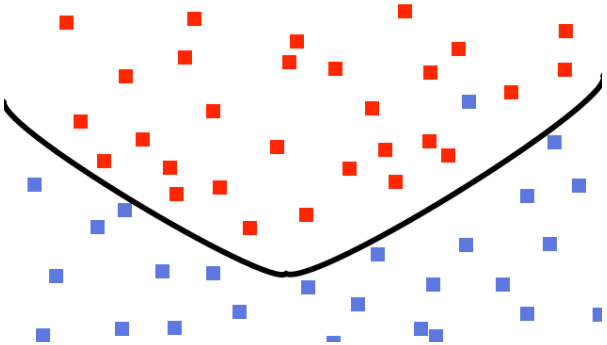
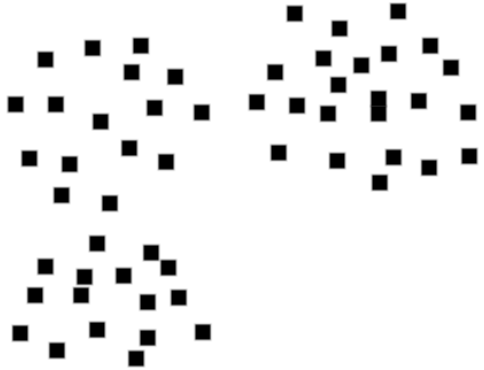
UNSUPERVISED LEARNING

3 Modes of Learning

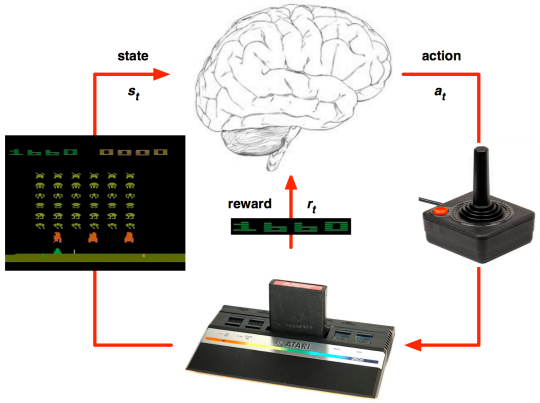


Supervised Learning

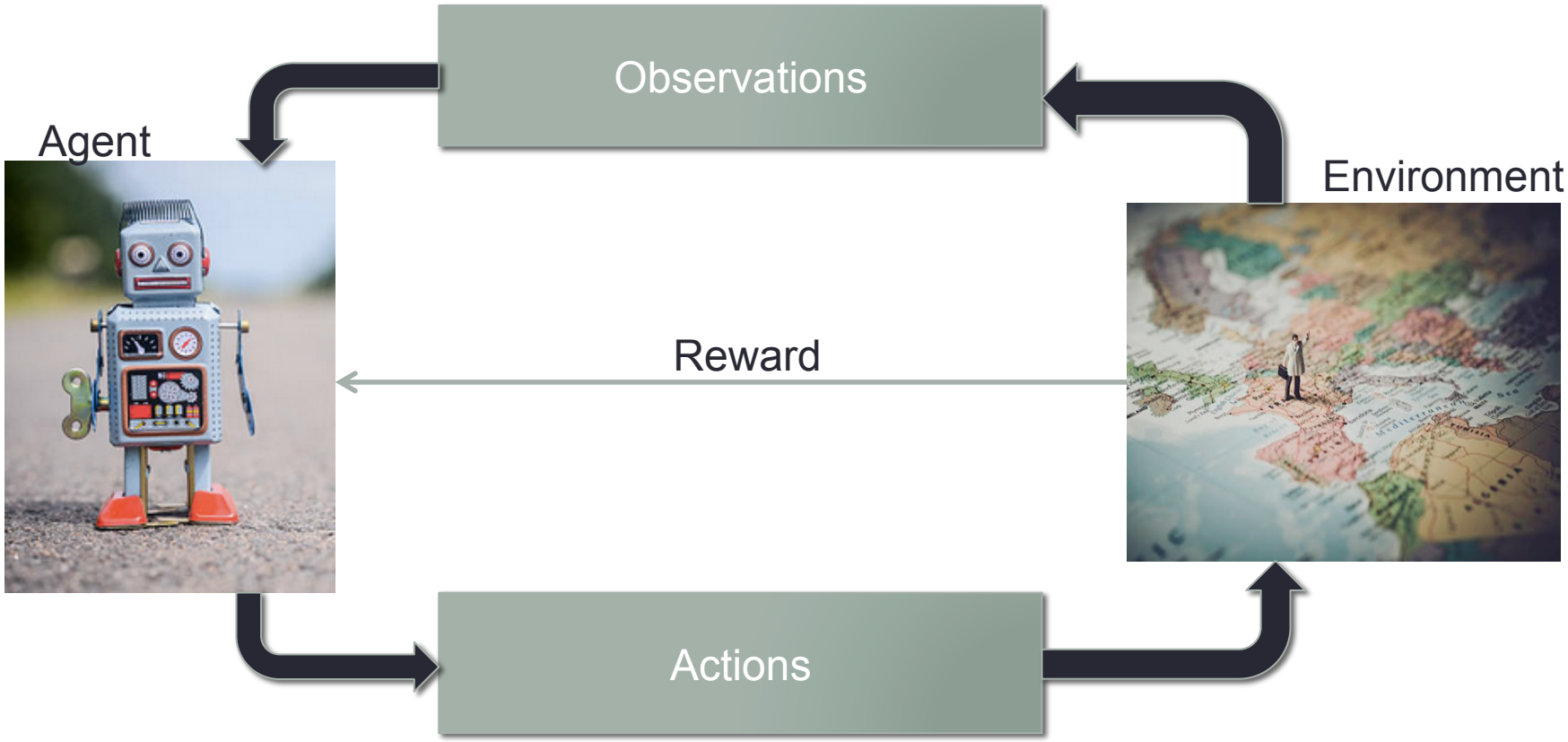
Reinforcement Learning



Unsupervised Learning



Reinforcement learning framework

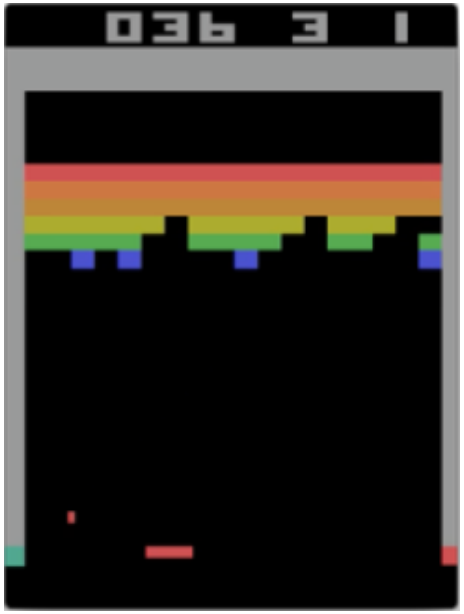


Learning through trial and error

Rewards-based learning

- Maximize the rewards

$R_t = \Delta\text{distance}$



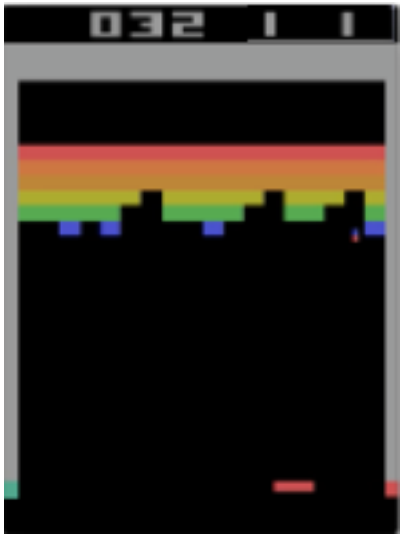
$R_t = \Delta\text{score}$

$$R_T = \begin{cases} 1 & \text{, win} \\ -1 & \text{, lose} \end{cases}$$

Credit assignment problem

- An action can have consequences further away in time
- Some movements might not have any effect on the outcome

10 time steps later



$r_t = 0$

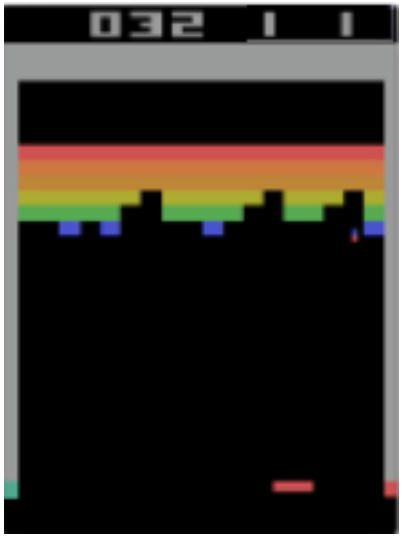
$r_{t+10} = 1$

Cumulative rewards (Return)

- Maximize the expected cumulative rewards with discounting

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots = \sum_{t'=t}^{\infty} \gamma^{t'-t} r_t$$

$r_t = 0$



$r_{t+10} = 1$

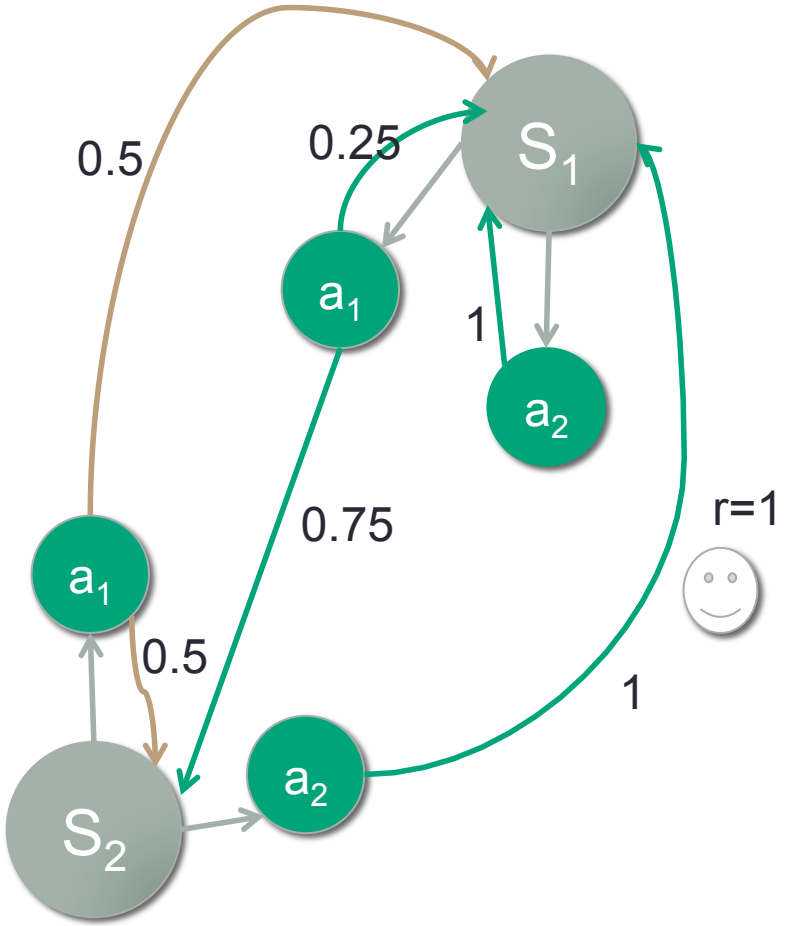
$r_{t+34} = -1$

Markov Decision Process (MDP)

- **S,A,P,R**,γ
- **S** – Set of states
- **A** – Set of actions
- **P** – Transition between states given an action

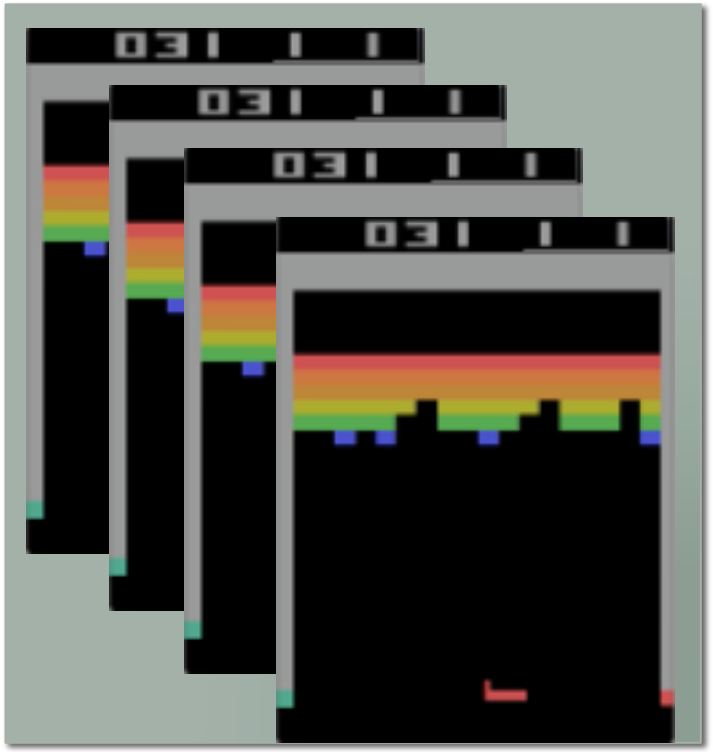
$$P^{a}_{s,s'} = \text{Prob}[s_{t+1}=s' \mid s_t=s, a_t=a]$$

- **R** – Rewards associated with actions and states
- γ – Discount factor



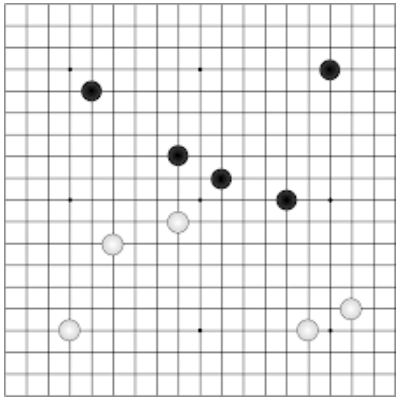
Markov State

- Markov property: All information from past is captured in the current state



A state

For Go, a board position
For simple video games, stack multiple frames



Policy

- A strategy to perform an action according to a particular state

$$\pi(s) = a$$

- A policy defines the behavior of the agent
- In MDP, we want to learn the best policy for the task
  - How?
    - Define some loss/reward function and minimize/maximize.

Action-Value Function (Q Function)

- Expected total rewards starting from state s taking action a then follows a given policy π

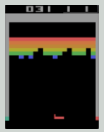
$$Q_\pi(s,a) = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \mid s_t = s, a_t = a]$$
$$= E[r_t + \gamma Q_\pi(s',a') \mid s_t = s, a_t = a]$$
$$= R_s^a + \gamma \sum_{s'} P_{s,s'}^a \max_{a' \in A} Q_\pi(s',a')$$

- Note, Q is a function of s, a, and
- Q* determines the optimal policy π^*

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

Q function and policy example

- For a policy, Q function gives the expected rewards

	Left	Right	Policy
	134	123	Left
	579	0	Left

- We want to find the best policy by maximizing the Q function.
- But we don’t know Q. Need to estimate this table.
  - How? – Bellman Equation (iteratively update Q)

Q-learning

- Q-learning is used to learn Q* by sampling the states s' and rewards r through events
  - *Starts with an original belief of Q values*
  - *Takes random actions to explore the possibilities*
  - *Update your belief of Q depending on the rewards from the environment*
- No need to explicitly learn the transition probability or the expected rewards
  - **"Learns by trial and error"**

Temporal difference Q-learning

One method to learn Q

Update Q by the reward received from taking action a in state s.

State s' is the state after taking action a.

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

Recall

$$Q(s,a) = R_s^a + \gamma \sum_{s'} P_{s,s'}^a \max_{a' \in A} Q(s',a')$$

R (the expected rewards) becomes r (the sampled reward)
The transition probability, P, is dropped because we sampled one outcome. This is a proxy for the sampled Q*

Temporal difference Q-learning

Update Q by the reward received from taking action a in state s.

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

Mixing coefficient
(learning rate)

Our original belief of Q*

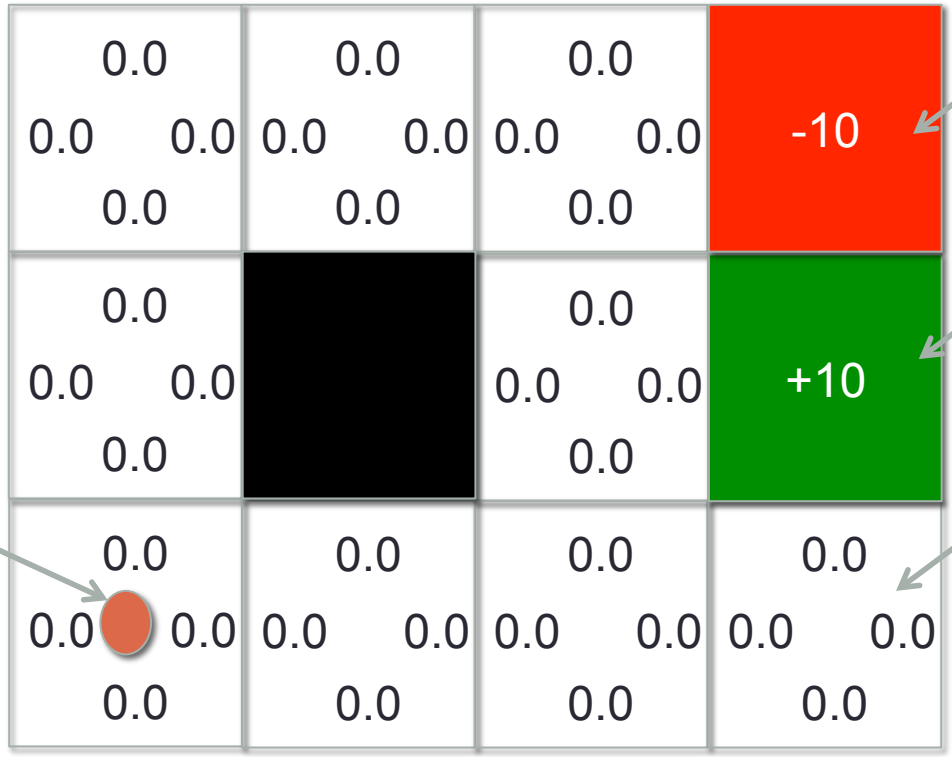
What the the simulation says our Q* should be

Toy example

The states are the different positions in the room
4 actions: up, down, left, right

$\gamma = 1.0$

$\alpha = 0.5$



Bad exit

Good exit

Q value for each direction

Agent

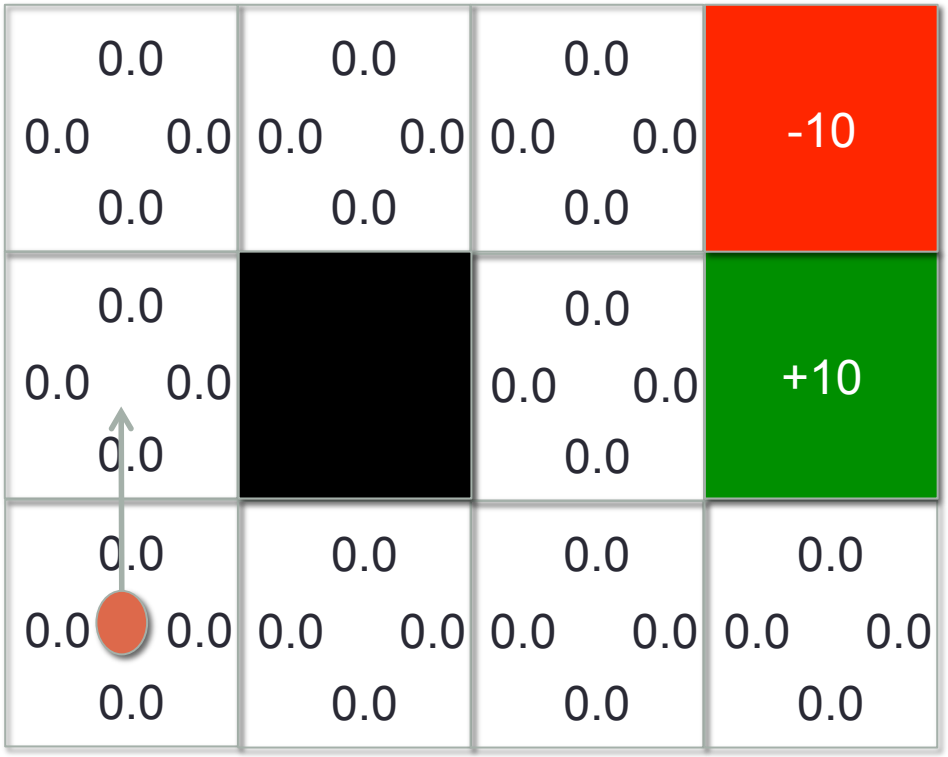
A reward of -1 for every move (penalty for time wasted)

Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$



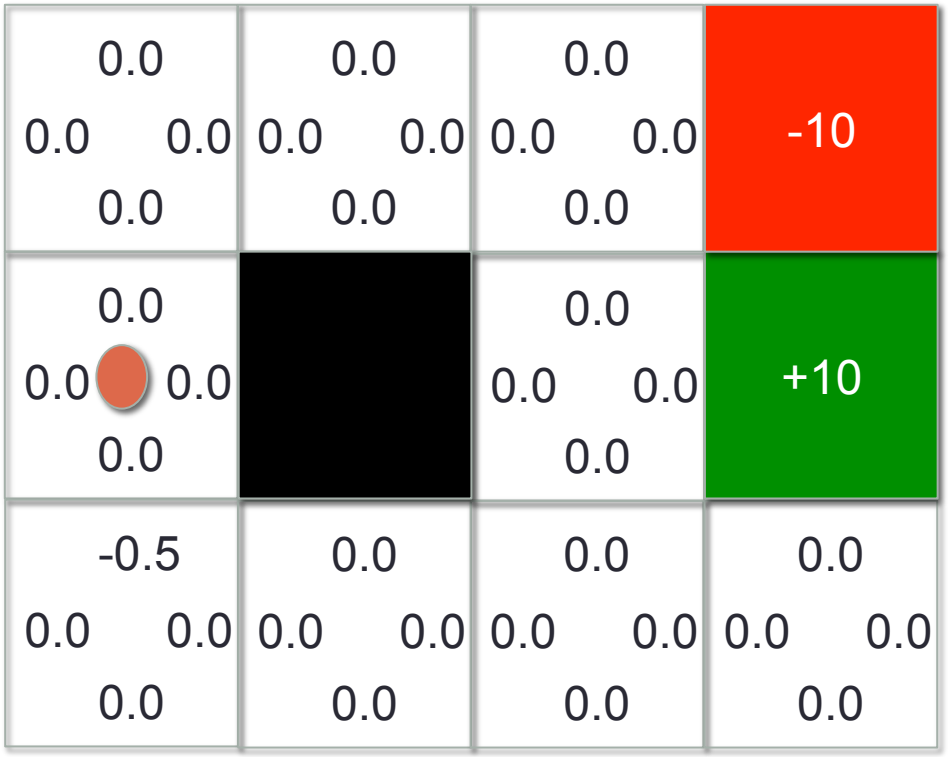
(1-0.5)0+0.5[-1 + 1* 0] = -0.5

Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$

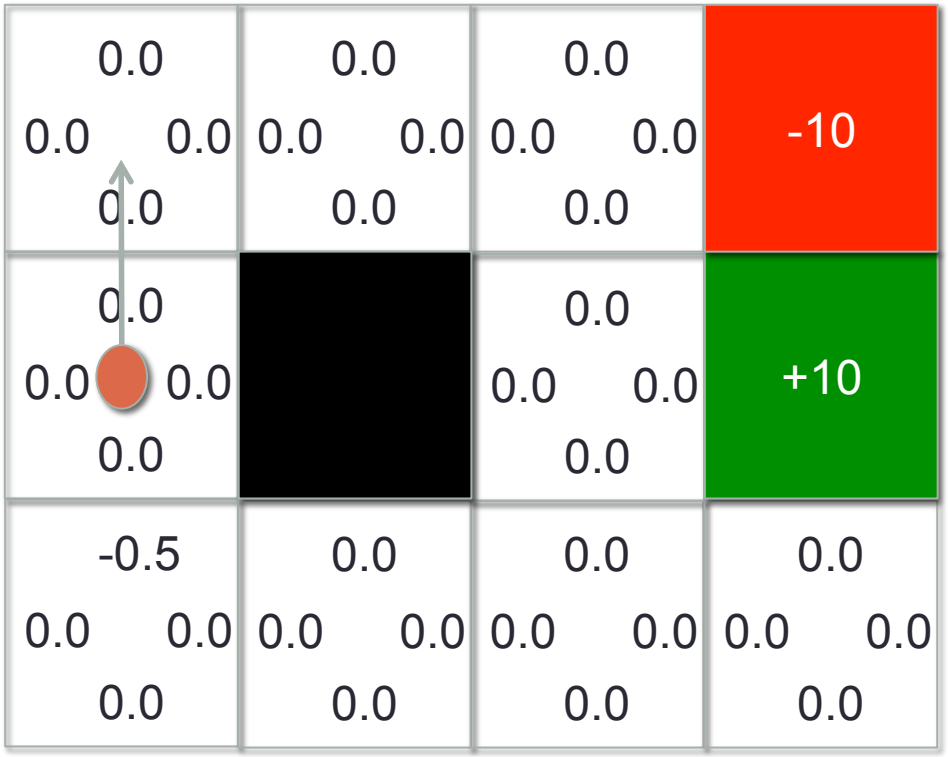


Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$



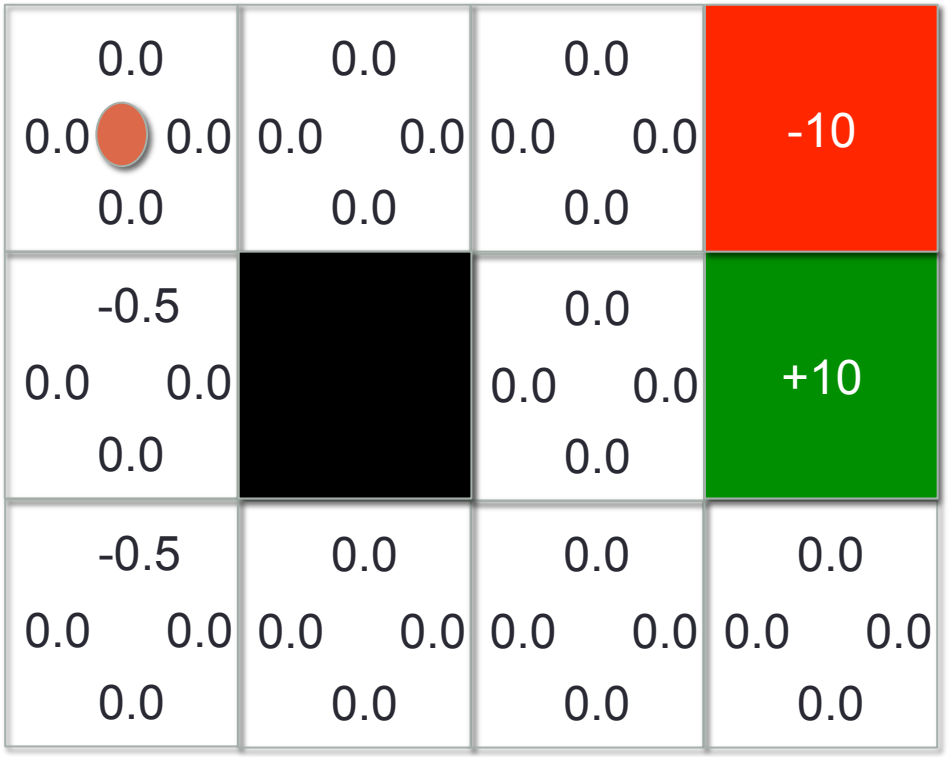
(1-0.5)0+0.5[-1 + 1* 0] = -0.5

Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$

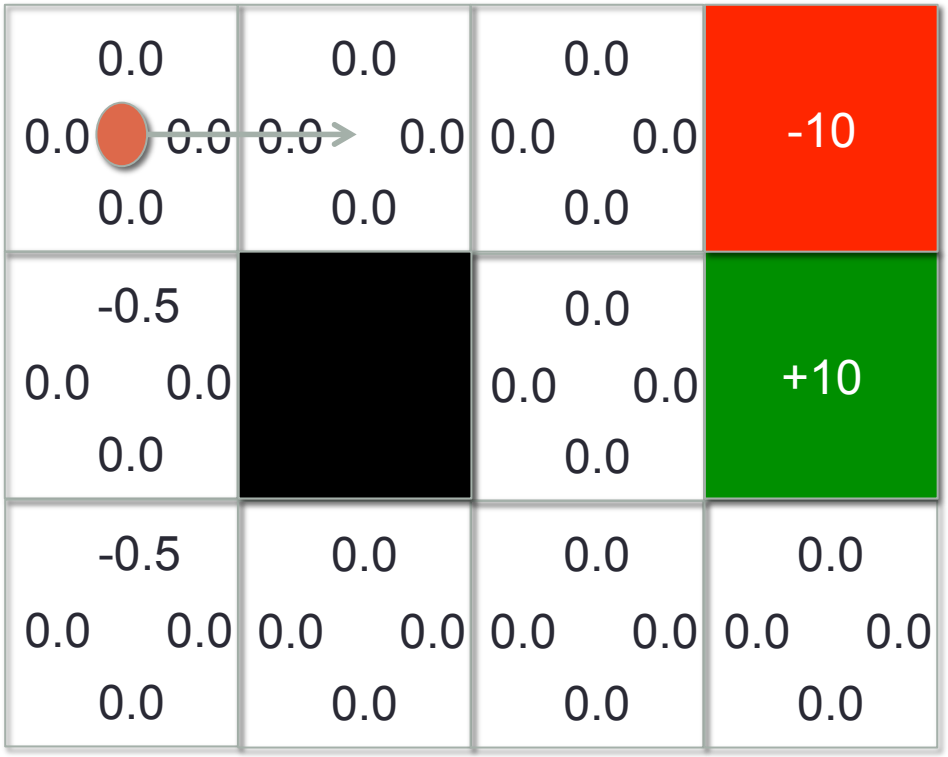


Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$



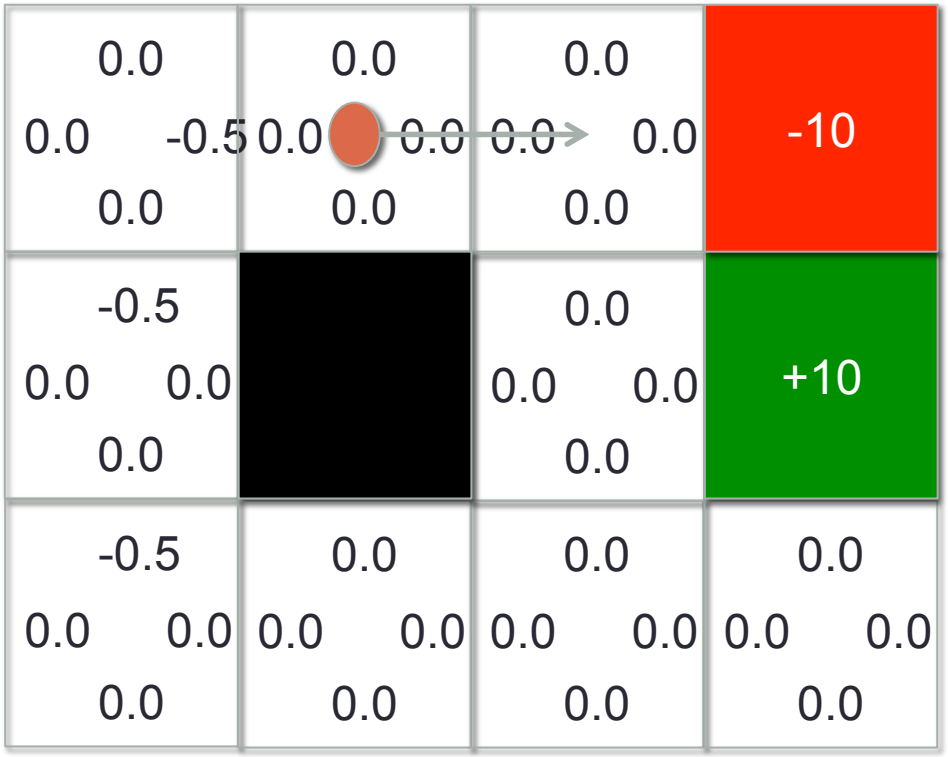
(1-0.5)0+0.5[-1 + 1* 0] = -0.5

Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$



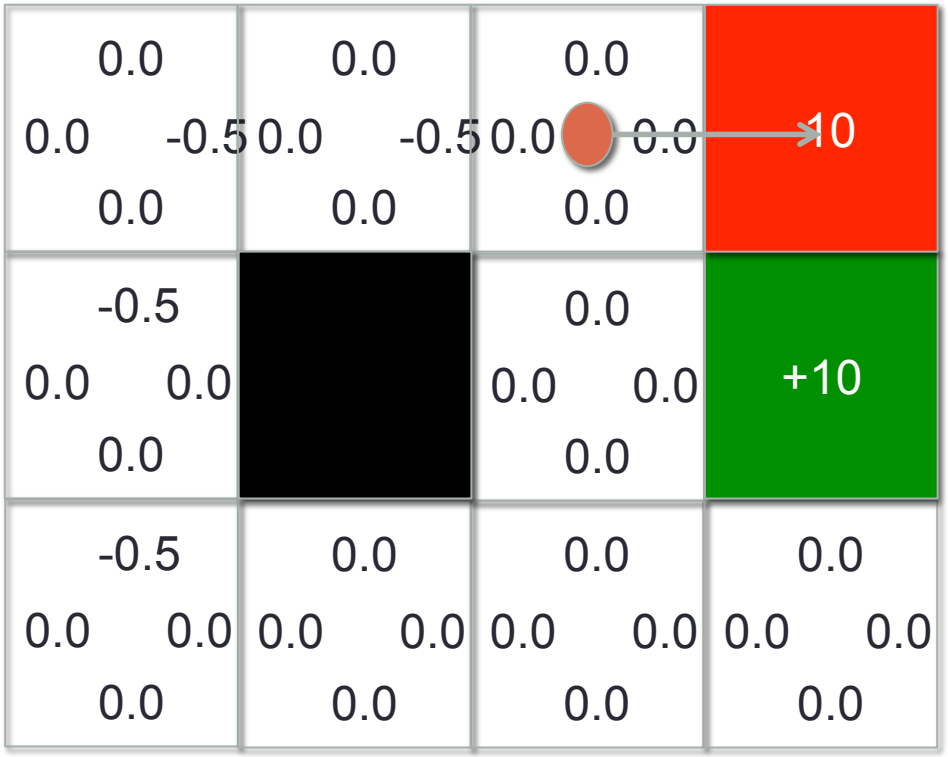
(1-0.5)0+0.5[-1 + 1* 0] = -0.5

Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$



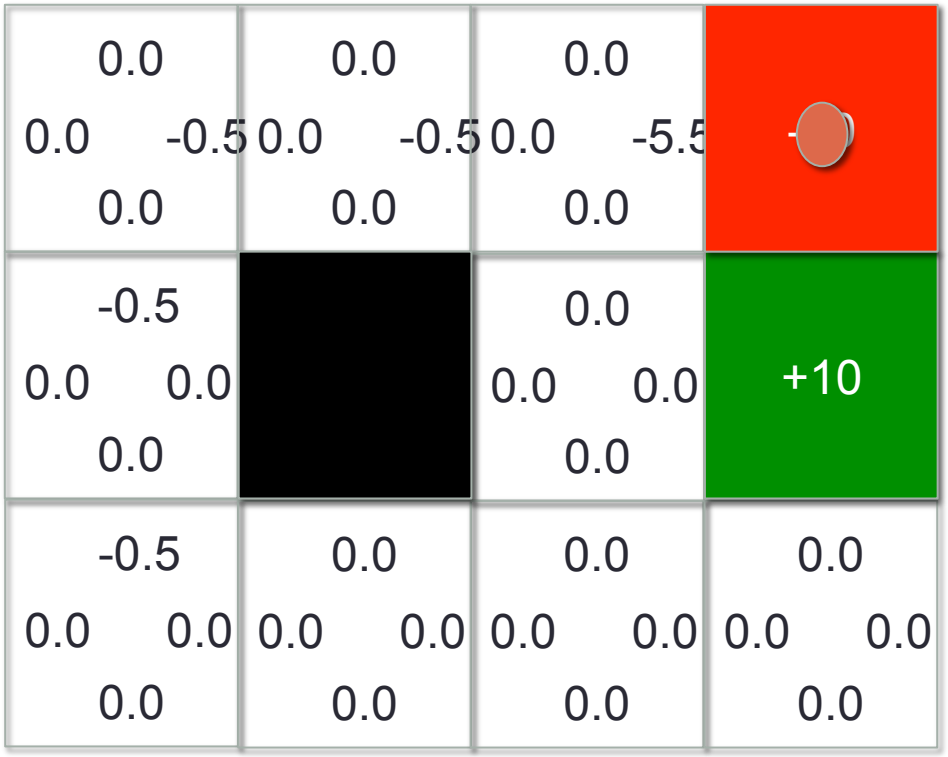
(1-0.5)0+0.5[-1 + 1* -10] = -5.5

Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$

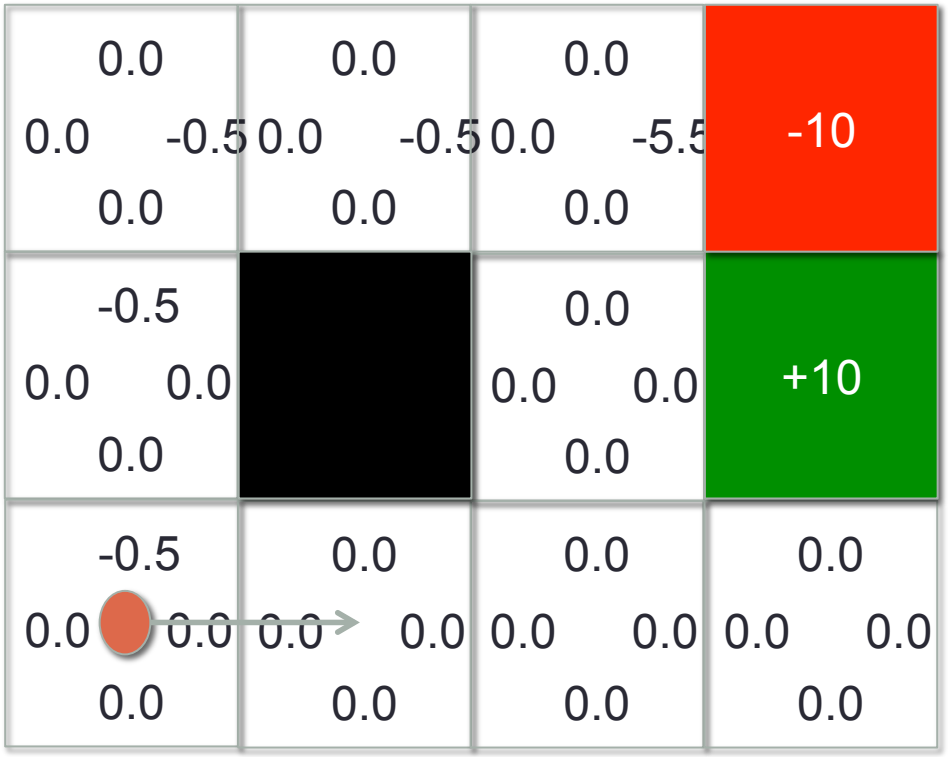


Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$

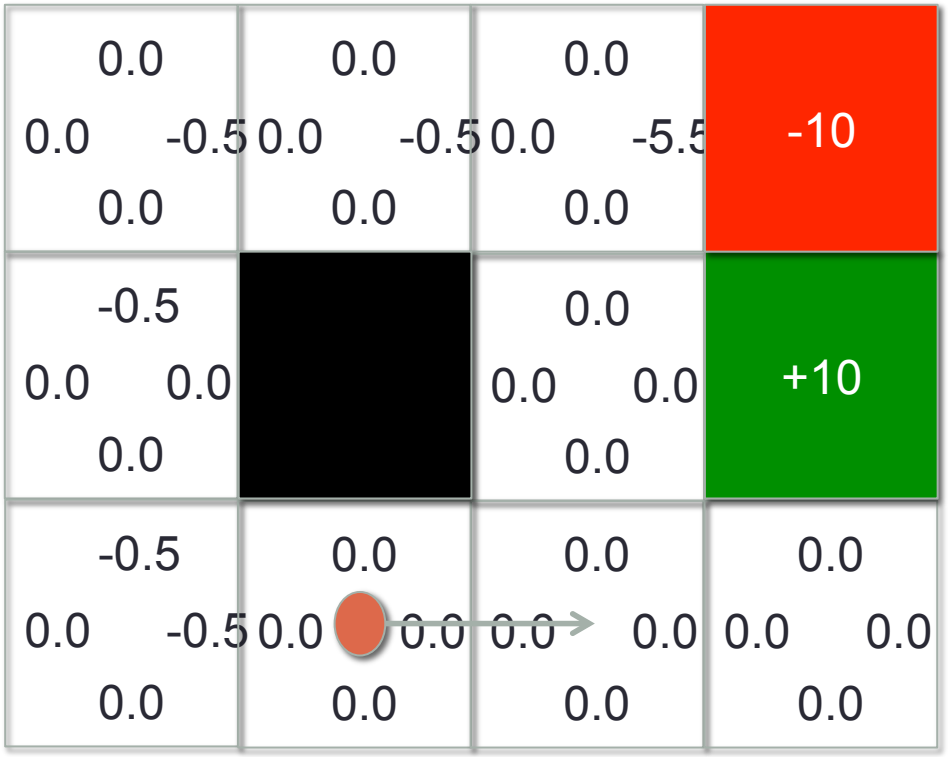


Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$

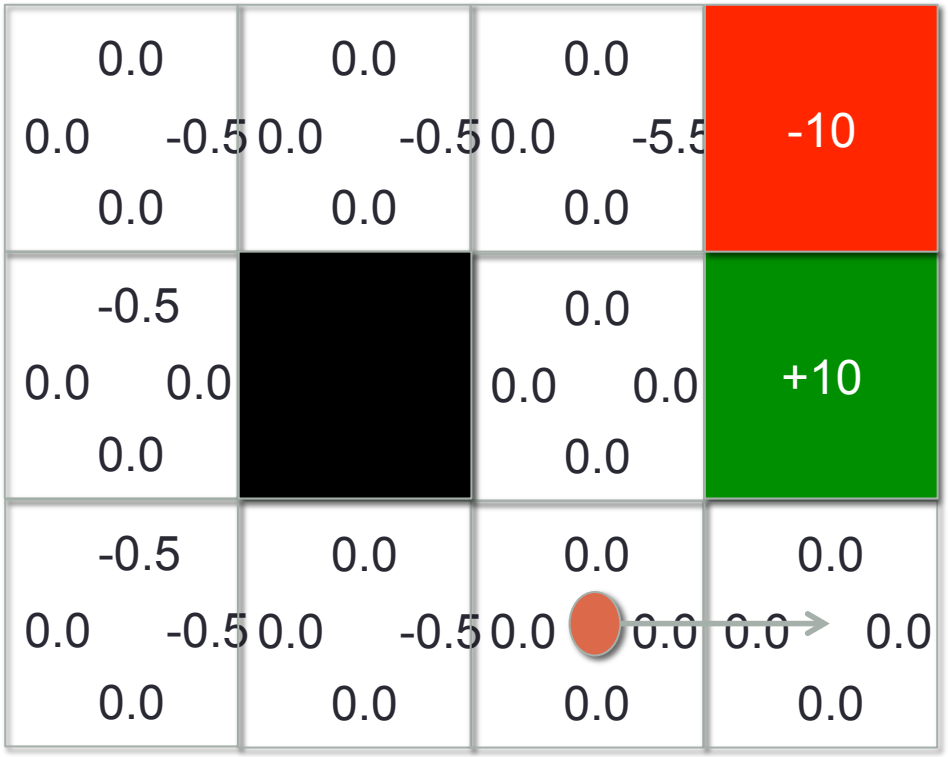


Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$

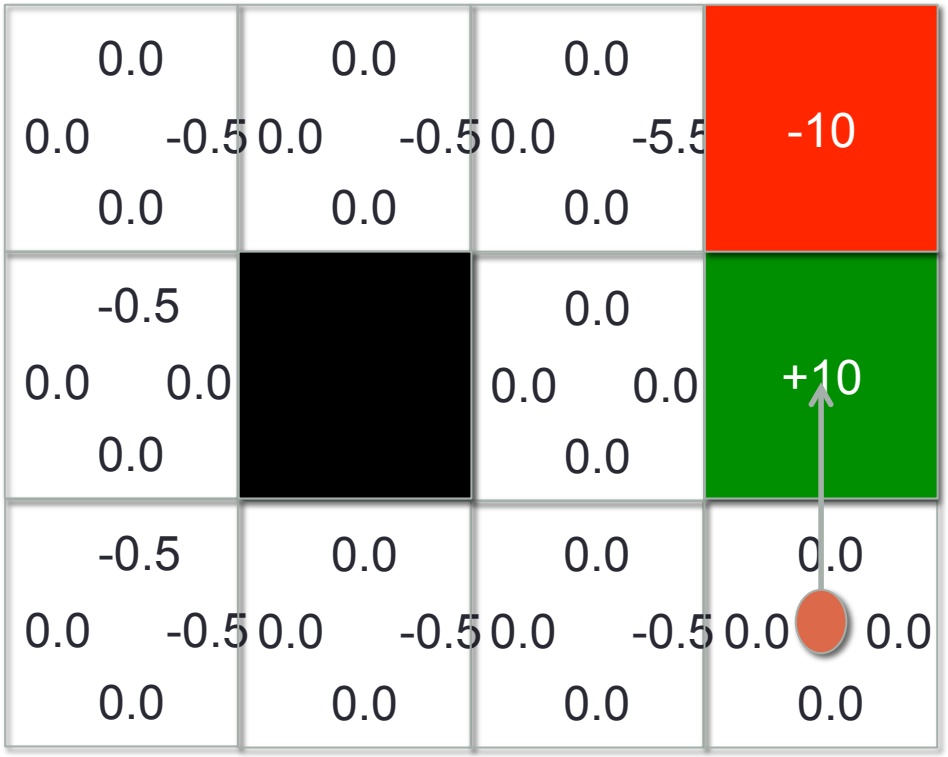


Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$



