# PCA to represent state spaces in reinforcement learning domain

**Oleg Dats**
*Student at UKU*
*Lviv, Ukraine*
*2019*

## Abstract

Reinforcement learning (RL) is an area of machine learning concerned with how software agents take action in environment to maximize cumulative reward. The simplest example can be a game of tetris. Where the state is a screen, discrete actions represented on joystick and cumulative reward is a total sum of a points. Our task is to find optimal policy: decide on action depending on state.

RL algorithms need to deal with the exponential growth of states and actions. This problem is known as the curse of dimensionality (Richard E. Bellman when considering problems in dynamic optimization). Our goal is to project agent's representation of a state onto space with a less number of dimensions. We will use this new state for training the agent. Canonical CartPole was chosen as test environment.

Principal Component Analysis is a well known approach in dimensionality reduction. The main hypothesis is to verify whether PCA will speedup training process. Second hypothesis is whether losing information during transformation prevent searching for the optimal policy to converge. The last hypothesis is show that PCA can be used to explain states (speed, position,...). Decide on what is important for agent to make the right decisions.

## 1. Introduction

Curse of dimensionality is a main topic in robotics and AI in general. Imagine trading agent that decides: sell or buy some specific stock. There are infinite number of factors that can influence decision: thousands of other stocks, weather conditions, political situation… State for the agent is continuous and huge. This situation reflects the curse of dimensionality described by Bellman. When the number of dimensions increases, the volume of the space increases so fast that the available data become sparse. It creates 2 main problems: the amount of data needed to make statistically significant decision increases exponentially and detecting areas where objects form groups with similar properties becomes impossible.

Our approach is to use PCA to reduce the dimensionality of the data. When you have that many dimensions it happens that some of them are correlated. For example S&P 500 stock market index  can describe the market situation more / less the same as all available stocks on the market. It means rather than having 1000+ features we can have only 1.

We use PCA to find transformation from high dimensions to low dimensional state space representation. During training and evaluation and all it's life the agent will live only in low-dimensional space. This leads to a critical trade-off and the main theme of this paper. Can the agent find optimal policy based on less representative state. Due to projection on lower dimensions we always lose some data. However, I expect to get faster converging to suboptimal policy.

## 2. Related work

PCA was invented in 1901 by Karl Pearson (Pearson, 1901). It is well known and used for a long time for dimensionality reduction. Previous work in dimensionality reduction focuses on reducing the space for regression, classification or function approximation.

In statistics, principal component regression (PCR) is a regression analysis technique that is based on principal component analysis (PCA). In PCR, instead of regressing the dependent variable on the explanatory variables directly, the principal components of the explanatory variables are used as regressors. Major use of PCR lies in overcoming the multicollinearity problem which arises when two or more of the explanatory variables are close to being collinear. In the same manner the PCA can be used in classification tasks and logistic regression.

Rather than using PCA for feature extraction in large data sets, we use PCA to reduce the dimensionality of the state space during learning.

William Curran et al. (2005) showed that it is possible to use PCA to train the agent. In my paper I followed their approach to present the information. Unlike the authors I will show that there are sceneries where training agent in low dimensional state space can overcome training in full dimensional state.

## 3. PCA and RL

First step is to find transformation between the high-dimensional state space and a lower-dimensional space. Agent executes random actions to collect the list of trajectories. Then we use PCA to reduce dimension and minimize losing the information. PCA provides us with a new set of dimensions, the principal components (PC). They are ordered: the first PC is the dimension with the largest variance. In addition, each PC is orthogonal to the preceding one. (Jonathon Shlens, 2014)

$$T = X*W$$

X - the list of trajectories
W - transformation matrix, n by n, columns are eigenvectors of covariance matrix (XT*X) and n is the number of principal components
T - low dimensional representation

In our paper we will run tests for all possible values of p=[1,2,3,4]. For each step we project state on to lower-dimensional manifold and train agent in this new state representation. The algorithm learns optimal policy in lower-dimensional space rather than in full dimensions.

The main benefit of learning in lower-dimensional representation is speed. I expect to get much faster convergence. The trade of is losing some important data. Throwing too much variance can stop agent of learning.

