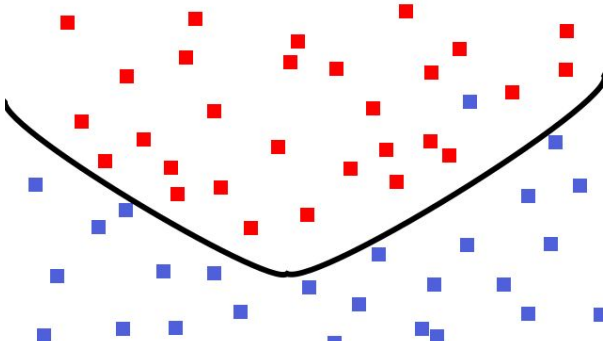
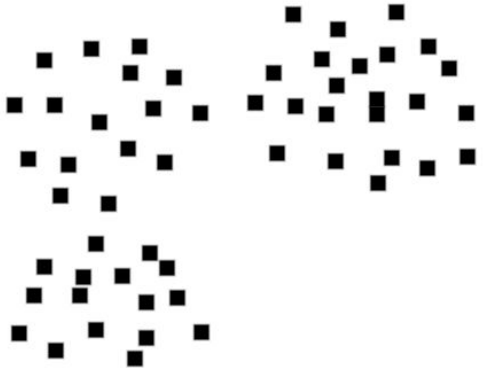
UNSUPERVISED LEARNING