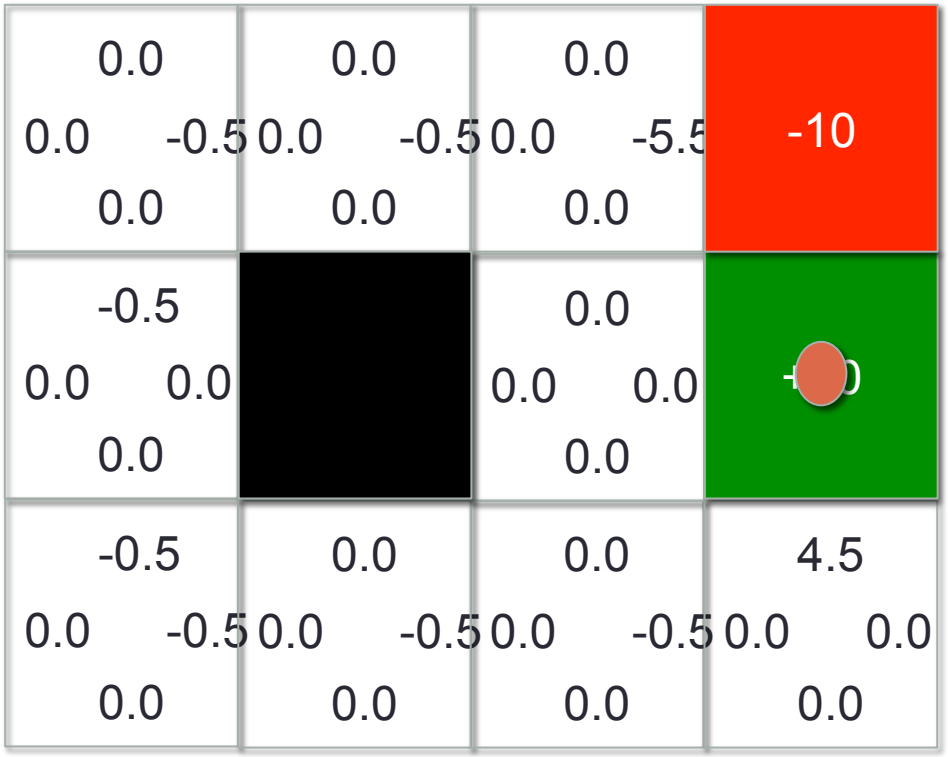
(1-0.5)0+0.5[-1 + 1* 10] = 4.5

Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$

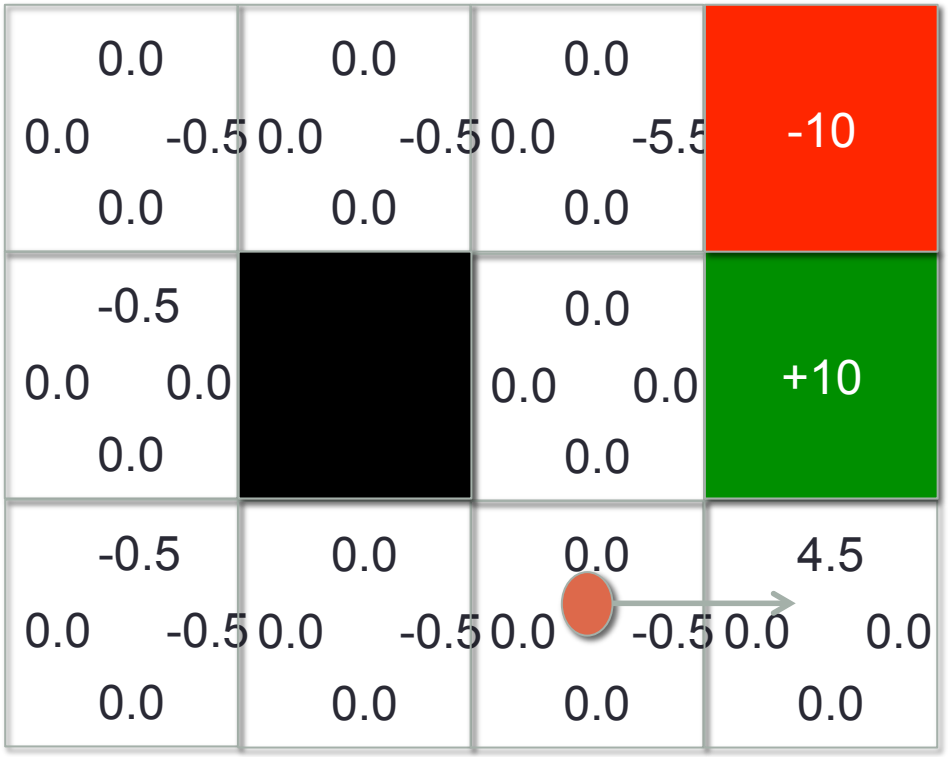


Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$



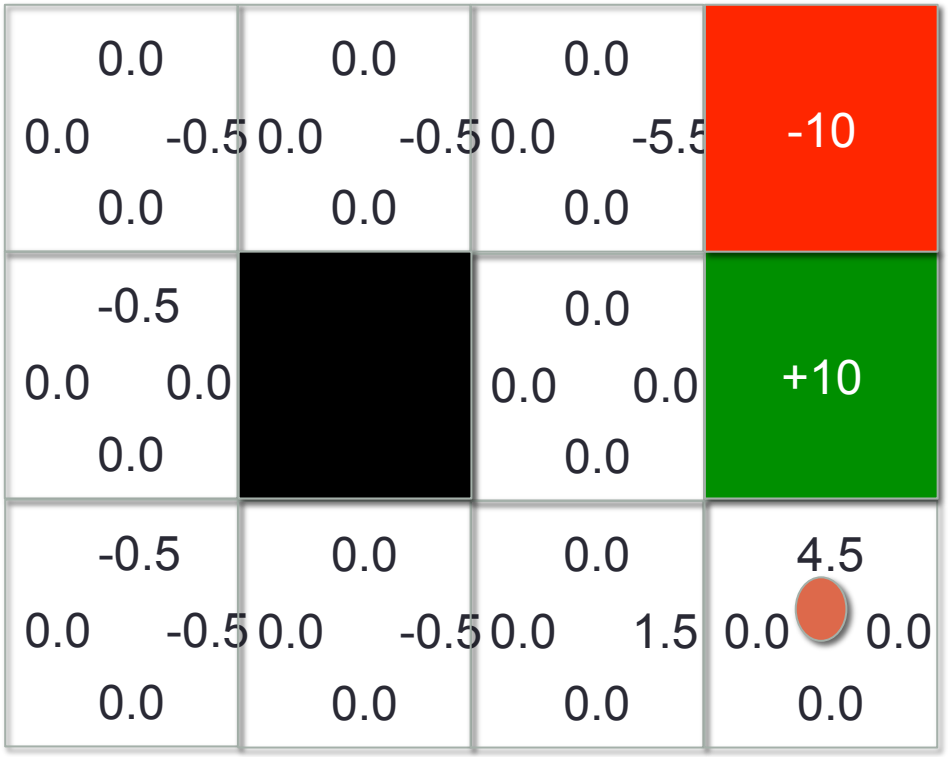
(1-0.5)(-0.5)+0.5[-1 + 1* 4.5] = 1.5

Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$

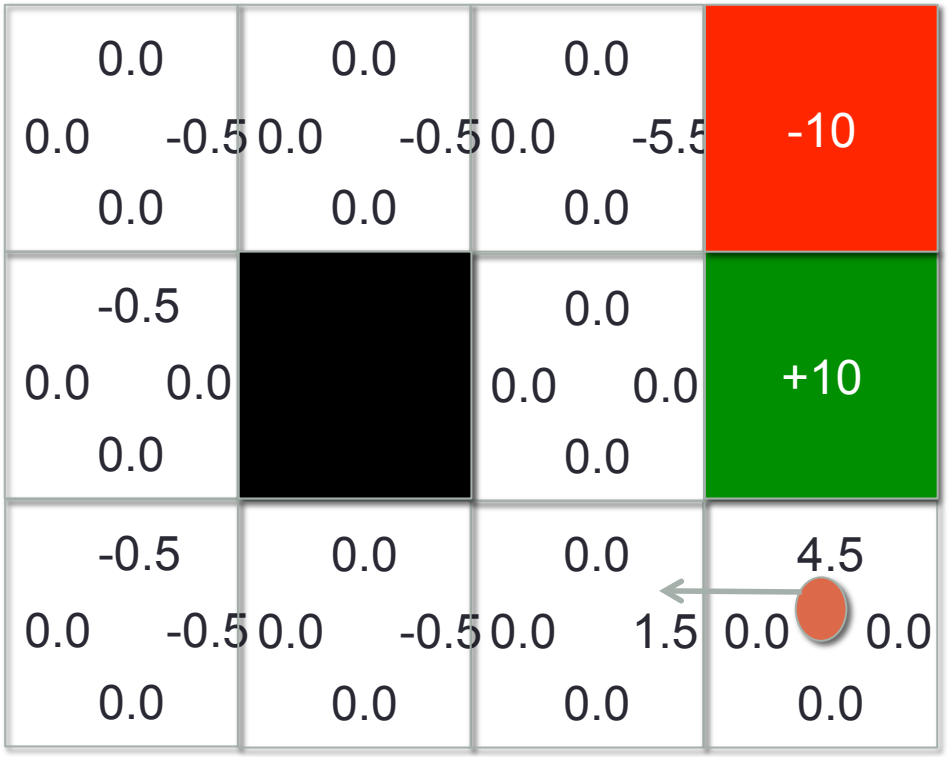


Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$



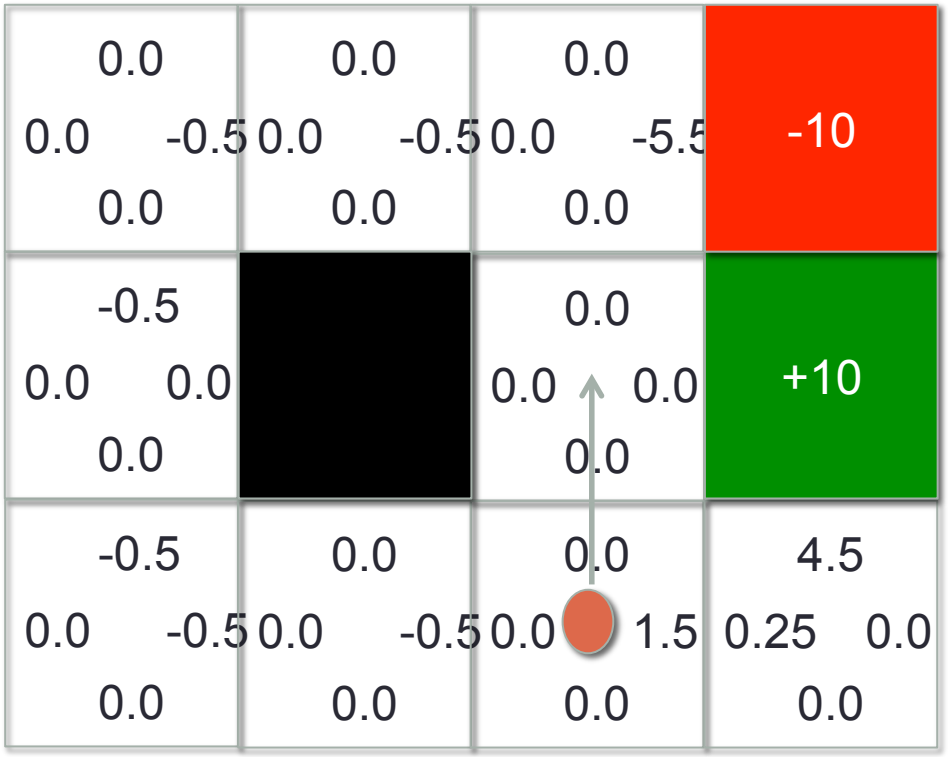
(1-0.5)(0.0)+0.5[-1 + 1* 1.5] = 0.25

Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$



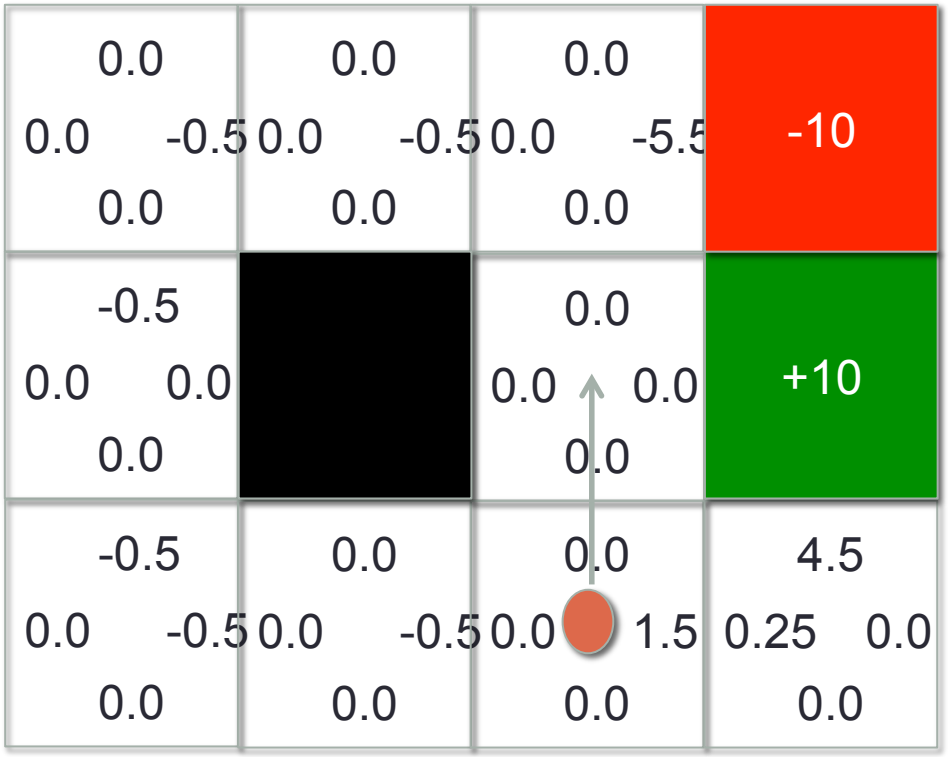
(1-0.5)(0.0)+0.5[-1 + 1* 1.5] = 0.25

Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$



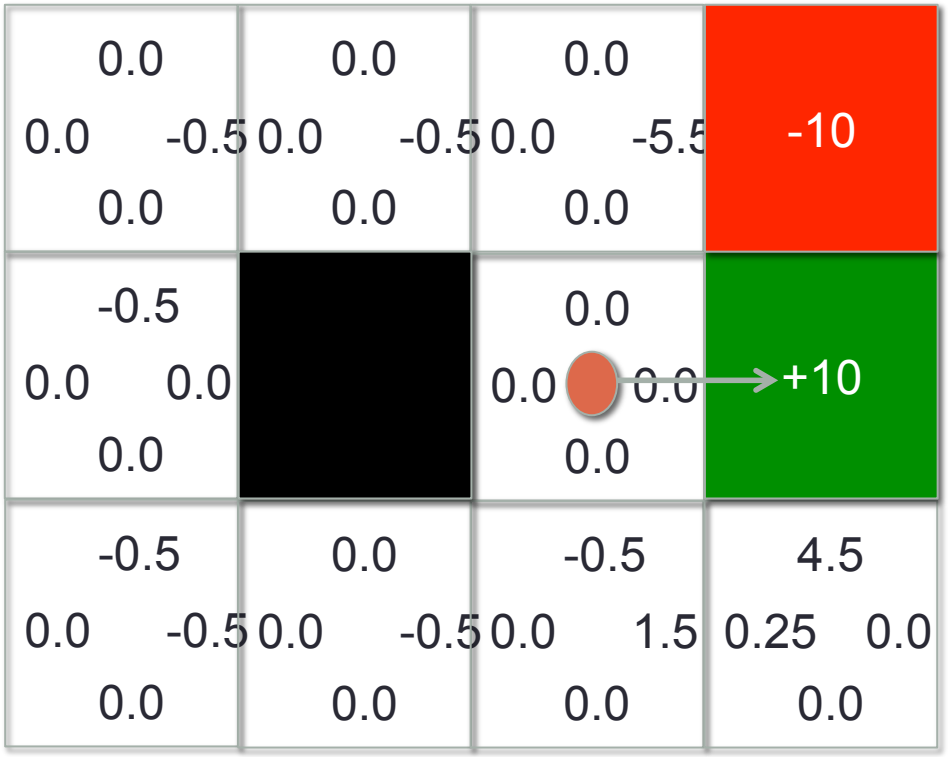
(1-0.5)(0.0)+0.5[-1 + 1* 0.0] = -0.5

Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$



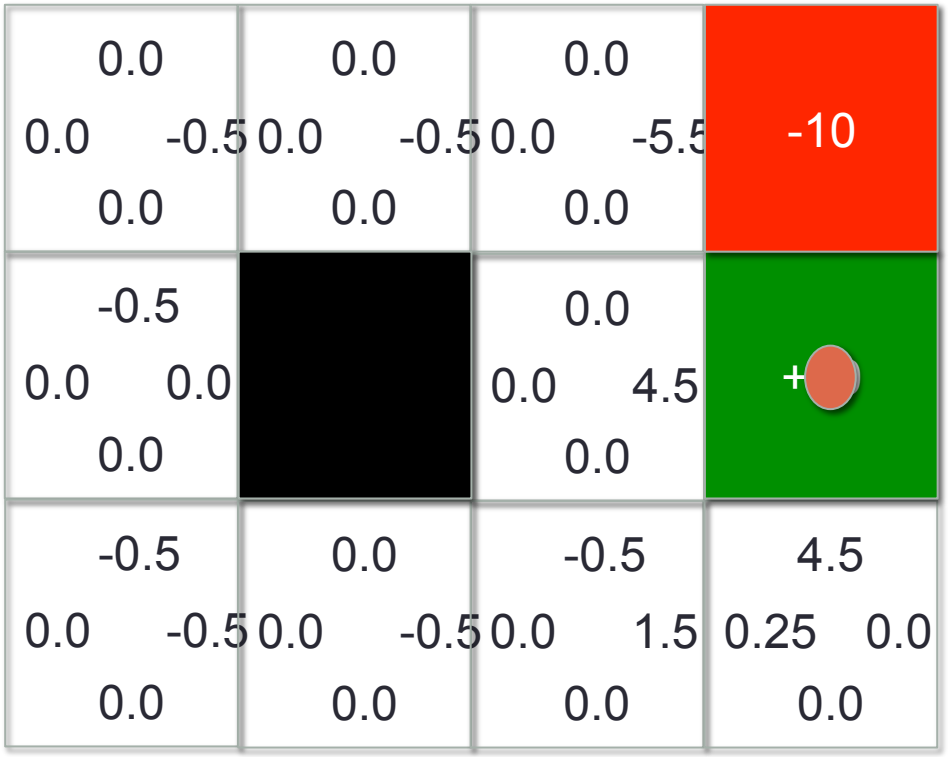
(1-0.5)(0.0)+0.5[-1 + 1* 10] = 4.5

Toy example

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

$\gamma = 1.0$

$\alpha = 0.5$



(1-0.5)(0.0)+0.5[-1 + 1* 10] = 4.5

Toy example

$\gamma = 1.0$

$\alpha = 0.5$

Many moves later

0.0 0.0 -0.5 0.0	0.0 0.0 0.37 0.0	0.0 -0.5 -8.25 2.625	-10
-0.5 0.0 0.0 0.0		0.375 0.0 6.75 0.25	+10
-0.5 0.0 -0.5 0.0	0.0 0.0 0.5 0.0	1.5 0.0 2.5 0.0	6.75 0.25 0.0 0.0