## 4. CartPole problem

CartPole is a canonical problem in RL. A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center. (Greg Brockman et al. (2016)

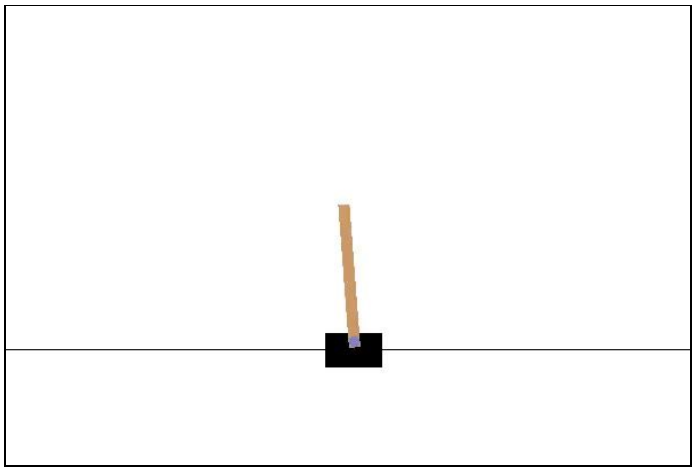

*Figure 1: a screenshot of CartPole*

The state space is represented by 4 numbers:

| Num | Observation | Min | Max |
|---|---|---|---|
| 0 | Cart Position | -2.4 | 2.4 |
| 1 | Cart Velocity | -Inf | Inf |
| 2 | Pole Angle | ~ -41.8° | ~ 41.8° |
| 3 | Pole Velocity At Tip | -Inf | Inf |

*Figure 2: state space*

CartPole was chosen because of it is simplicity. The main research question is PCA so we can use simple random search to find optimal policy and concentrate on the main topic.

```
def train():
    env = gym.make('CartPole-v0')

    counter = 0
    bestparams = None
    bestreward = 0
    for _ in range(2000):
        counter += 1
        parameters = np.random.rand(4) * 2 - 1

        reward = 0
        # run few times to make sure it is good parameters
        for _ in range(5):
            run = run_episode(env, parameters)
            reward += run
        reward = reward / 5

        if reward > bestreward:
            bestreward = reward
            bestparams = parameters
            # stop if we reached maximum
            if reward == 200:
                print('super win')
                break

    return counter
```

*Figure 3: simple algorithm to find optimal policy.*

```
action = 0 if np.matmul(parameters, observation) < 0 else 1
```

*Figure 4: policy function.*

```
pca_observation = pca.transform([observation])[0]
action = 0 if np.matmul(parameters, pca_observation) < 0 else 1
```

*Figure 5: updated policy function to project state in to lower dimension and retrain agent*

## 5. Results

To decide on the number of principal components we need to plot the cumulative variance explained by components ordered by variance. From the plot we see that the first 3 components explain 99% of data.