3 Modes of Learning

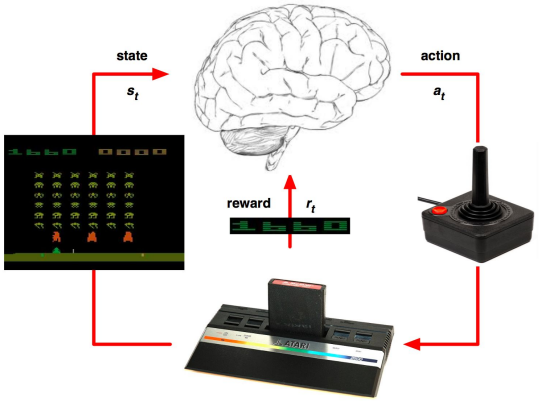


Supervised Learning

Reinforcement Learning

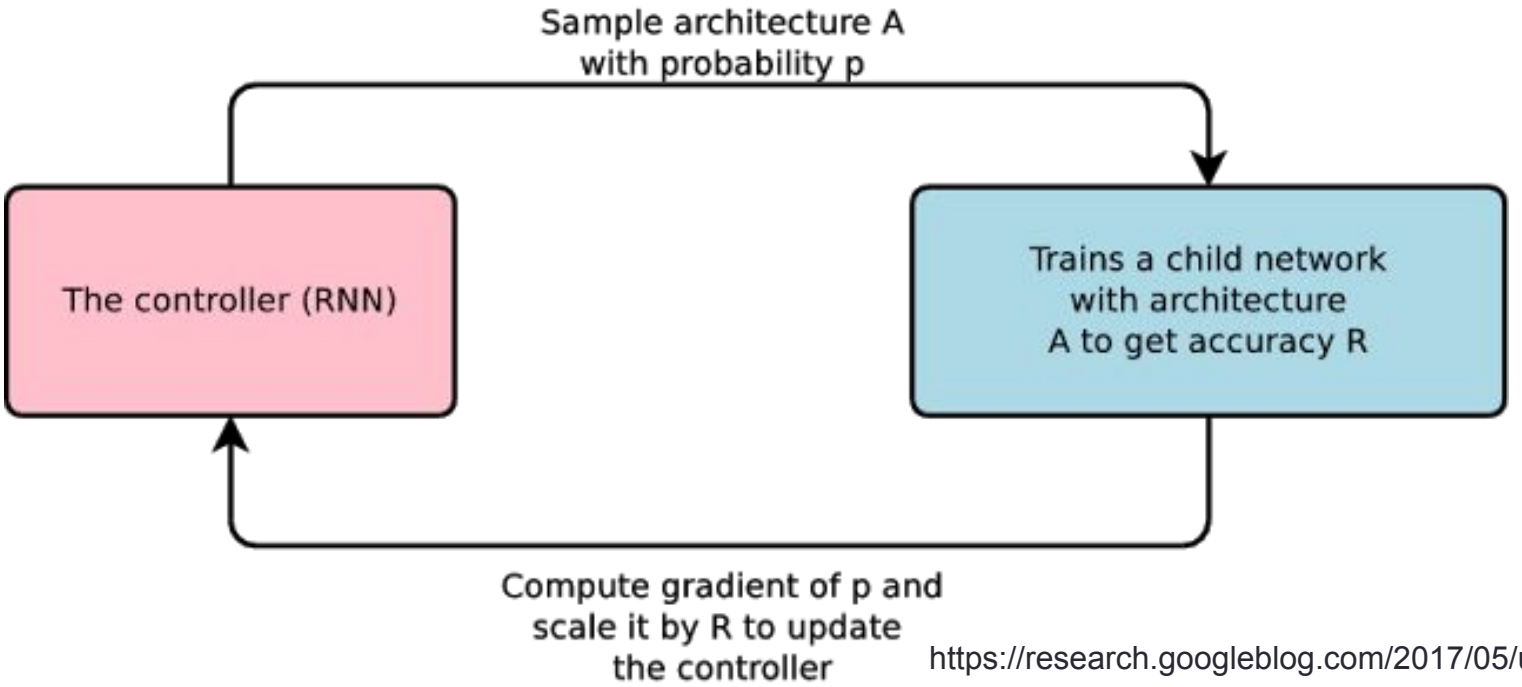


Unsupervised Learning



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 lowering variance
- 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

How Much Information Does the Machine Need to Predict?

Y LeCun

- "Pure" Reinforcement Learning (cherry)
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- Supervised Learning (icing)
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- Unsupervised/Predictive Learning (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

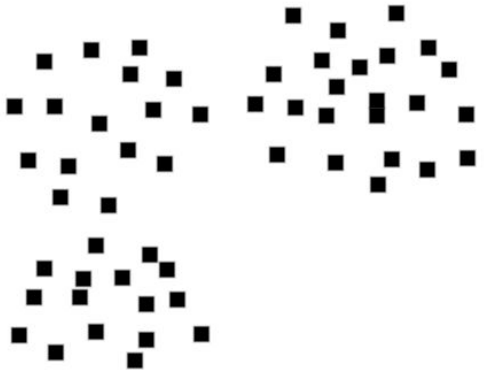
- (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

Unsupervised Learning

- Clustering
  - Introduction to probabilistic graphical models
  - LDA
- Autoencoders
  - Denoising Autoencoder
  - VAE
    - Evidence lower bound
  - Perceptual loss
  - Image prior
- Generative Adversarial Networks
- Random Cool stuff

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

Can we give priors knowledge to clustering?

Latent Dirichlet Allocation (LDA)

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

  - A kind of probabilistic graphical model

- Let's cover some basics

Topic modeling motivation

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

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

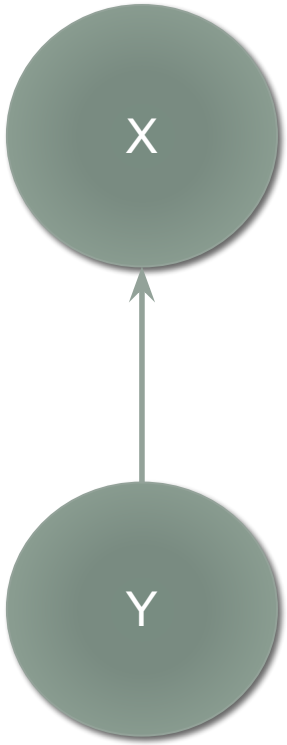
- P(X_1 = Rosses, X_2 = are, X_3 = red, …)
  - This probability distribution is intractable