Toy example

$\gamma = 1.0$

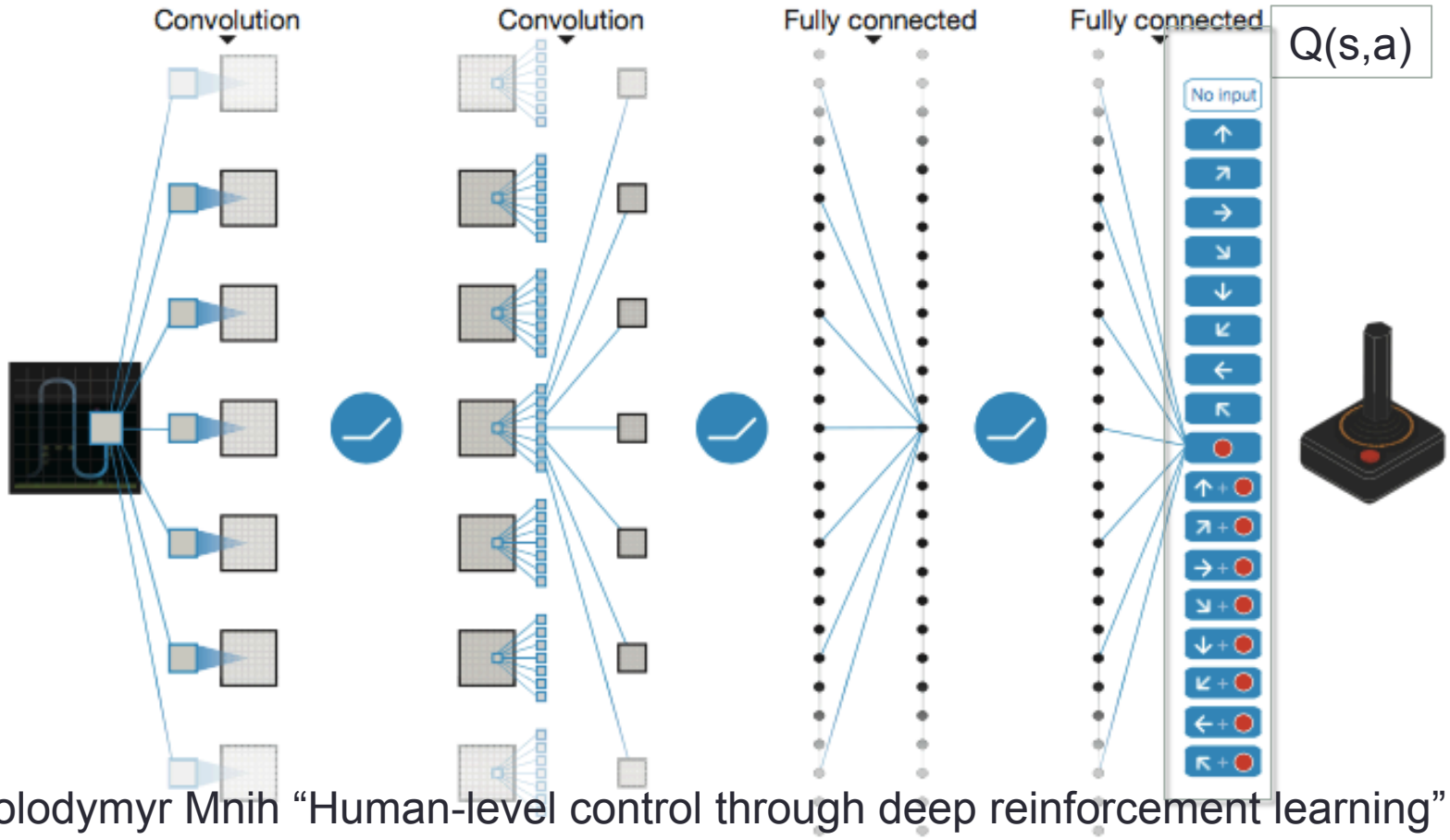
$\alpha = 0.5$

Many many more moves later

0.0 0.0 6.0 4.0	0.0 5.0 7.0 0.0	0.0 6.0 -11 8.0	-10
5.0 0.0 0.0 3.0		7.0 0.0 9.0 7.0	+10
4.0 0.0 6.0 0.0	0.0 5.0 7.0 0.0	8.0 6.0 8.0 0.0	9.0 7.0 0.0 0.0

Deep Q learning (DQN)

- Use a DNN to learn the Q function. Outputs are the possible actions.



Volodymyr Mnih “Human-level control through deep reinforcement learning” 2015.

DQN Loss Function

- Similar to the update equation of Q-learning

$$\text{new}Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a' \in A} Q(s',a')]$$

- We want to move our current belief (the Q generated from the network) towards the sampled Q

$$Loss = \{Q(s,a) - [r + \gamma \max_{a' \in A} Q(s',a')]\}^2$$

A couple more tricks

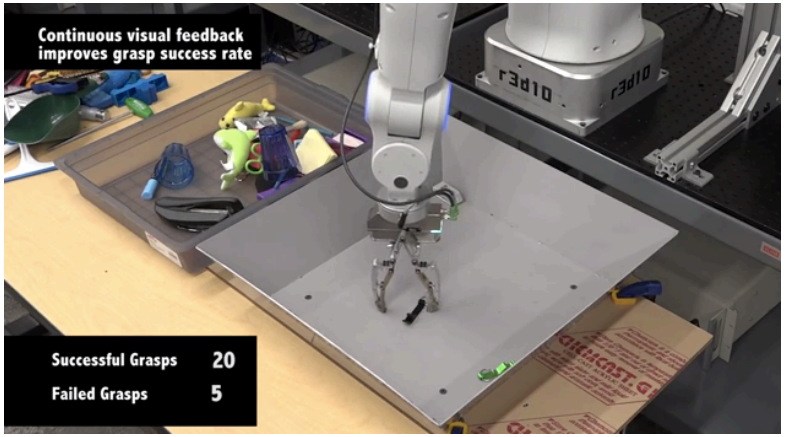
- Experience replay
  - Adjacent frames in a game are highly correlated
  - Saves past experiences and re-use them in the mini-batch
- Target network
  - Updating the Q belief too quickly leads to instability
  - Keeps a target network that updates less frequently.

$$Loss = \{Q(s,a) - [r + \gamma \max_{a' \in A} \hat{Q}(s',a')]\}^2$$

Other applications using reinforcement learning

Robotics

http://spectrum.ieee.org/automaton/robotics/drones/drone-uses-ai-and-11500-crashes-to-learn-how-to-fly

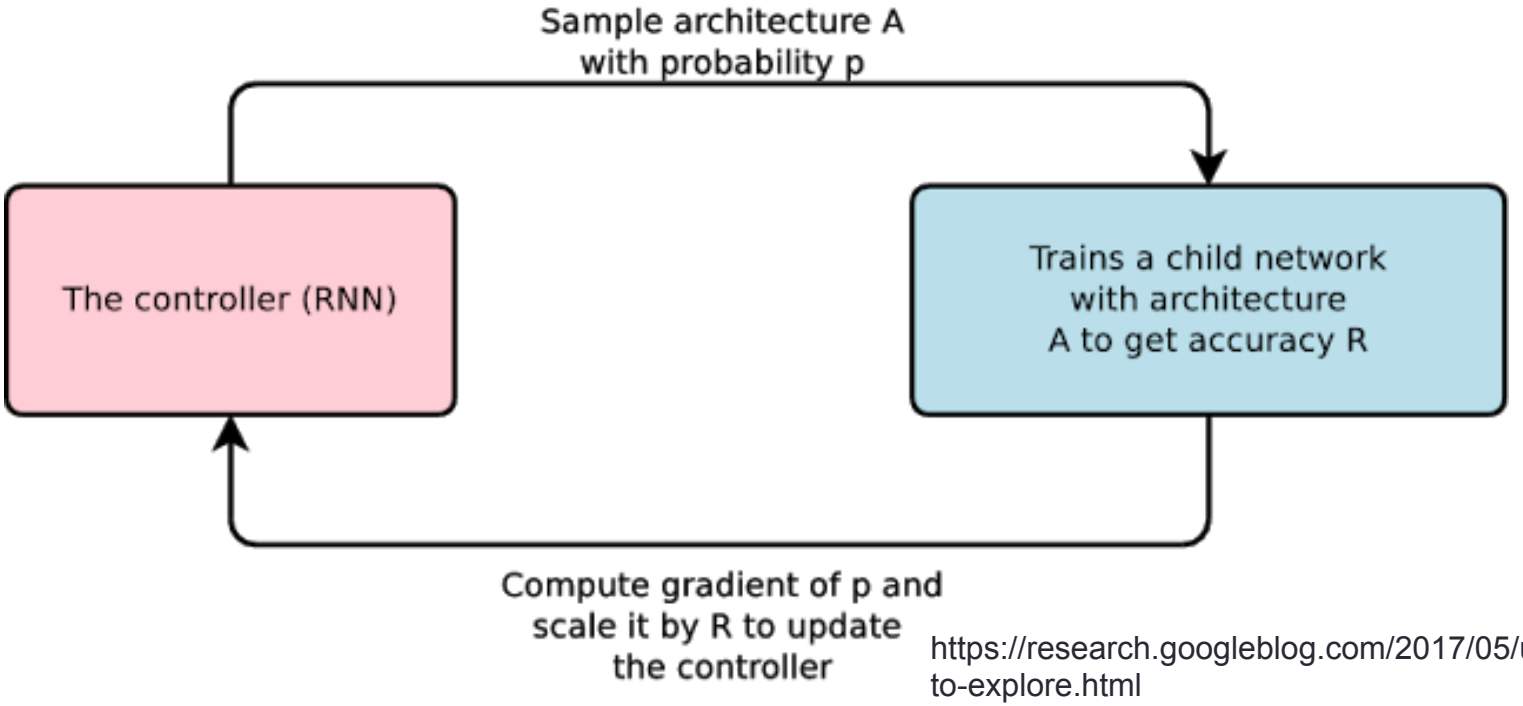


https://www.youtube.com/watch?v=HbHqC8Himol

https://research.googleblog.com/2016/03/deep-learning-for-robots-learning-from.html

Model selection for deep architectures

- Tuning a network takes time
- Let machine learning learns how to tune a network
- Matches or outperforms ML experts performance



https://research.googleblog.com/2017/05/using-machine-learning-to-explore.html

Genetic Algorithm for Reinforcement learning

- Updating based on the gradients for RL is hard
  - Credit assignment problem
  - Target network as a workaround for too fast updates
- Estimate the direction to minimize the loss by random perturbations
  - Perturb the weights and let the new agents roam in the environment
- Highly parallelizable

https://blog.openai.com/evolution-strategies/

Flappy bird genetic algorithm

- https://www.youtube.com/watch?v=aeWmdojEJf0

Yann LeCun's comment

Yann LeCun
March 14, 2016 ·

Follow

Statement from a Slashdot post about the AlphaGo victory: "We know now that we don't need any big new breakthroughs to get to true AI"

That is completely, utterly, ridiculously wrong.

As I've said in previous statements: most of human and animal learning is unsupervised learning. If intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake. We know how to make the icing and the cherry, but we don't know how to make the cake.

We need to solve the unsupervised learning problem before we can even think of getting to true AI. And that's just an obstacle we know about. What about all the ones we don't know about?

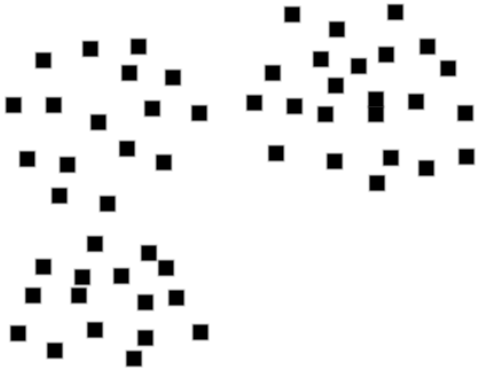
#deeplearning #AI #AlphaGo

Unsupervised Learning

- Clustering
- Autoencoders
- Generative Adversarial Networks

Clustering

- Discover underlying structure of data



- Learn the hidden class labels in the data.

Simple clustering

- K-means
  - Find the means, and assign
- GMM
  - Find the means, covariance, then soft assignment
- K-means vs GMM
  - Spherical covariance vs any covariance
  - Hard vs soft assignment

- Need to pick k

Latent Dirchlet Allocation (LDA)

- Another method that learns the hidden labels.
- Mostly used for topic modeling and document classification

- Let's cover some basic knowledge about graphical models

Topic modeling motivation

- I want to be able to give probabilities to words/sentences/ documents.

- P(“Roses are red. Violets are blue.”) = ?

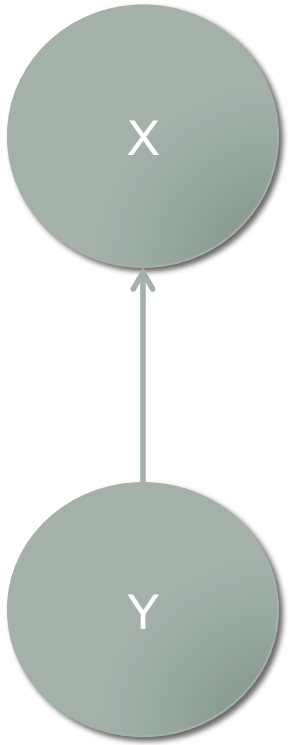
- P(X1 = Rosses, X2 = are, X3 = red, …)
  - This probability distribution is intractable

Introduction to graphical models

- Given rain in the morning predict sunny or cloudy in the afternoon

P(X|Y)	Y = Rain	Y = No rain
X = cloudy		
X = sunny		

Graphical representation of the relationship

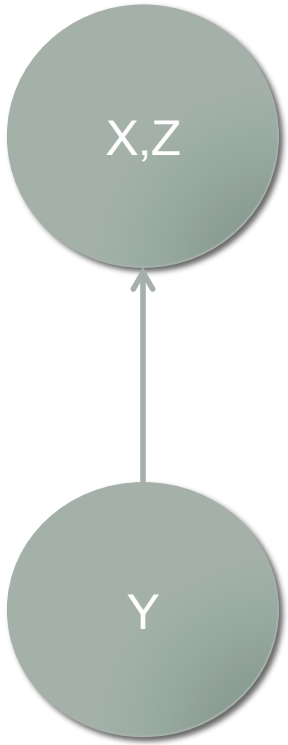


Curse of dimensionality

- Given rain in the morning predict sunny or cloudy and windy or cloudy in the afternoon

P(X,Z|Y)	Y = Rain	Y = No rain
X = cloudy Z = windy		
X = sunny Z = windy		
X = cloudy Z = no wind		
X = sunny Z = no wind		

Graphical representation of the relationship

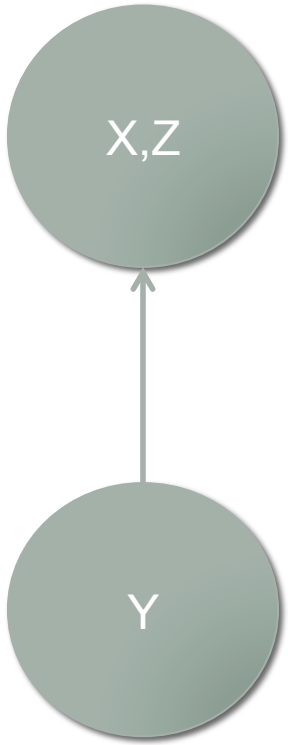


Curse of dimensionality

- If there are more variables, this table keeps getting bigger and bigger

P(X,Z|Y)	Y = Rain	Y = No rain
X = cloudy Z = windy		
X = sunny Z = windy		
X = cloudy Z = no wind		
X = sunny Z = no wind		

Graphical representation of the relationship



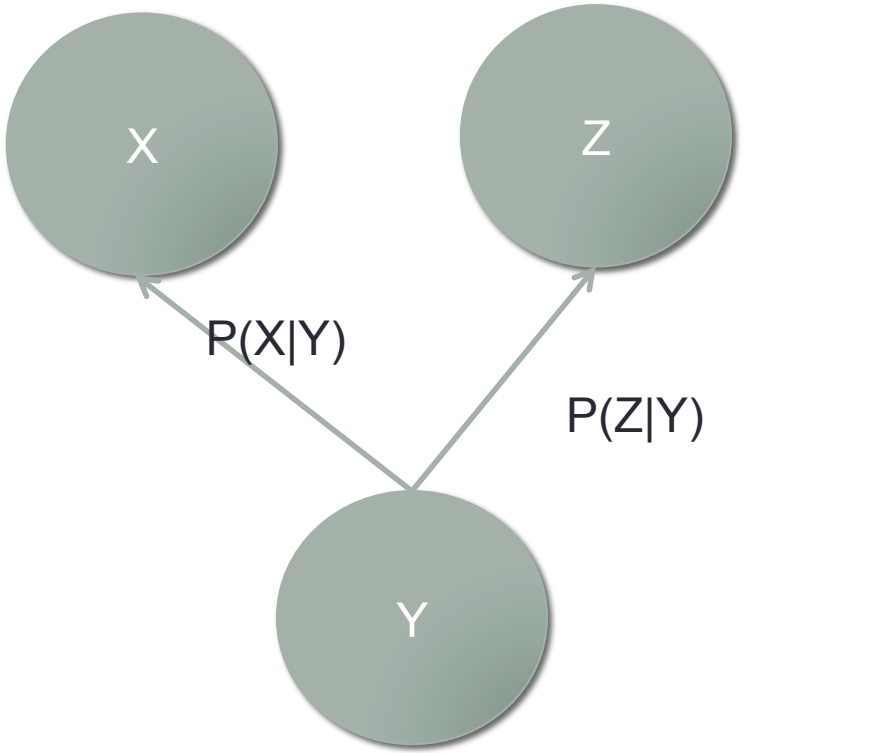
Naïve Bayes

- Assume conditional independence

P(X|Y)	Y = Rain	Y = No rain
X = cloudy		
X = sunny		

P(Z|Y)	Y = Rain	Y = No rain
Z = windy		
Z = no wind		

Graphical representation of the relationship

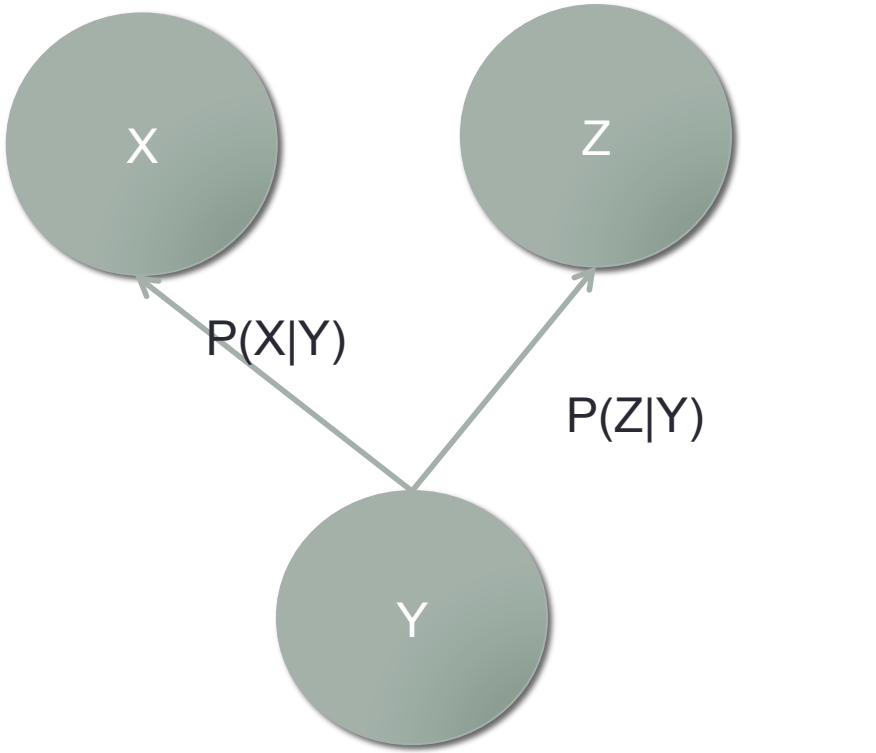


Probabilistic models and independence assumptions

- Most probabilistic models will assume some independence assumption in order to reduce the parameter space

- Graphical representations helps summarize the relationship
  - Arrows denote a conditional relationship

Graphical representation of the relationship



Probabilistic models and independence assumptions

- Also implies
- P(X,Y,Z)
  = P(Y)P(X|Y)P(Z|Y)

Graphical representation of the relationship

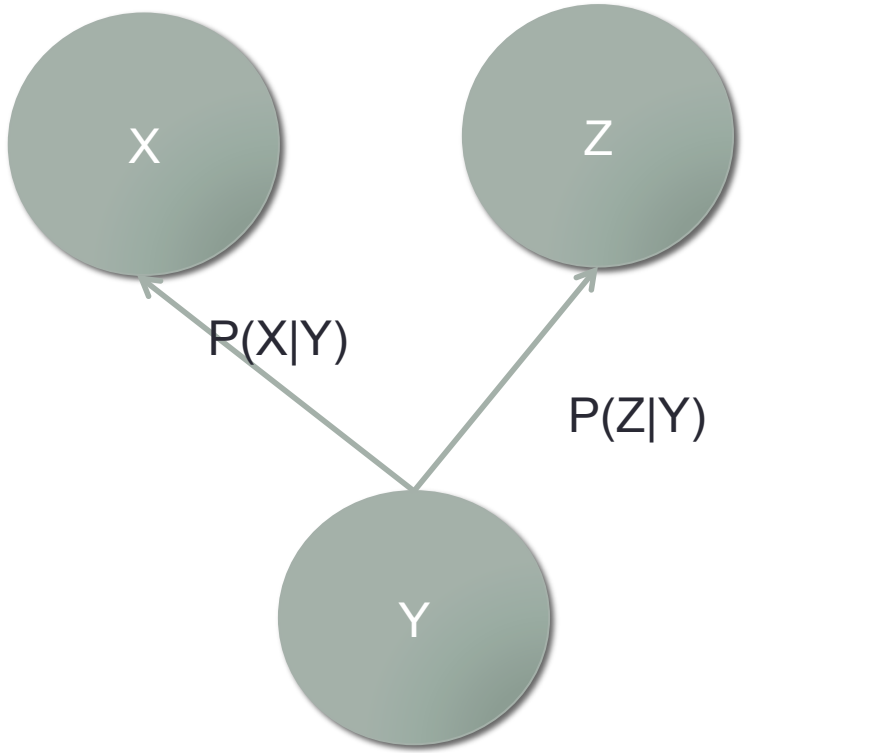


Plate notation

- Sometimes you have too many relationships.