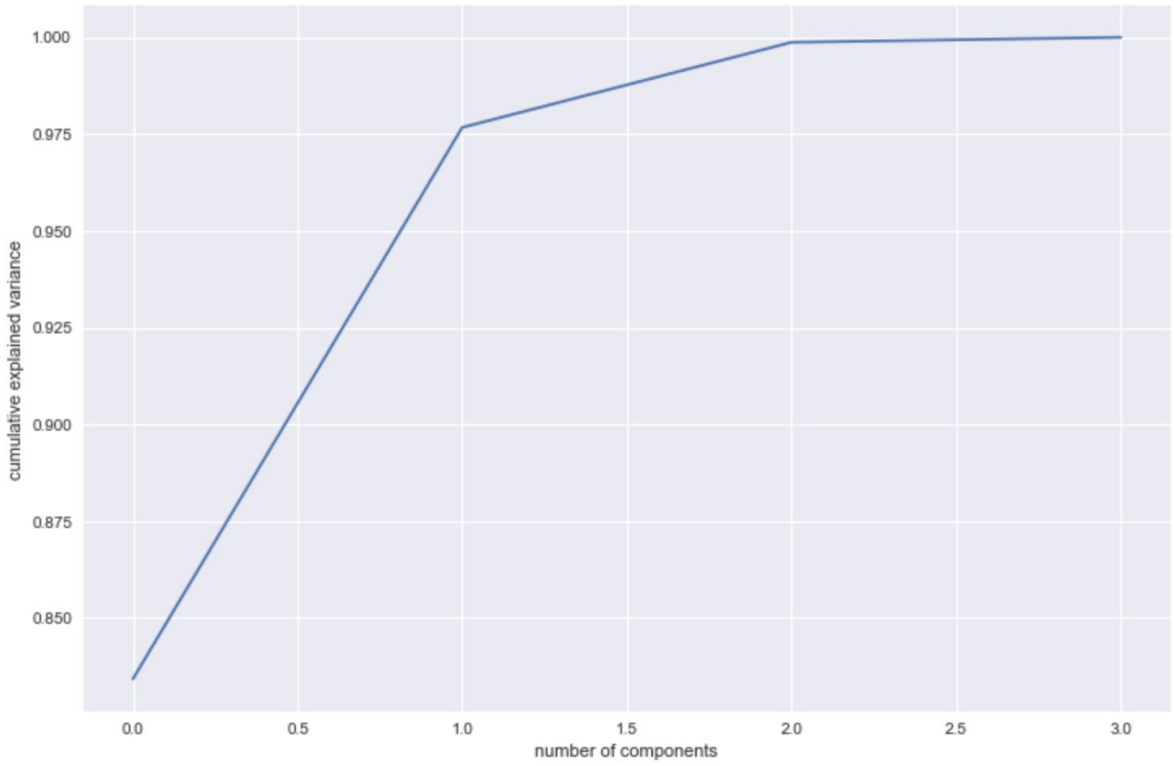


*Figure 6: cumulative explained variance by principal components*

To make sure that 3 components should be enough we should run empirical tests. In our case it is training agent in real environment and check whether it is possible to find sub optimal policy:

| State space representation | Number of episodes to converge |
|---|---|
| Full state | 25 episodes |
| 1 Principal Component | Algorithm can not converge |
| 2 Principal Components | Algorithm can not converge |
| 3 Principal Components | 18 episodes |
| 4 Principal Components | 25 episodes |

*Figure 7: average number of episodes required to find optimal policy and reach 200 points.*

The full state is represented in 4 dimensions. In average the algorithm needs 25 attempts to guess the right policy. As we expected, when we have 4 PC the result should be the same.

The algorithm can not converge when the state is represented in 1 or 2 dimensions. It means we lose important information. That is crucial point in my research. Trade off should be always taken into account. No free lunch theorem for optimization (David H. Wolpert, 1997)

The best results we get when project high dimensional state onto low dimensional state described by 3 dimensional manifold. First 3 Principal components preserve all important information and generalize the state.