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

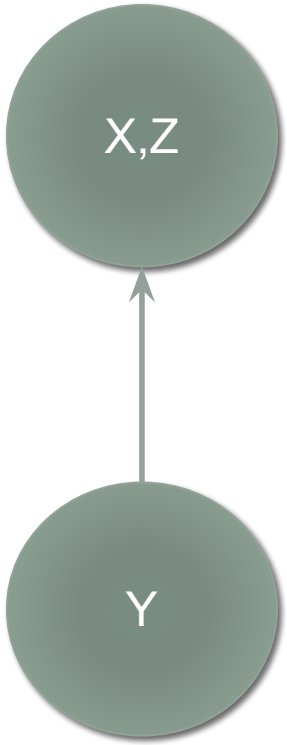


Curse of dimensionality

- Given rain in the morning predict sunny or cloudy and windy or no wind 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

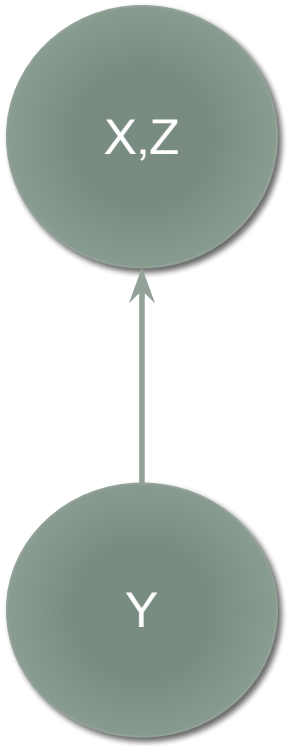


Curse of dimensionality

- 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



