sudo apt-get install golang python3-dev python-dev libcupti-dev libjpeg-turbo8-dev \
make tmux htop chromium-browser git cmake zlib1g-dev libjpeg-dev  \
xvfb xorg-dev python-opengl libboost-all-dev libsdl2-dev swig
```

```
% (C) RAHUL BHADANI
% Problem 5

% Kennedy Receiver - BPSK

N = 0.0000001:0.0001:10;
P = zeros(size(N));
b_root = zeros(size(N));

for j = 1:length(N)
    a = sqrt(N(j));

    %Newton Raphson Method, %Find the roots of dPE/dbeta = 0
    bn = 0.5; %Initial Guess of the roopts of dPE/dbeta
    %We are doing 10000 iterations
    for i  = 1:10000
        b_next = bn - (f(bn,a)./fdash(bn,a));
        bn = b_next;
    end
    b_root(j) = b_next;
    P(j) = PE(b_next, a);

end

fig= figure;
fig.Position = [652   490   593   479];
PEDD = 0.5*exp(-4.*N);

P_Dolinar = 0.5.*(1- sqrt(1 - exp(-4.*N)));
subplot(2,1,1);
semilogx(N, PEDD,'r','LineWidth',1);
hold on;
semilogx(N, P,'b--','LineWidth',1);
semilogx(N, P_Dolinar,'k-','LineWidth',1);
```

```
xlabel('N','Interpreter','latex');
ylabel('$P_e$ Average probability of the error','Interpreter','latex');
set(gca,'FontSize',16);
set(gca,'FontName','Times');
title('$P(error)$ for the Kennedy receiver','Interpreter','latex');
grid on;
grid minor;
legend({'$\textbf{Direct Detection-Equal Prior}$',...
    '$\textbf{Kennedy Receiver-Equal Prior}$',...
    '$\textbf{Dolinar Receiver}$'...
    },'Interpreter','latex');

subplot(2,1,2);
plot(N, b_root, 'g--','LineWidth',2);
xlabel('N','Interpreter','latex');
ylabel('$\beta$','Interpreter','latex');
set(gca,'FontSize',16);
set(gca,'FontName','Times');
title('$\textbf{Optimal Beta as a function of N}$','Interpreter','latex');
grid on;
grid minor;

% savefig(fig, 'figures/rahulbhadani_OPTI_595B_Q5.fig');
% saveas(gcf,'figures/rahulbhadani_OPTI_595B_Q5.pdf');

function PRET = PE(b, a)
    PRET = 0.5*(1 - exp(-b.*b)) + 0.5*exp(-( 2.*a + b).^2);
end

function fRET = f(b, a)
    fRET = b*exp(-b*b) - (b+2.*a)*exp(-(b+2.*a)^2);
end

function fdashRET = fdash(b, a)
    fdashRET = 2*((b+2.*a)^2)*exp(-(b+2.*a)^2) - exp(-(b+2.*a)^2) - 2*b*b*exp(-b*b) + exp(-b*b);
end
```