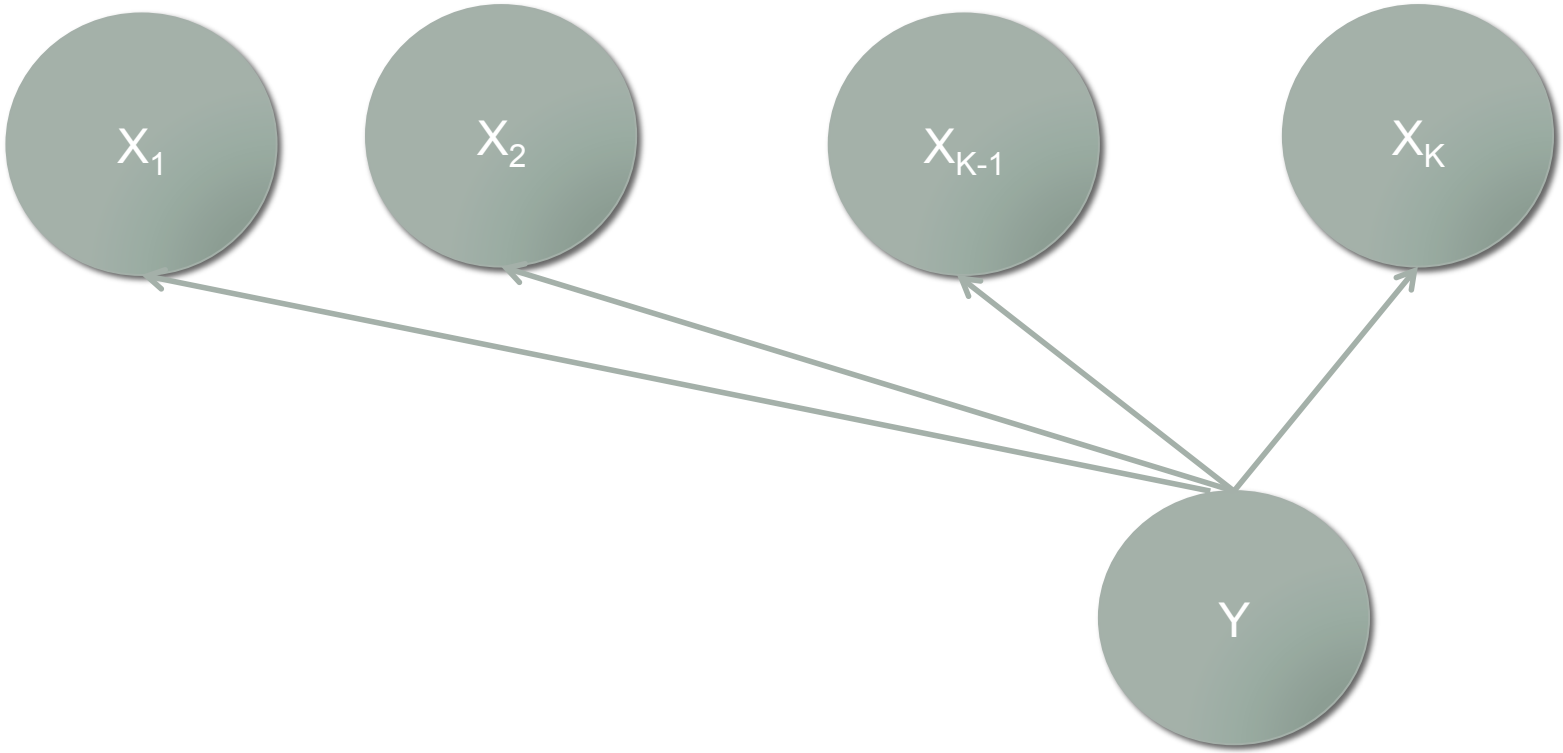
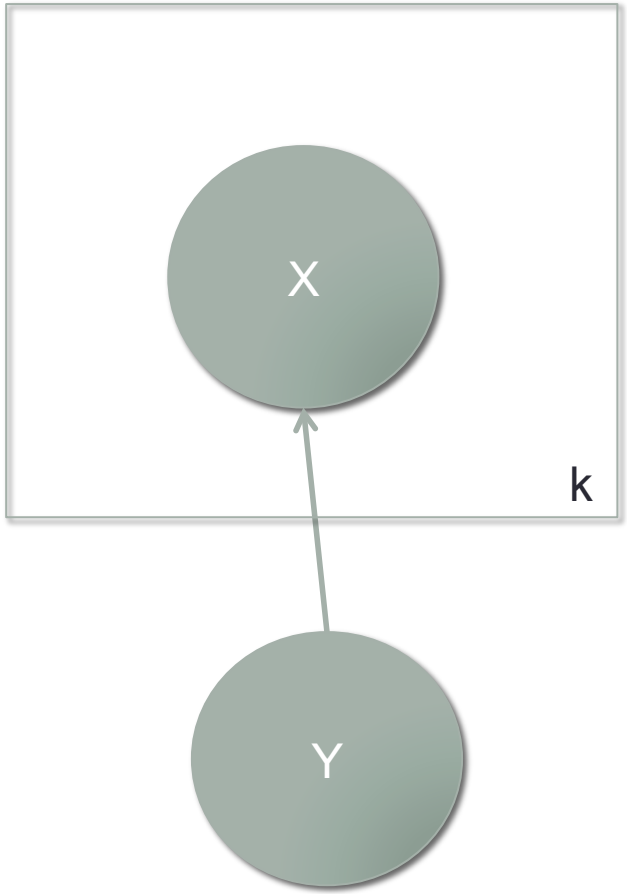


Plate notation

- Summarize by using a square box with number



Topic modeling motivation

- I want to be able to give probabilities to words/sentences/ documents.
- P(“Roses are red. Violets are blue.”) = ?
- P(X1 = Rosses, X2 = are, X3 = red, …)
  - This probability distribution is intractable
- Assume independence
  - P(X1) P(X2) P(X3)…
  - Lost all relational structure. Can we do better?
    - Assume words come from topics. How?

Topic modeling example

- I want to write about Pattern Recognition

- Topics I want to cover
  - GMMs
  - Neural networks
  - Linear regression

- Let's assign the proportion of topics
  - P(GMMs) = 0.3
  - P(Neural networks) = 0.6
  - P(Linear regression) = 0.1

Topic modeling example

- I want to write about Pattern Recognition

- Topics I want to cover
  - GMMs
  - Neural networks
  - Linear regression

- Let's assign the proportion of topics
  - P(GMMs) = 0.3
  - P(Neural networks) = 0.6
  - P(Linear regression) = 0.1

Topic modeling example

- Each topic can have a word proportion
  - GMM
  - P(Guassian) = 0.7, P(MLE) = 0.1,…
  - Neural networks
  - P(neuron) = 0.3, P(softmax) = 0.5, …
  - Linear regression
  - ….

Topic modeling

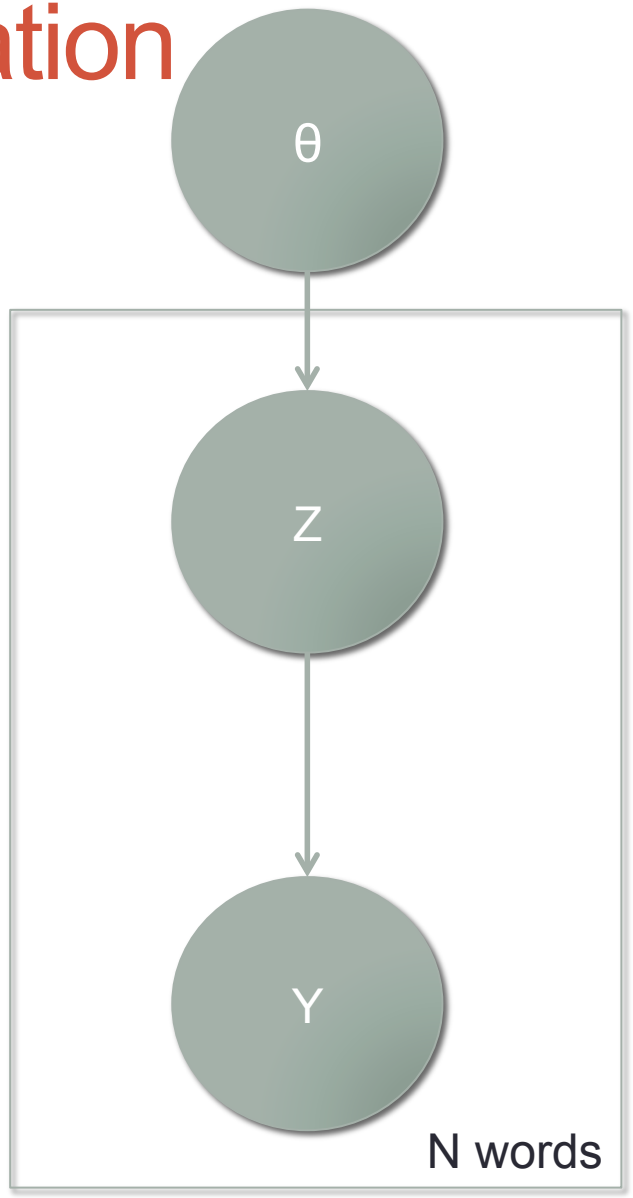
- P(W1=Guassian,W2=neuron,…)
- Let Zn be the topic of Wn
- = P(W1= Gaussian|Z1= GMM) P(W2 = neuron| Z2 = neural networks)* …
  *P(Z1 = GMM) P(Z2 = neural networks) *…

- = 0.7 * 0.3* …

*0.3 * 0.6 *…

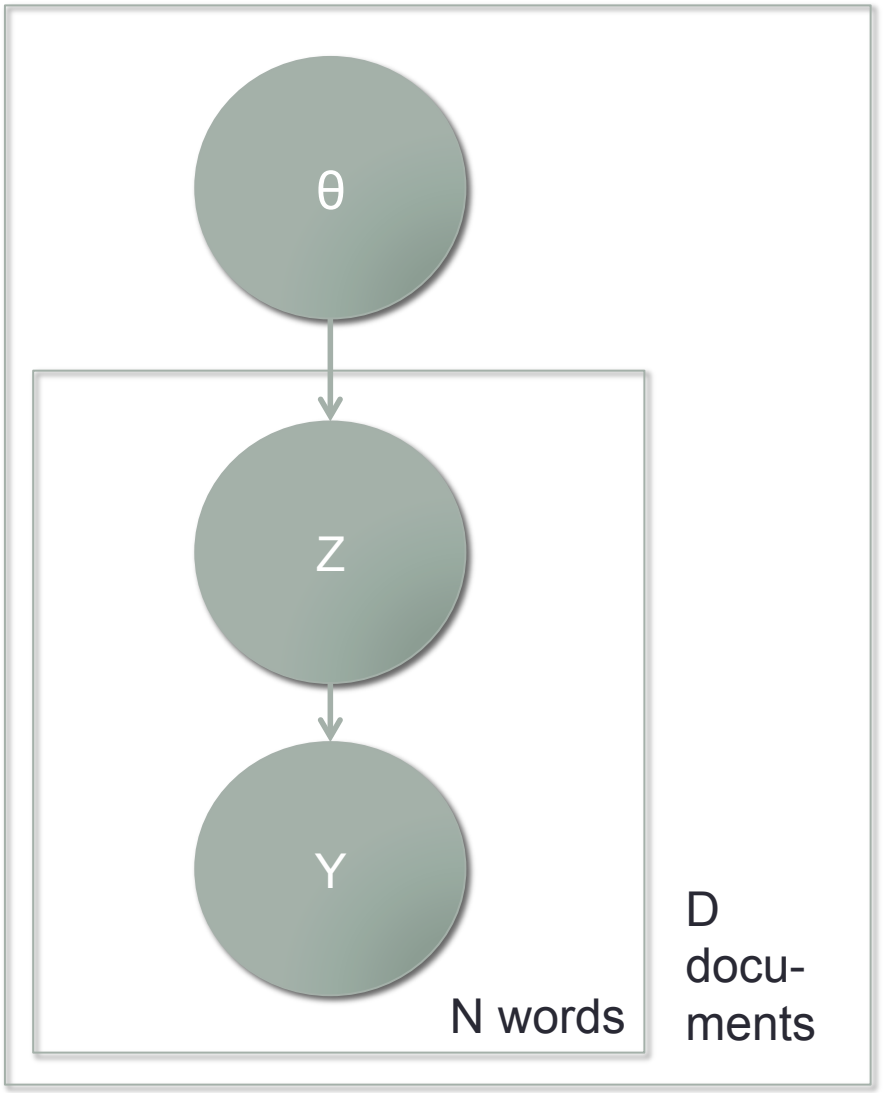
Graphical representation

θ is the distribution of topics for each word



Multiple documents

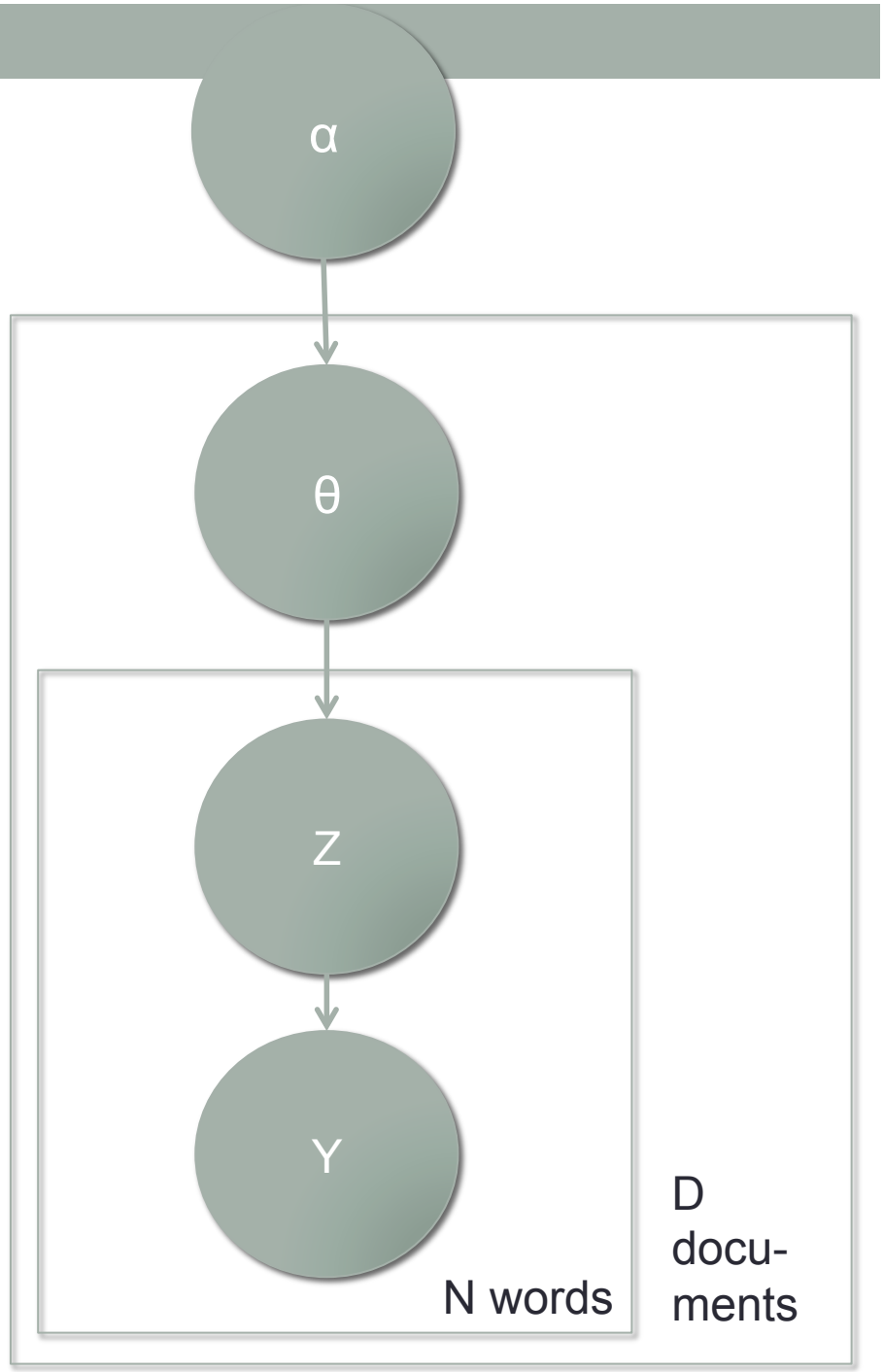
θ is the distribution of topics for each word in the document



Multiple documents

θ is the distribution of topics for each word in the document

α is the distribution of documents topics for each document



α

- α is a Dirichlet distribution
- Or a distribution of distributions…
- ??????
- Example: rolling a die
- P(x=1) = 0.3, P(x=2) = 0.1,…
- This is a distribution. (Multinomial distribution)
- But what if I have a bag of dices, each with different distributions.
- α tells me how probability of what kind of die I will get

Dirichlet distribution

- pdf

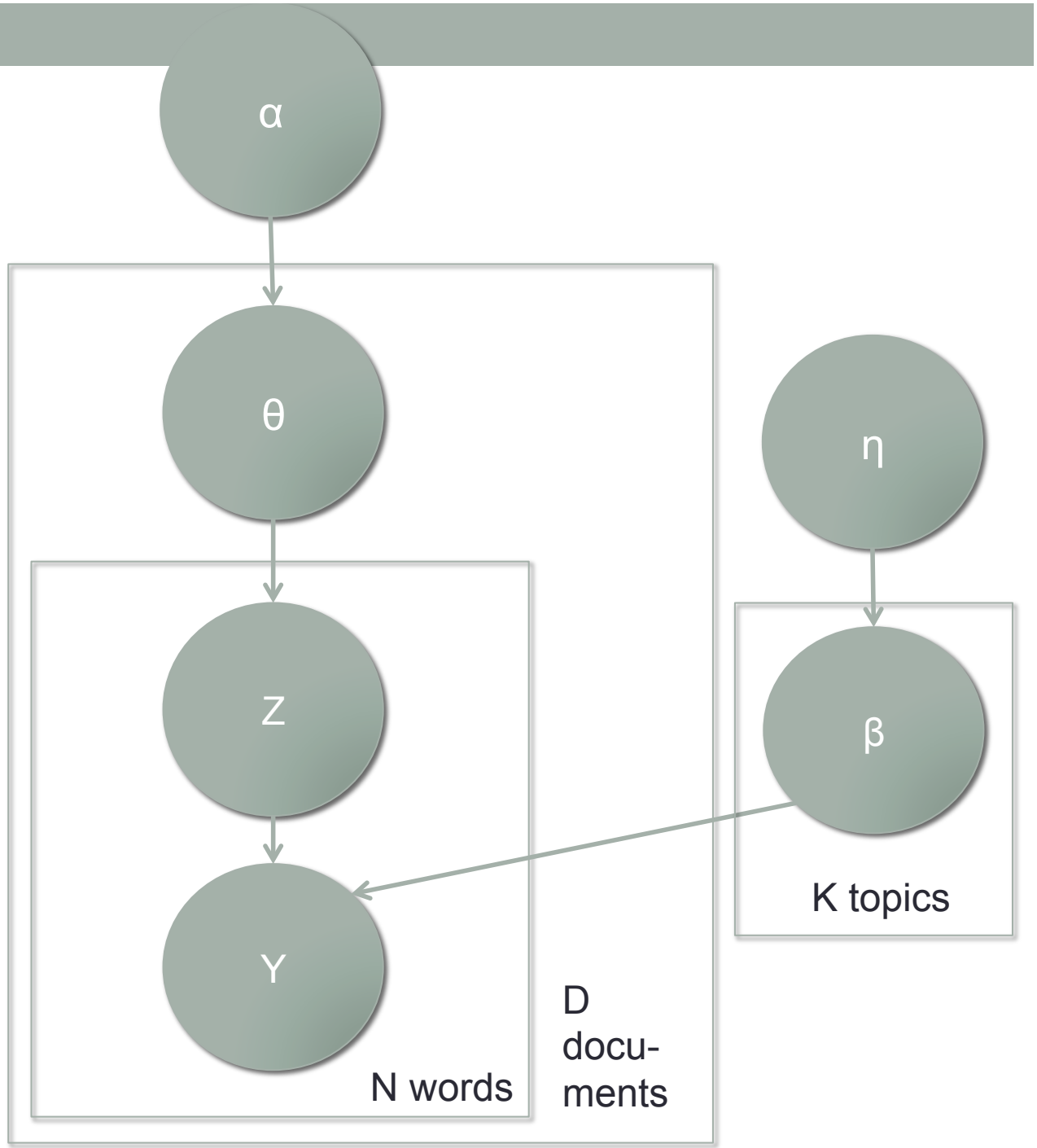
Parameters giving preference to each side of die/topic

$$f(x_1, \ldots, x_K; \alpha_1, \ldots, \alpha_K) = \frac{1}{\mathrm{B}(\boldsymbol{\alpha})} \prod_{i=1}^{K} x_i^{\alpha_i - 1}$$