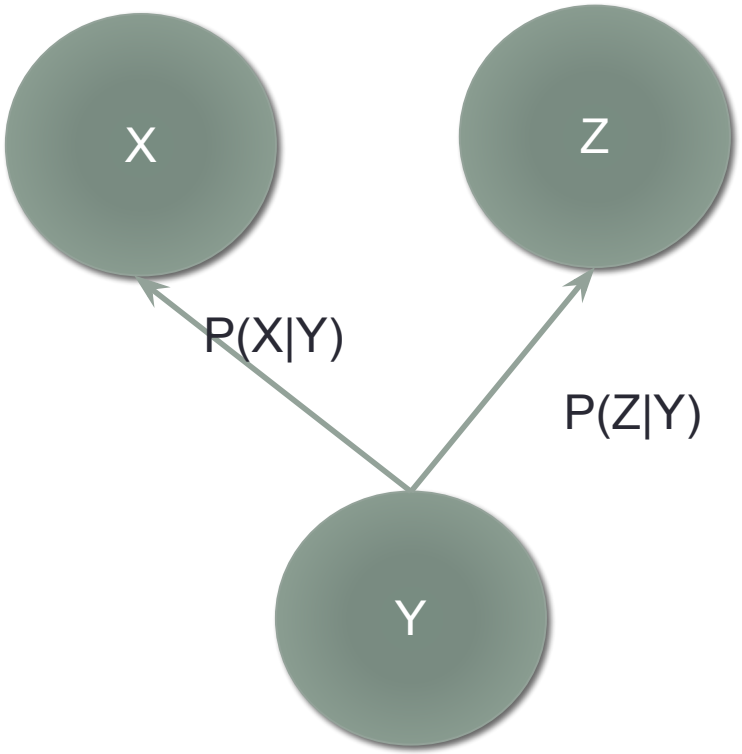
Naïve Bayes

- 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

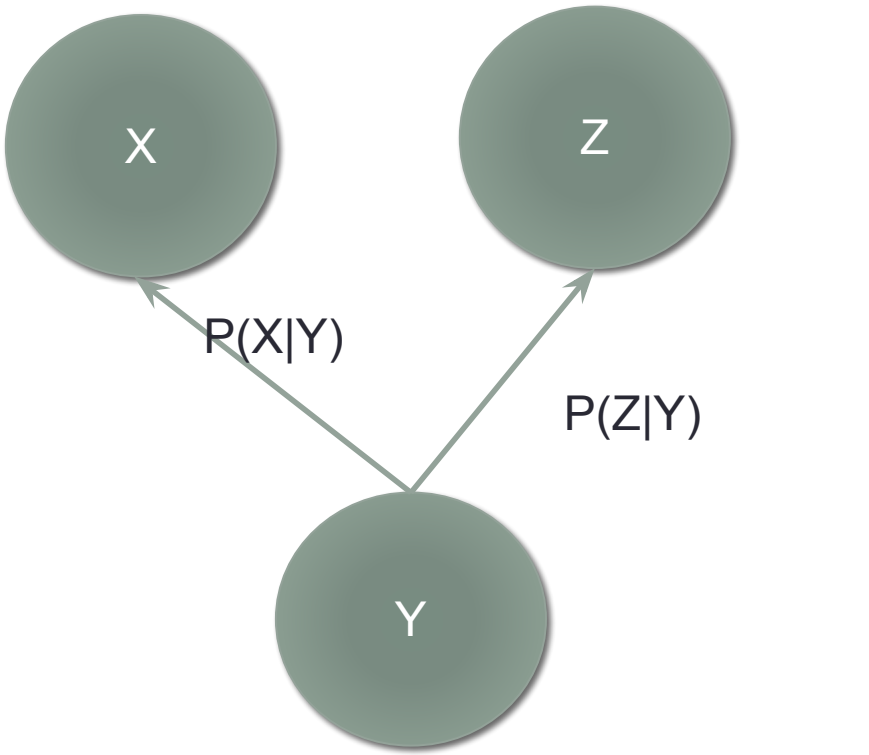


Probabilistic models and independence assumptions

- 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

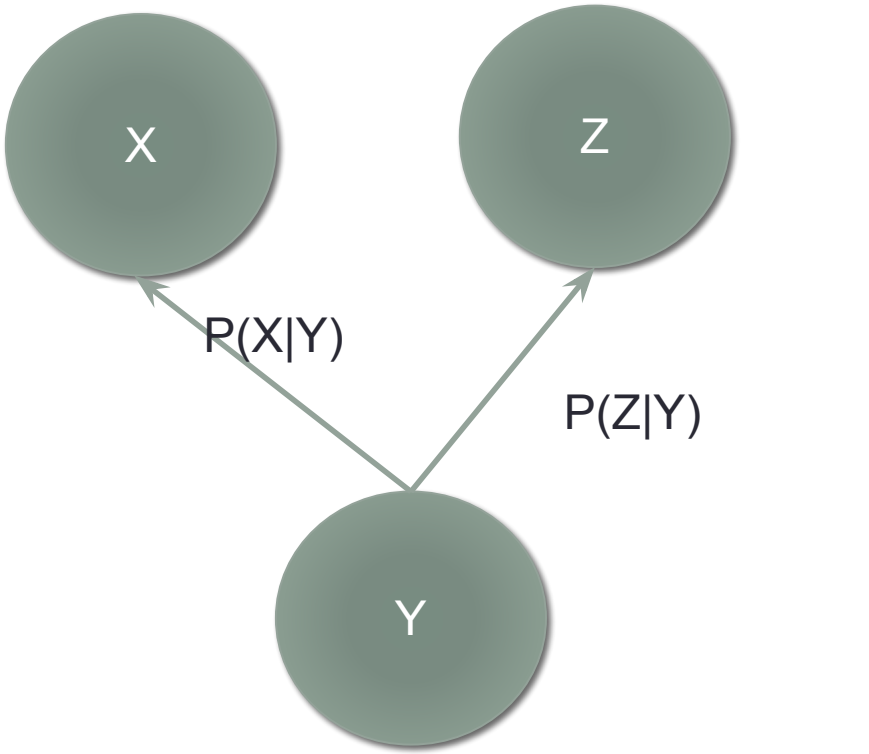


Plate notation

- Sometimes you have too many relationships.