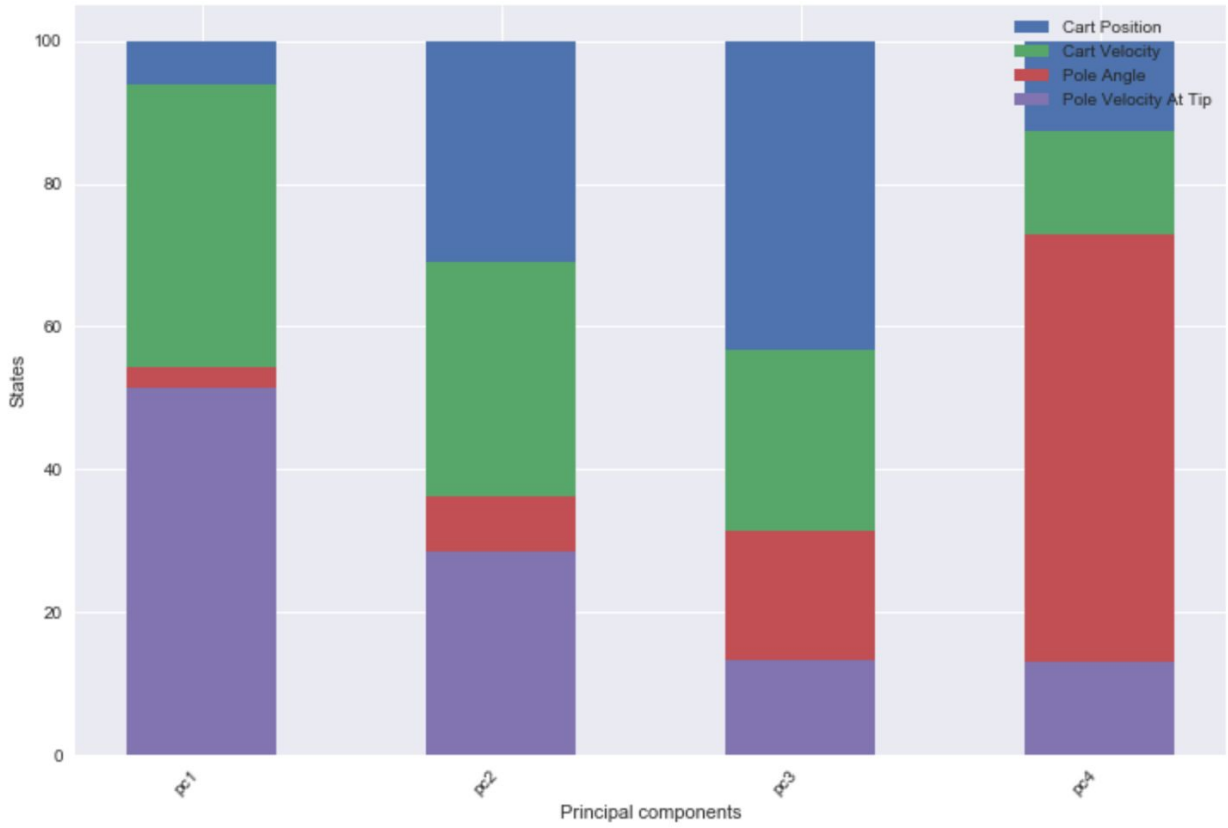


*Figure 8: Correlation between an original variable and Principal Component*

Since we could not find the optimal policy with only 2 Principal Components we can make inference that state space approximation needs more information. Based on chart we see that first 2 components mostly include: velocity of cart and velocity of pole. Adding third components with the information about pole angle dramatically increases performance. These data confirms our hypothesis that PCA explains data and is useful to identify important aspects of the state.

Correlations between variable and factors (principal component) are called loadings.

$$r = v_{ij} \cdot \mathrm{std}(Y_j)/\mathrm{std}(X_i) = v_{ij} \cdot \sqrt{e_j}/\mathrm{std}(X_i),$$

*Figure 9: correlation coefficient between variable X and principal component Y*

## 6. Discussion and future work

The main benefit of using PCA it is simplicity in understanding and computation. Usually it is just matrix multiplication. Our findings confirm possibility to find optimal policy for simple environments by dimensionality reduction of the state and successful training the agent only in low dimensional manifold. We showed that for some environments applying PCA will speedup converging the search of optimal policy.

There are 2 main cons:

1) Algorithm thinks that all states are important. Nevertheless, usually we can drop some not important dimensions.
2) We always converge to suboptimal policy. Having infinite time and full state the algorithm will always converge to better policy.

The topic should be interesting for researchers in reinforcement learning field. They can apply our findings in the next way: chose the problem, use PCA to transform the state in low dimensional representation and use dynamic programming to find optimal policy. The next step in research could be applying transfer learning of agent's knowledge from low dimensional representation to improve training in full dimensional state.

The other promising topic is to apply PCA for the environments represented by pixels and compare this approach with CNN. The eigenfaces is a good starting point. (Turk and Pentland, 1991)

## Environment

Open AI gym, v0.9.6
CartPole-v0
Working code url: https://github.com/odats/uku-la-final-project/blob/master/PCA%20simple.ipynb

## References

Pearson, K. 1901. On lines and planes of closest fit to systems of points in space. Philosophical Magazine 2:559-572.
URL http://pbil.univ-lyon1.fr/R/pearson1901.pdf

Jonathon Shlens. A tutorial on principal component analysis, 2014.
URL https://arxiv.org/abs/1404.1100

David H. Wolpert and William G. Macready. No Free Lunch Theorems for Optimization, 1997
URL https://ti.arc.nasa.gov/m/profile/dhw/papers/78.pdf

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, Wojciech Zaremba. OpenAI Gym, 2016
URL https://arxiv.org/abs/1606.01540

William Curran, Tim Brys, Matthew Taylor, William Smart. Using PCA to Efficiently Represent State Spaces, 2005.
URL https://arxiv.org/abs/1505.00322

M.A. Turk and A.P. Pentland. Face recognition using eigenfaces. In Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on, pages 586– 591, Jun 1991. doi: 10.1109/CVPR.1991.139758.
URL https://www.cs.ucsb.edu/~mturk/Papers/jcn.pdf