Probability of each side of die, probability of each topic

Learning topic distributions

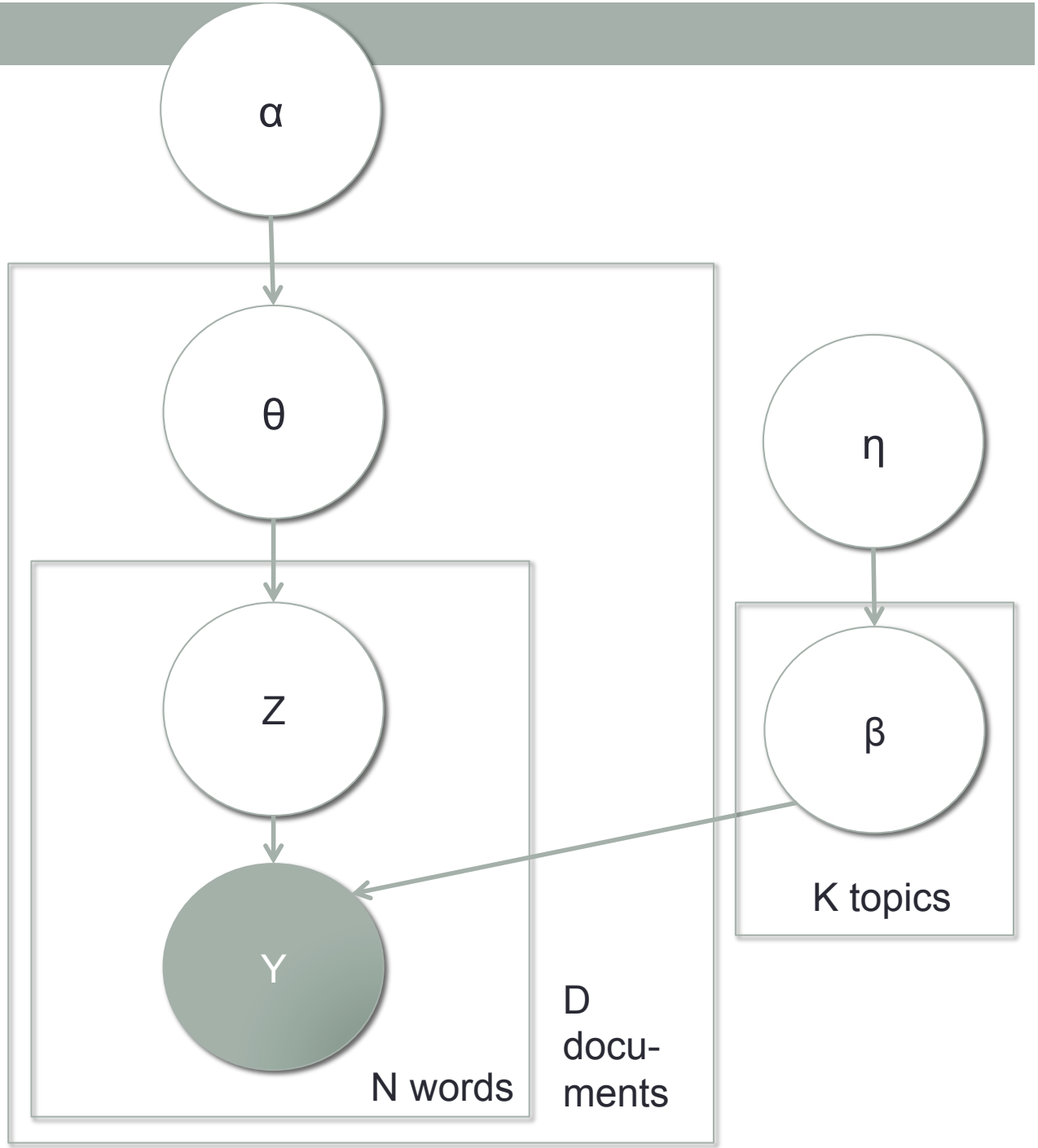
β is probability of word given topic
η is dirchlet for β



Latent variables

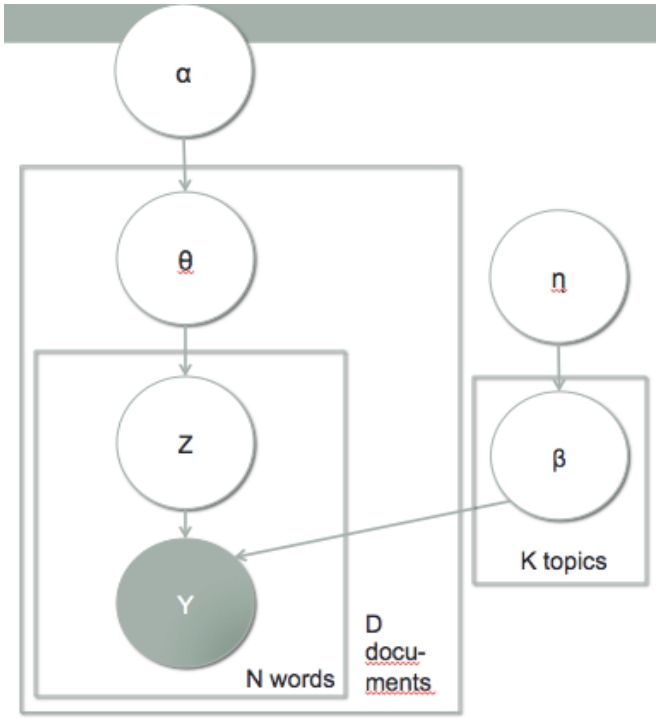
Graphical representation also use coloring to represent observed variables (fixed). Shaded are observed. While non-shaded are latent variables (z,β,θ) or hyperparameters (α,η)

Latent Dirchlet Allocation



Generative process

- For each topic k $\in$ {1,...,K}:
  - Draw word distributions β_k ~ Dir(η)
- For each document d $\in$ {1,...,D}:
  - Draw topic proportions θ_d ~ Dir(α)
  - For each word in a document n $\in$ {1,...,N}:
    - Draw a topic index z_{dn} ~ Mult (θ)
    - Generate word from chosen topic
      - w_{dn} ~ Mult(β_{zdn})



Latent Dirchlet Allocation

- Guess the topic distributions of the document
- Guess the topic assignment of each word
- Guess the distribution of words for each topic

- How do we actually learn these parameters and latent variables?
  - Gibbs sampling
  - Variational methods
- Totally beyond the scope of this class

Example usage

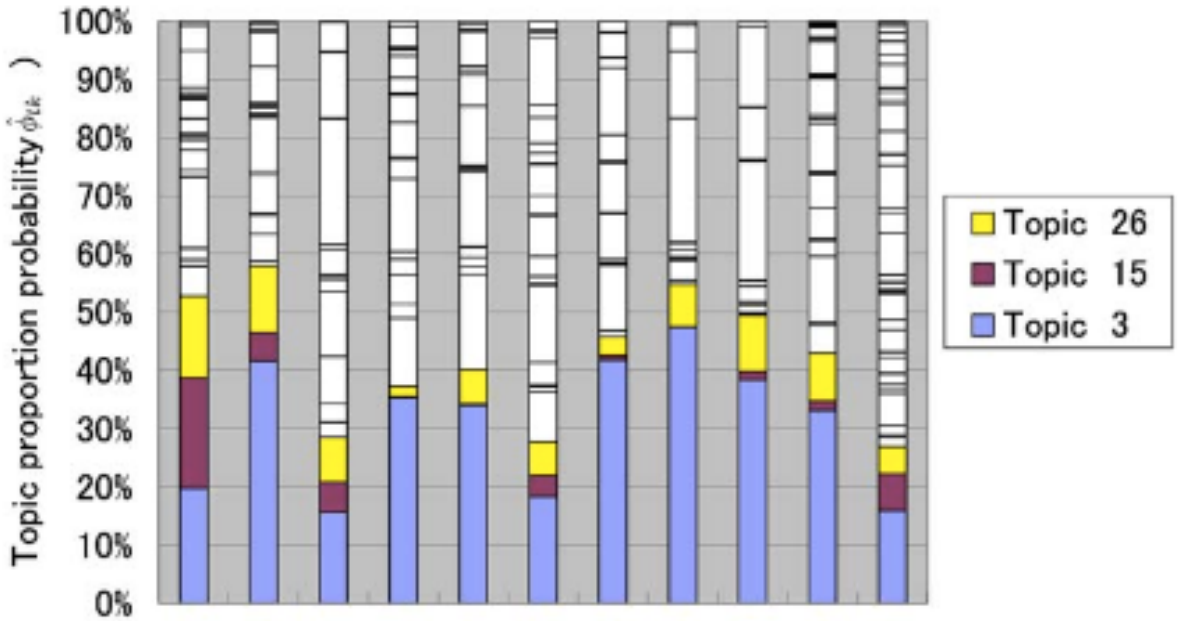
- Lots of lectures video. Can we discover the topics? Can we classify the videos? No topic labels given

Top 10 high probability nouns in word probabilities of Topics 3 ($\theta_{k=3}$), 15 ($\theta_{k=15}$), and 26 ($\theta_{k=26}$).

Topic 3 (~ classical mechanics)	Topic 15 (~astronomy)	Topic 26 (~ (time) unit)
m	Light	Percent
Energy	Degrees	Time
Force	Angle	Dollars
Mass	Frequency	Times
Point	Energy	Minutes
Velocity	Direction	Day
Direction	Waves	Bit
v	Sun	Year
Times	Star	Hour
Speed	Speed	Half

http://www.sciencedirect.com/science/article/pii/S0885230810000641

Topic distribution of each document



Discovering K

- In K-means, GMMs, LDA, we need K, the number of latent classes.
- Can we find it automatically?

Chinese Restaurant Process

- A random process is analogous to seating customers in a restaurant with infinite tables.
  - Each table has infinite seats
- First person sits in the first table
- n^{th} person sits at a table based on
  - Join existing table according to
    - number of person in that table, and table characteristics
  - Join a new empty table α c (some hyperparameter)

https://en.wikipedia.org/wiki/Chinese_restaurant_process

Chinese Restaurant Process

- After an assignment to a table, the seat is fixed. The table distribution is also updated.
- CRP automatically starts a new class when it thinks there should be a new class.
- Hyperparmeter c control how often to start a new class.

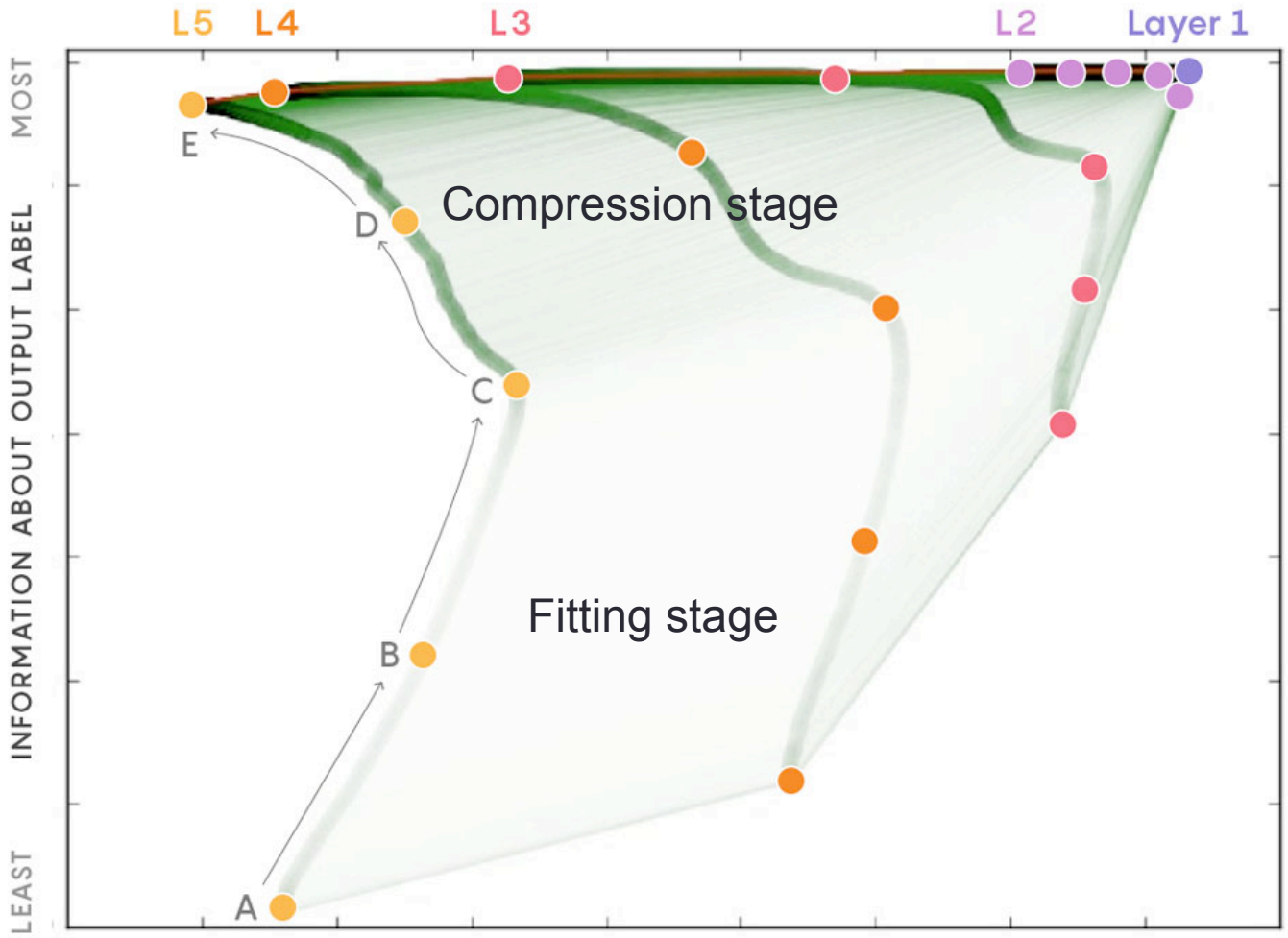
- Note: This is theoretically nice, but for most applications just pick a K.

Unsupervised Learning

- Clustering
- Autoencoders
- Generative Adversarial Networks

Inside Deep Learning

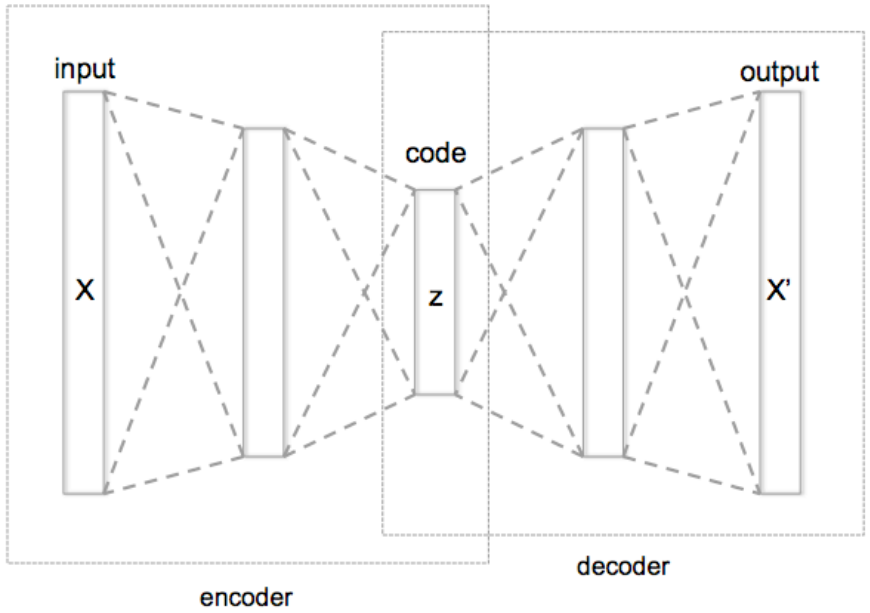
New experiments reveal how deep neural networks evolve as they learn.



https://www.quantamagazine.org/new-theory-cracks-open-the-black-box-of-deep-learning-20170921

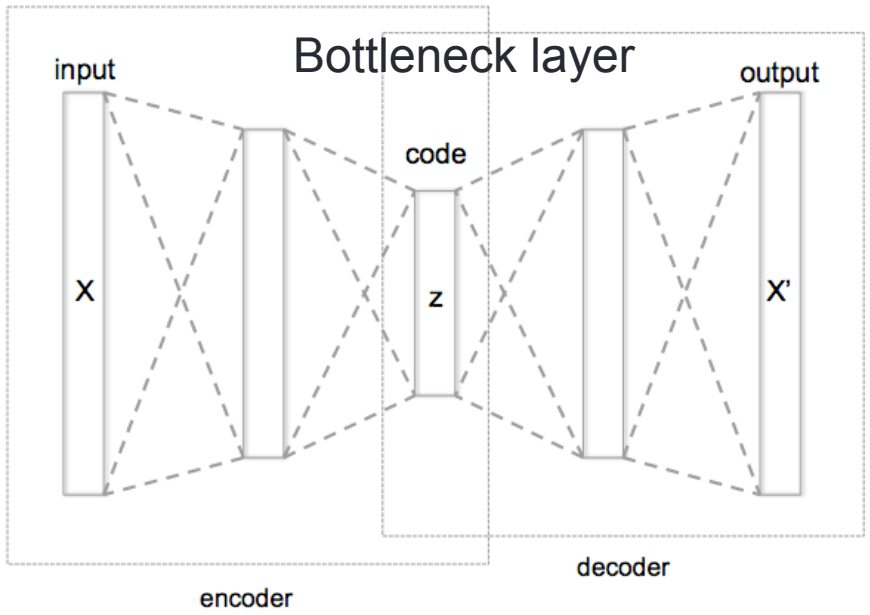
Autoencoders

- Use neural network properties to compress data
  - Automatically learn features



Simplest model

- Input X, and wants to output the same X
- Needs a bottleneck layer to enforce non-trivial solutions (identity matrix)
- Loss function = $||x-x'||^2$



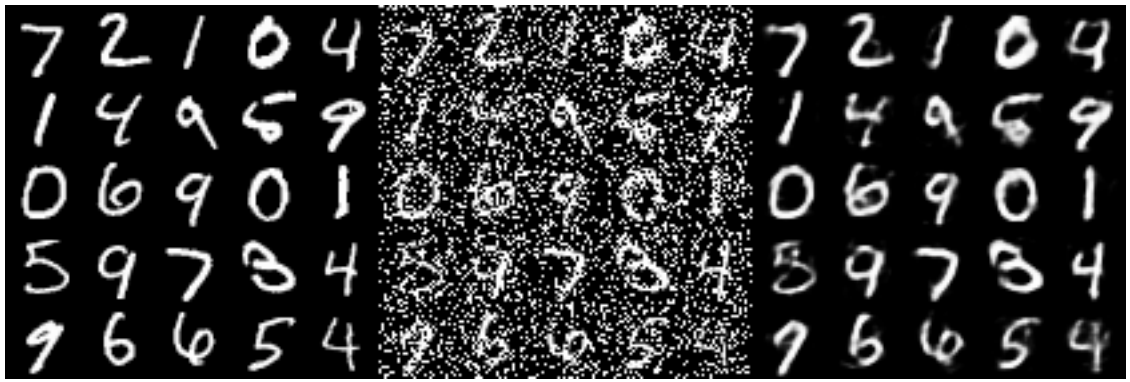
Denoising autoencoder

- An autoencoder that denoise corrupted inputs
  - Blur image, sound corrupted by noise

- How?
  - Corrupt your input x -> x''
  - Use x'' as the input, and minimize the same loss
  - Loss = $||x-x'||^2$

Example

Original Corrupted Generated

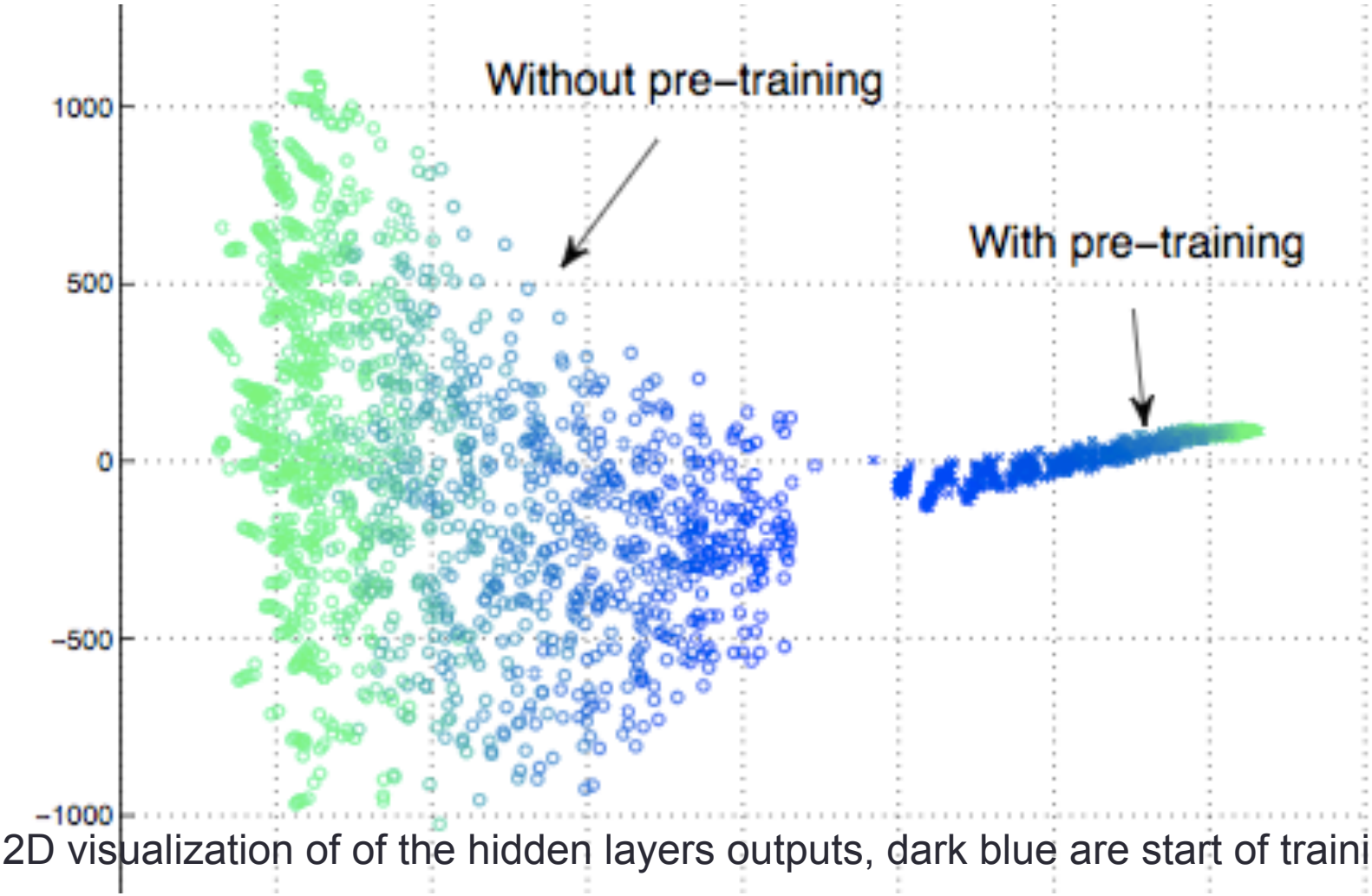


https://www.doc.ic.ac.uk/~js4416/163/website/autoencoders/denoising.html

Other uses

- Besides denoising, can also be used to make your “code” more robust to the type of corruption you introduced.
- Another usage is for pre-training the neural network when you have lots of un-labeled data.
  - Use autoencoder on un-labeled data.
    - Do it layer by layer: Train with 1 layer, then train with 2 layer, … until finish
  - Do regular SGD on labeled data with regular cross entropy loss
  - Helps with initialization (pre-training)

Pre-training effects



2D visualization of of the hidden layers outputs, dark blue are start of training

http://www.jmlr.org/papers/volume11/erhan10a/erhan10a.pdf

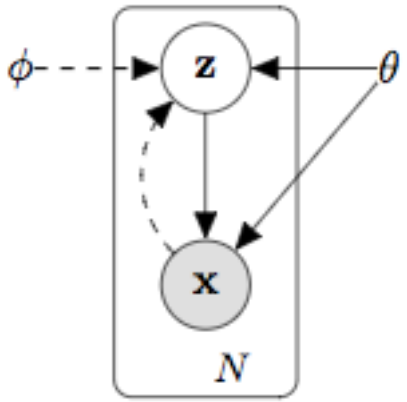
Variational Autoencoders (VAE)

- Assume X is generated from a latent variable Z
  - Ex: a picture of a cat is generated from class label cat
- VAE wants to find this latent variable Z in the coding layer
- Our code can be generated from
  - $q_\phi(z|x)$ (our neural network)
  - which tries to mimick the true $p_\theta(z|x)$ that we don't know.

VAE reward (not loss)

$$E[\log P(X|z)] - D_{KL}[Q(z|X)\|P(z)]$$

Decoder Encoder True Code



VAE reward explained

- We want to maxmimize

$$E[\log P(X|z)] - D_{KL}[Q(z|X)\|P(z)]$$

- D_{KL} is the KL-divergence
  - Distance between distributions

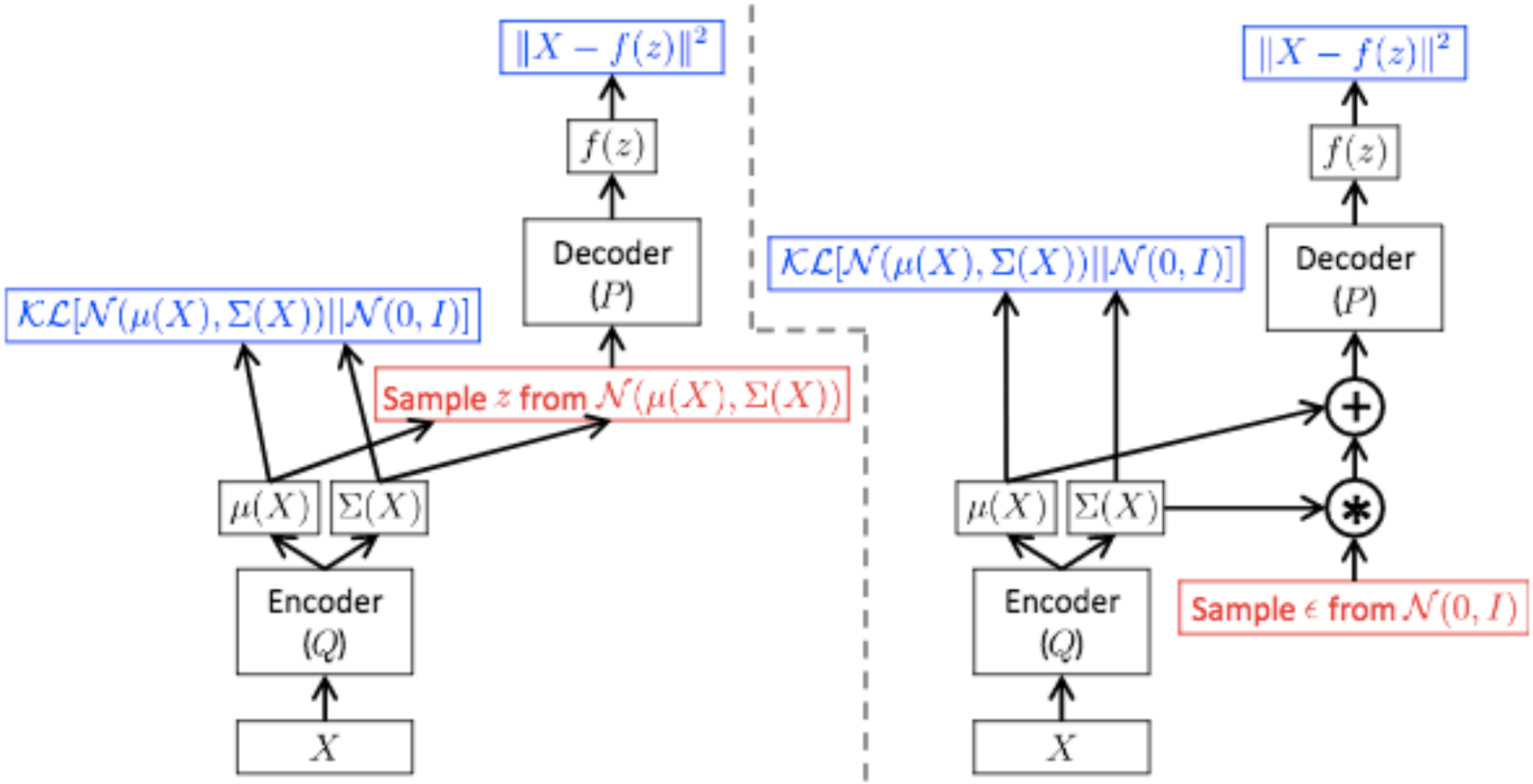
$$D_{\mathrm{KL}}(P\|Q) = -\sum_i P(i)\,\log\frac{Q(i)}{P(i)},$$

- The first term means we need to generates X in the best way possible from z. This is actually maximum likelihood estimation.
- What about P(z)?
  - This is what we get to pick, and the VAE will try to match it.
  - Simplest choice is N(0,1)
- For more info, read https://arxiv.org/abs/1606.05908

And https://wiseodd.github.io/techblog/2016/12/10/variational-autoencoder/

VAE Training

$$E[\log P(X|z)] - D_{KL}[Q(z|X)\|P(z)]$$



Generative Adversarial Networks

- Let’s recall what generative models are

Generative Models

- Naïve Bayes, Bayes classifiers are generative models
- Learn the model for each class y given input features x p(x|y). (The likelihood probability)
  - Note we also model p(y) and essentially p(x,y)
- To do classification we want to solve for the best y given input feature x (the posterior)

$$y^* = argmax_y P(y|x)$$

- We can use Bayes' rule

$$y^* = argmax_y \frac{P(x|y)P(y)}{P(x)}$$

Discriminative models

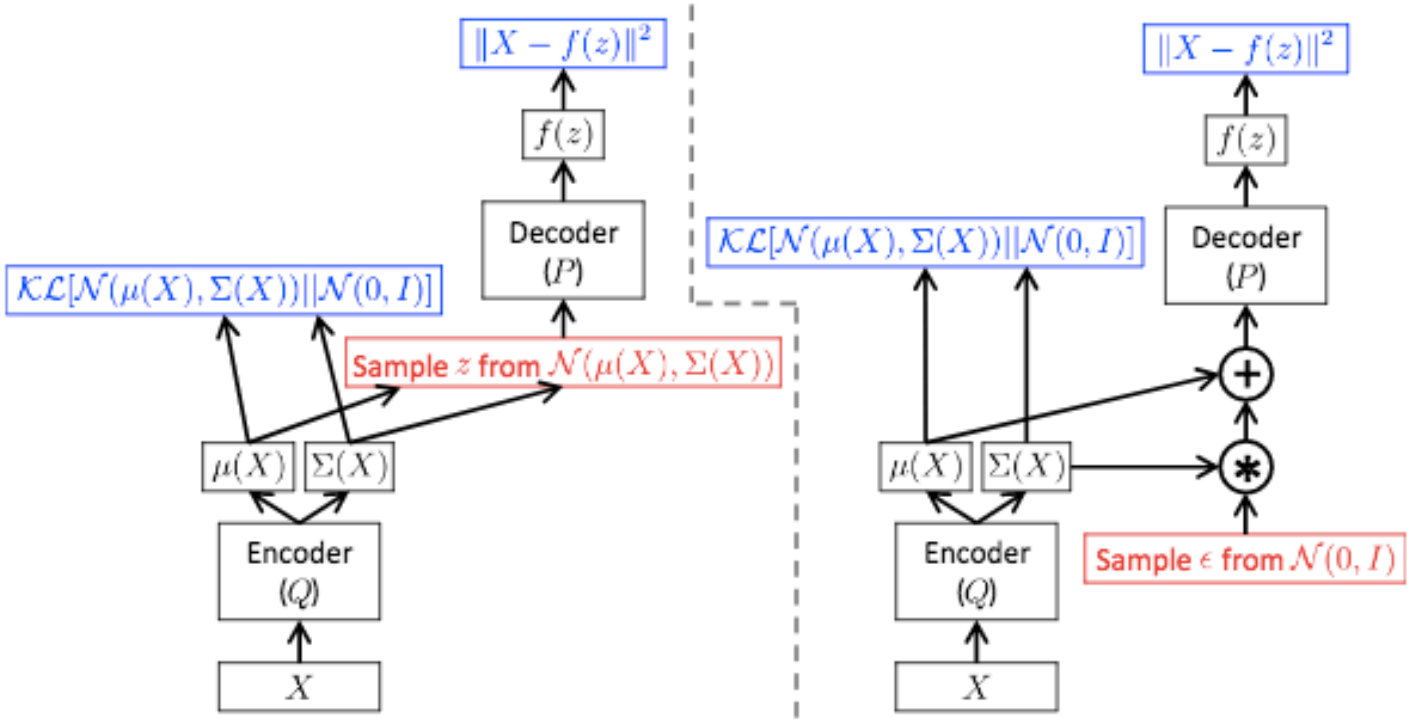
- Discriminative models model P(y|x) directly

$$y^* = argmax_y P(y|x)$$

- P(y|x) is called the posterior probability
- Generally, P(y|x) can be any function h(y,x) that gives a score for each class
  - Logistic regression
  - SVM
  - Neural networks

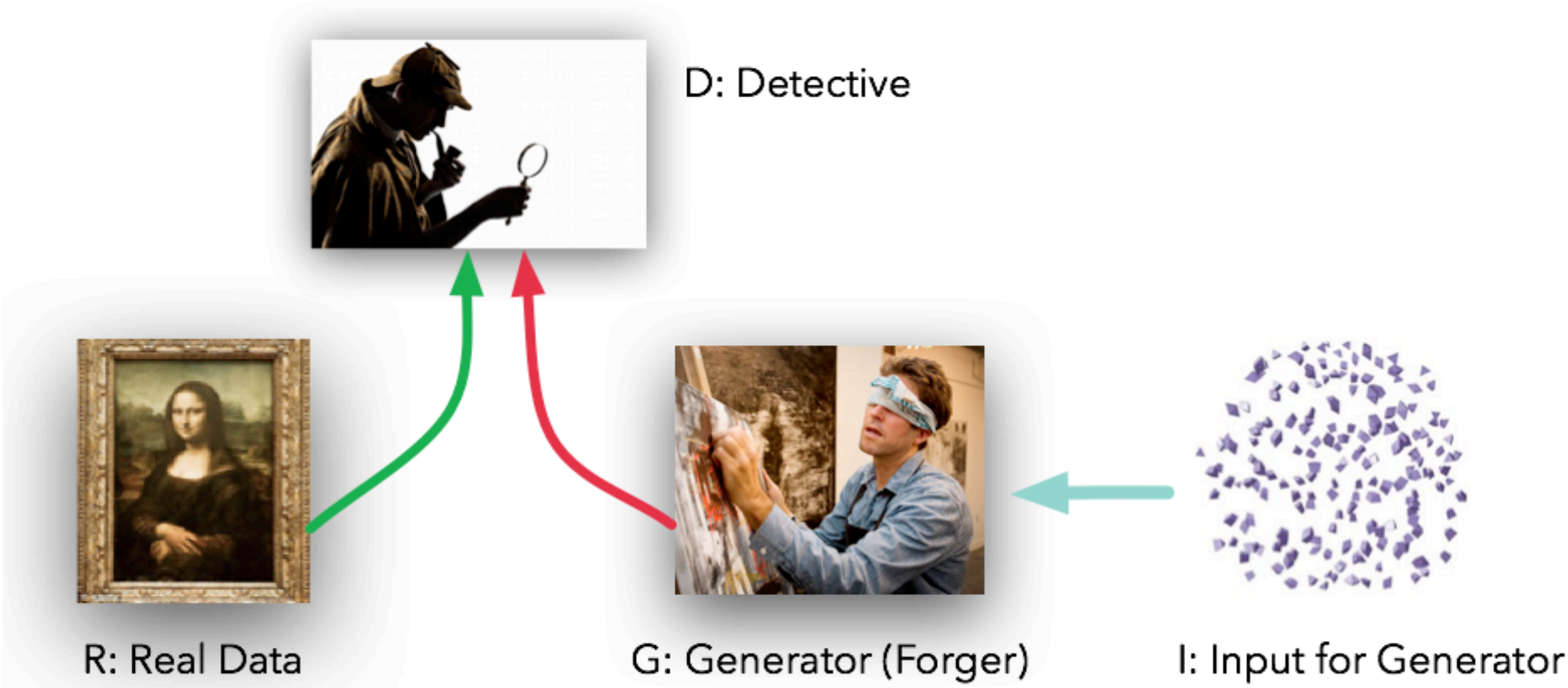
Generating via sampling

- With P(X|Y) we can create a new X by being given Y



Generative Adversarial Networks (GAN)

- Consider a money counterfeiter
  - He wants to make fake money that looks real
  - There's a police that tries to differentiate fake and real money.
- The counterfeiter is the adversary and is generating fake inputs. – Generator network
- The police is try to discriminate between fake and real inputs. – Discriminator network



https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f

GAN in a nutshell

D( ) – discriminator, outputs probability of real input
G( ) – generator, generates new data from random noise

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Samples from z (noise), and generate G(z). The discriminator wants to give a low probability

Samples from data (real), The discriminator wants to give a high probabiliy

- Minmax: minimize the loss in the worst case scenario.
- At equilibrium, the discriminator will output 0.5 for real and fake data.

GAN training

- Create two networks, D and G
  - Discriminator training
    - Generates mini-batch of fake things + real things
    - GSD with loss

$$\frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(\boldsymbol{x}^{(i)}\right)+\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$$

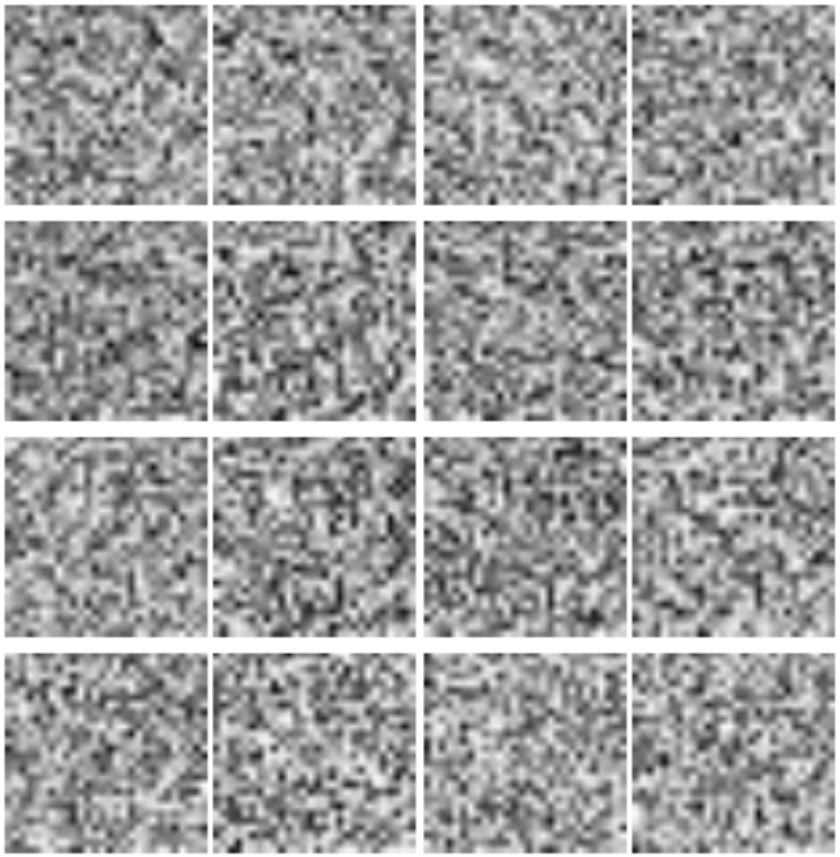
  - Generator training
    - Generate fake things
    - GSD with loss

$$\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

- Alternate between the two

GAN example

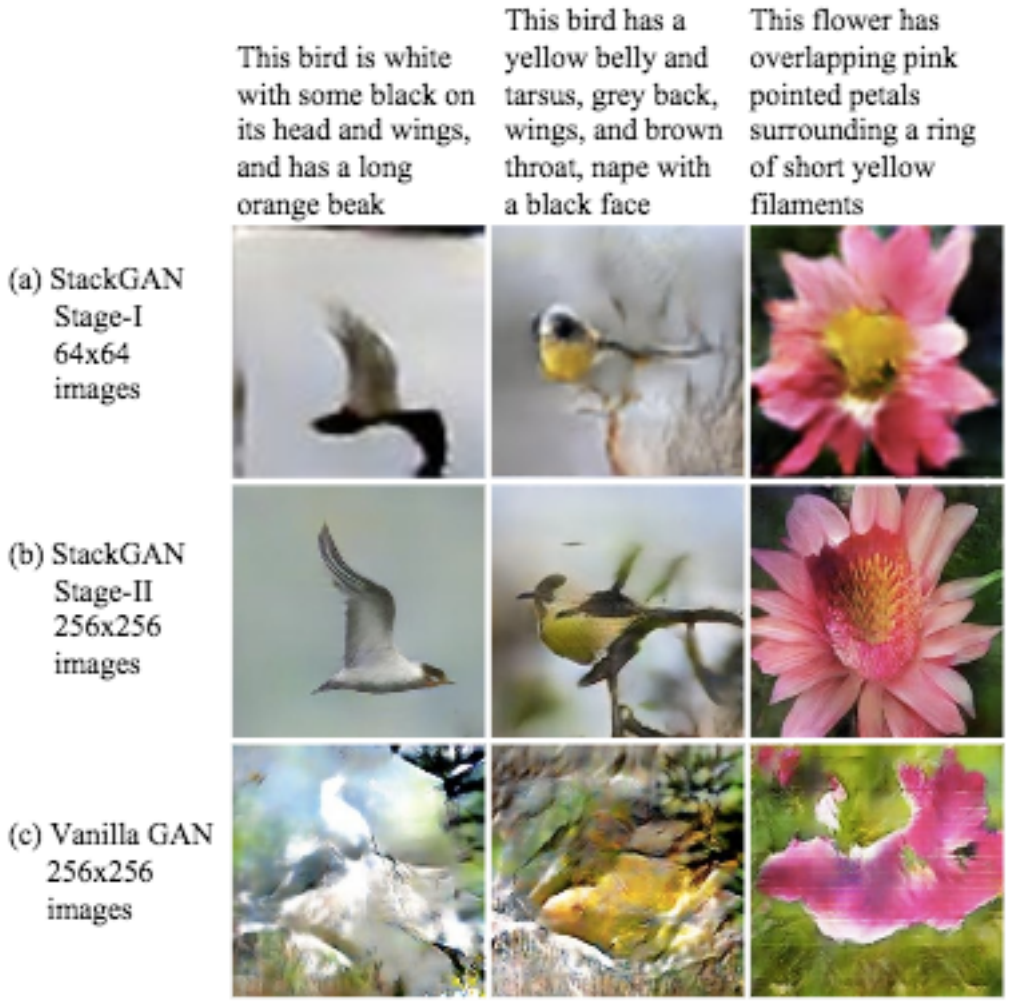
Generator output starts from random noise and gets better as we train.

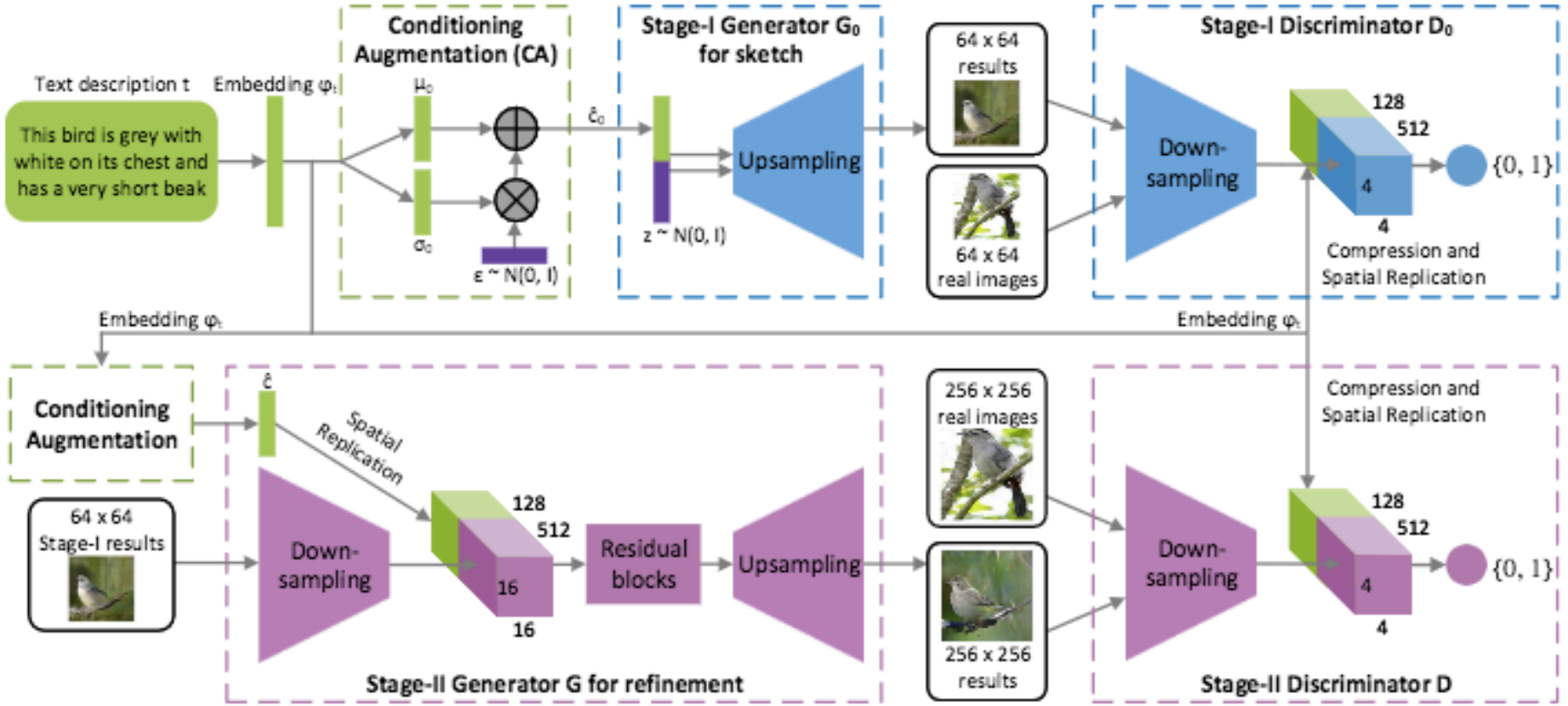


https://wiseodd.github.io/techblog/2016/09/17/gan-tensorflow/

Other uses

- StackGAN: text to photo

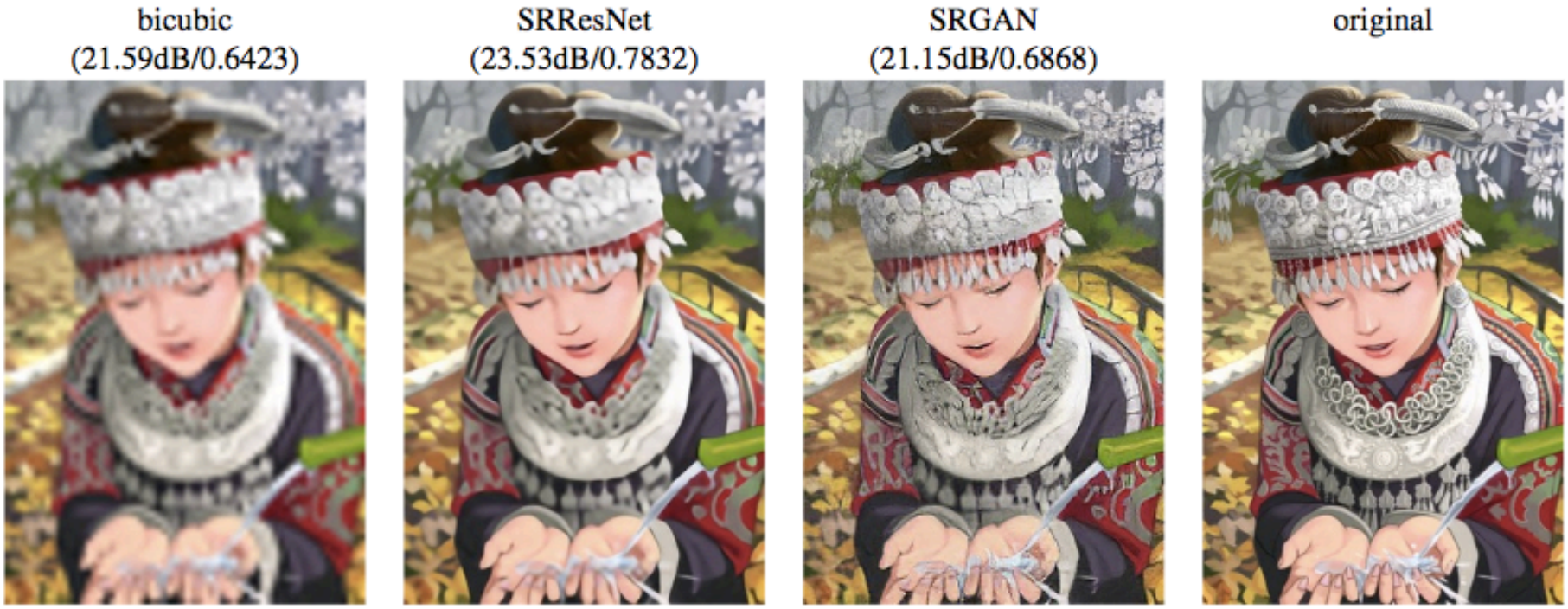


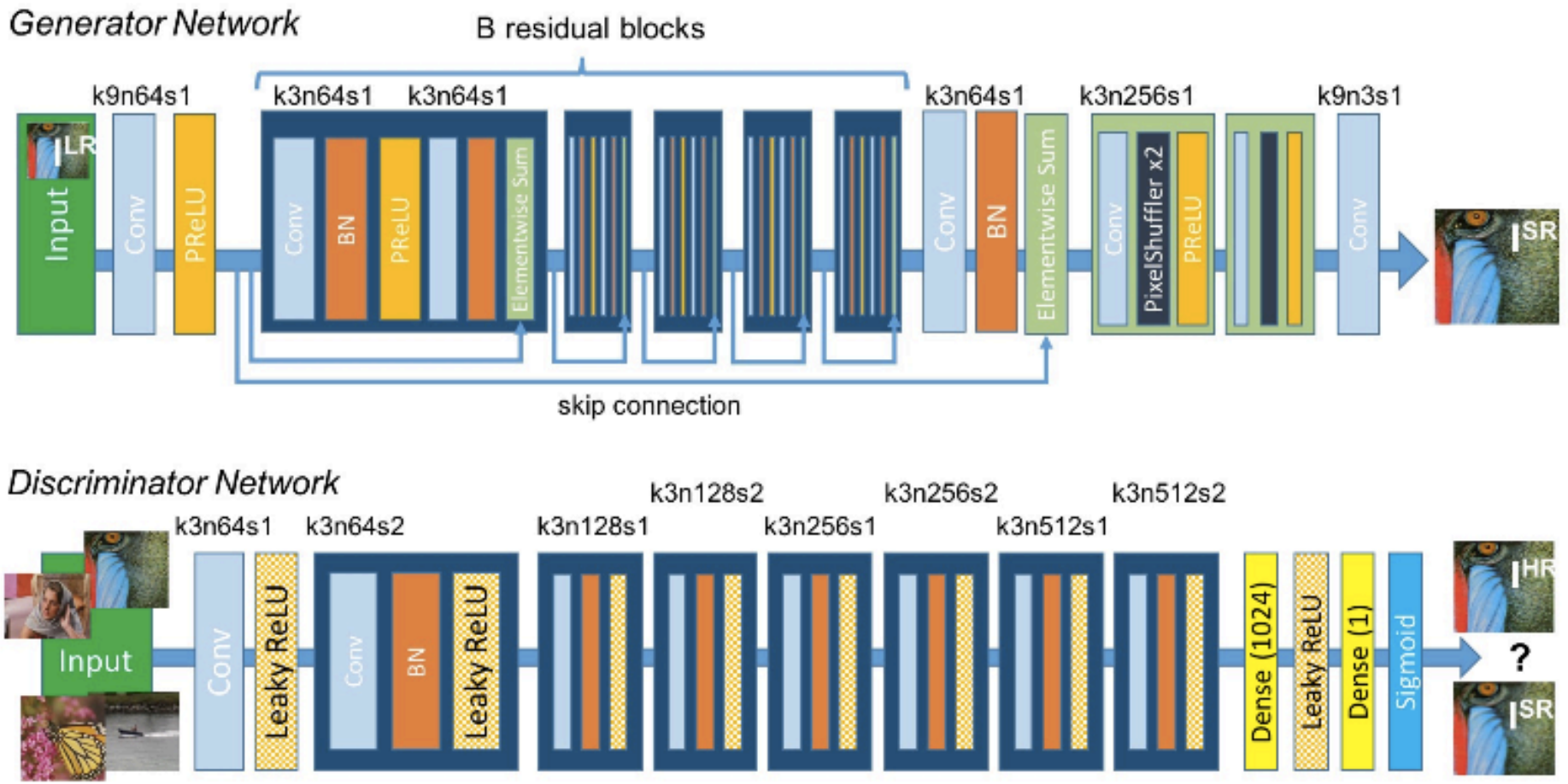


https://arxiv.org/pdf/1612.03242.pdf

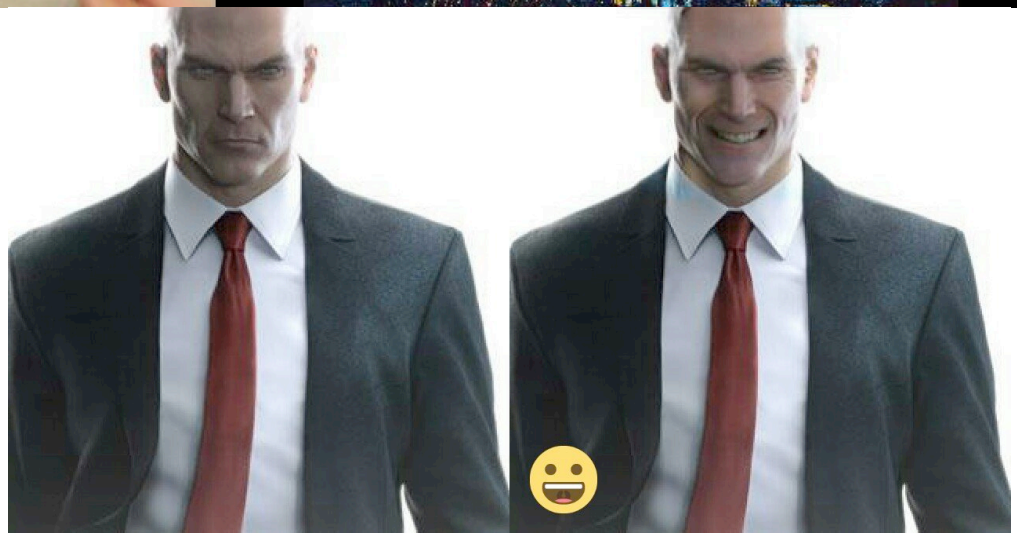
SRGAN

- https://arxiv.org/pdf/1609.04802.pdf

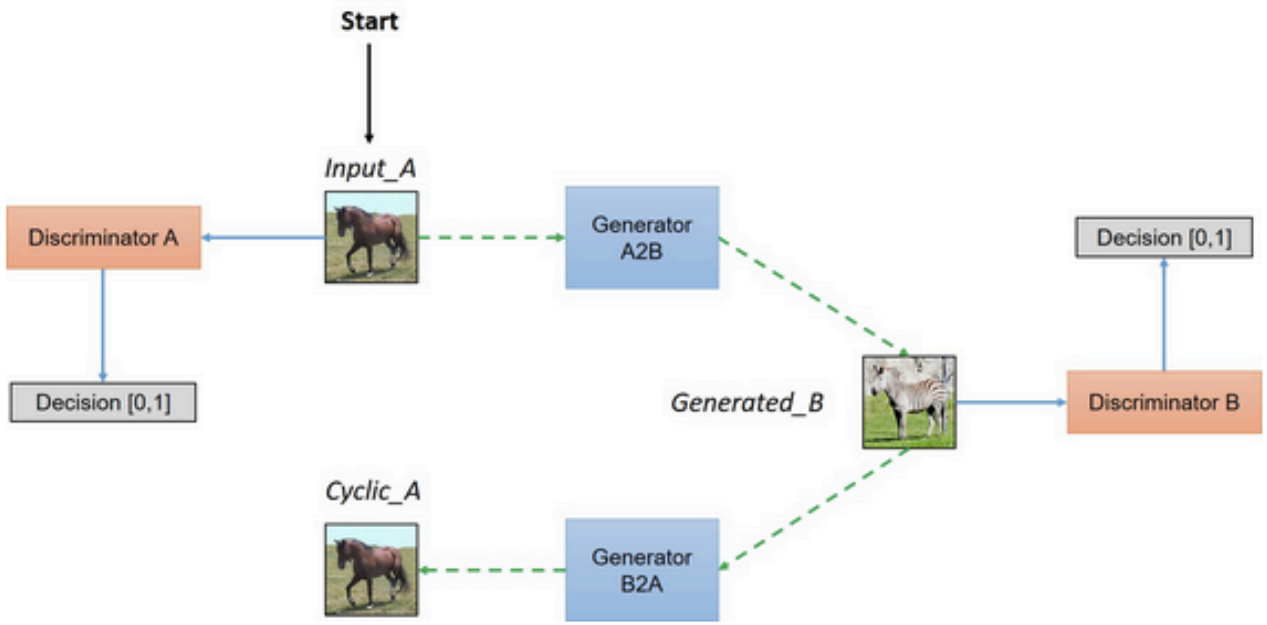



Generator Network
B residual blocks
k9n64s1
k3n64s1
k3n64s1
k3n64s1
k3n256s1
k9n3s1
LR
Input
Conv
PReLU
Conv
BN
PReLU
Elementwise Sum
Conv
BN
Elementwise Sum
Conv
PixelShuffler x2
PReLU
Conv
SR
skip connection
Discriminator Network
k3n64s1
k3n64s2
k3n128s1
k3n128s2
k3n256s1
k3n256s2
k3n512s1
k3n512s2
Input
Conv
Leaky ReLU
Conv
BN
Leaky ReLU
Dense (1024)
Leaky ReLU
Dense (1)
Sigmoid
HR
?
SR

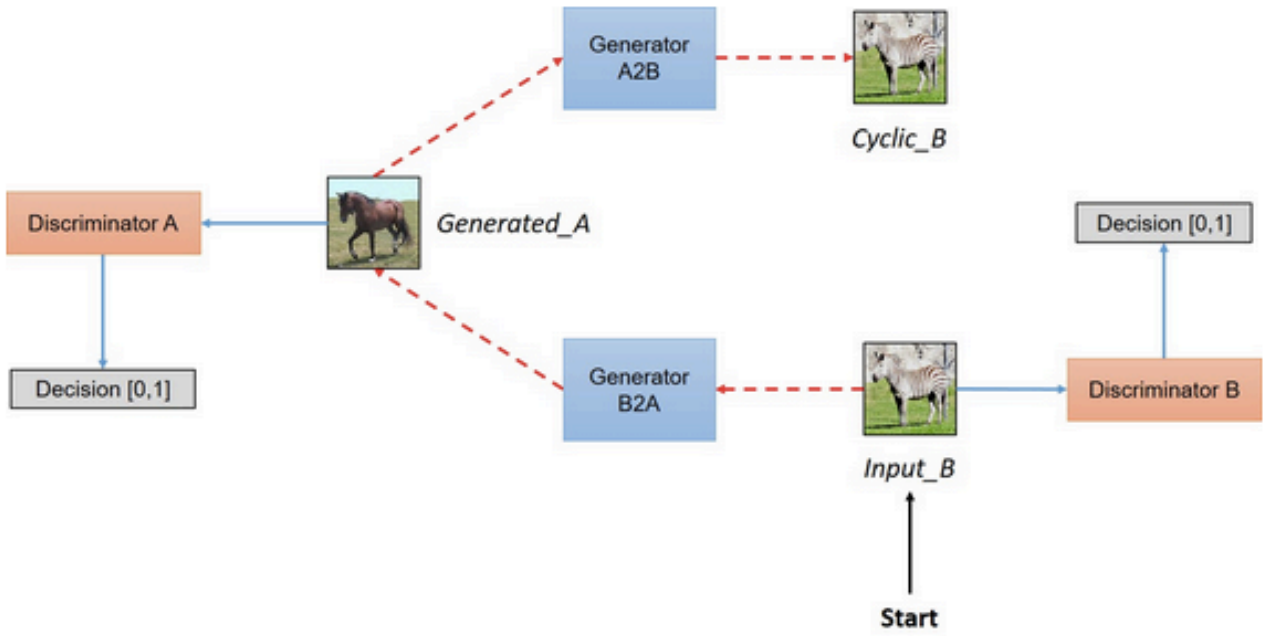
Style transfer with cycleGAN



https://hardikbansal.github.io/CycleGANBlog/



Cycle consistency



Note, you don't need a paired dataset

Machine translation with no parallel text

- MT usually requires parallel text

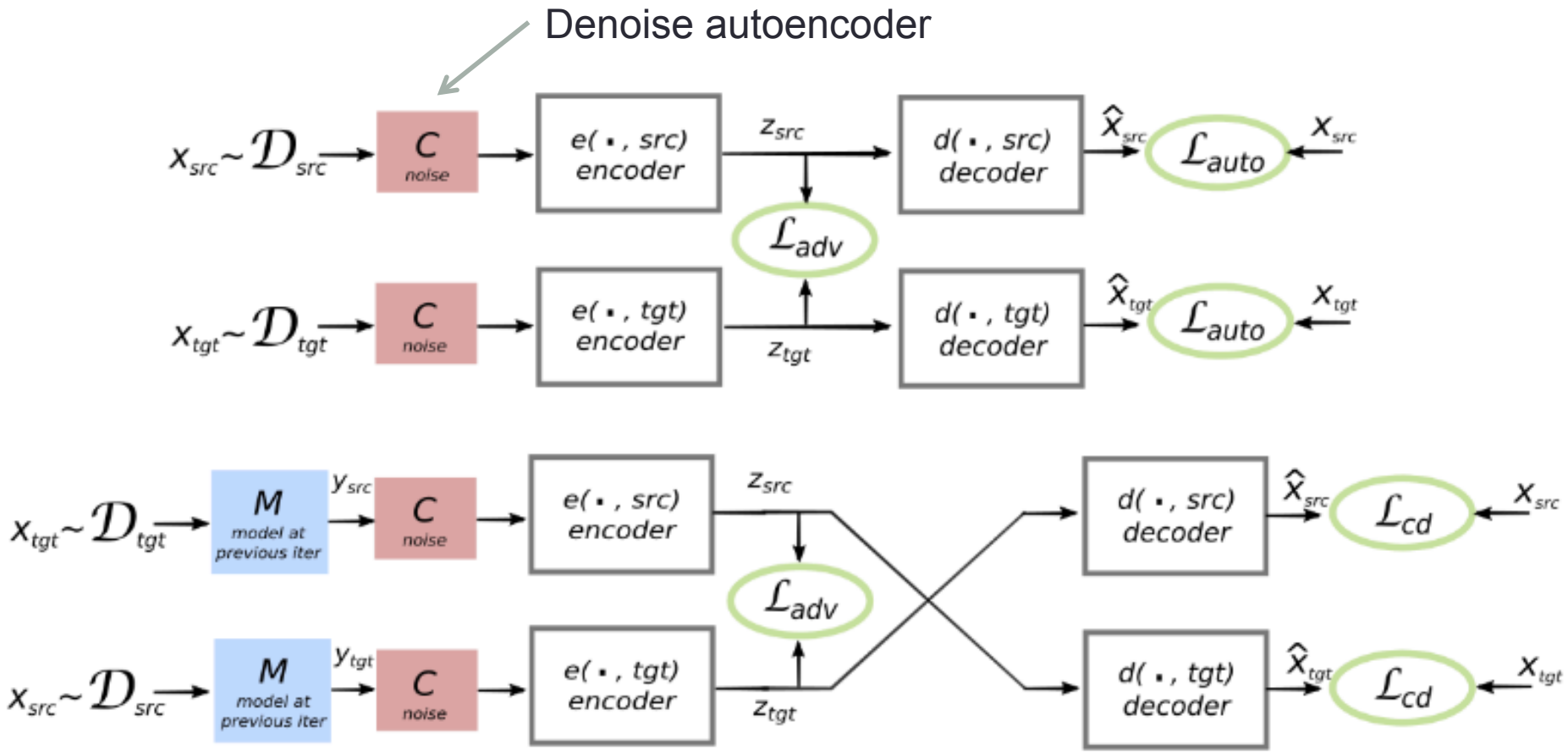
  This is a cat

  นี่คือแมว

- Most of the time we don’t have parallel text. Just text.
- Can we still do MT?
  - Use GANs + Autoencoders

https://arxiv.org/abs/1711.00043

Use GAN Loss (L_{adv}) to enforce that source and target language pairs share the same distributions.
Then enforce the translation distribution matches.



https://arxiv.org/abs/1711.00043

Results

	Multi30k-Task1				WMT			
	en-fr	fr-en	de-en	en-de	en-fr	fr-en	de-en	en-de
Supervised	56.83	50.77	38.38	35.16	27.97	26.13	25.61	21.33
word-by-word	8.54	16.77	15.72	5.39	6.28	10.09	10.77	7.06
word reordering	-	-	-	-	6.68	11.69	10.84	6.70
oracle word reordering	11.62	24.88	18.27	6.79	10.12	20.64	19.42	11.57
Our model: 1st iteration	27.48	28.07	23.69	19.32	12.10	11.79	11.10	8.86
Our model: 2nd iteration	31.72	30.49	24.73	21.16	14.42	13.49	13.25	9.75
Our model: 3rd iteration	32.76	32.07	26.26	22.74	15.05	14.31	13.33	9.64

https://arxiv.org/abs/1711.00043

Summary

- Various ways of clustering
  - K-means, GMM, LDA
- Encoders
  - Denoising, VAE
  - Generate nice representations of the input
- GAN
  - A neural network that we can sample (generate) data
  - Discriminator vs Generator

Reminder

- Mid report due tomorrow
- Meeting Thursday

- Next class: practical issues in machine learning application deployments, e.g. more various random stuff