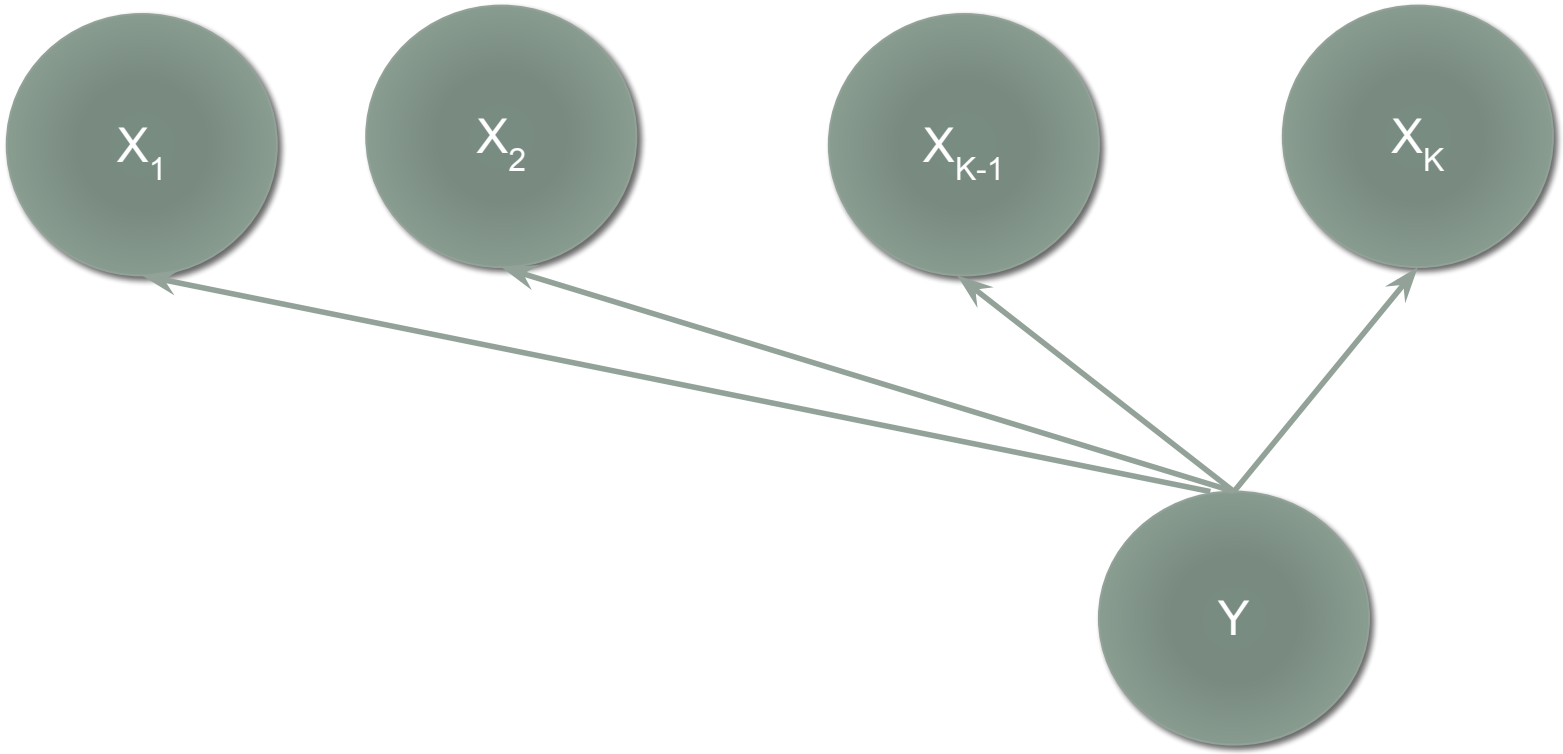


Plate notation

- Summarize by using a square box with number

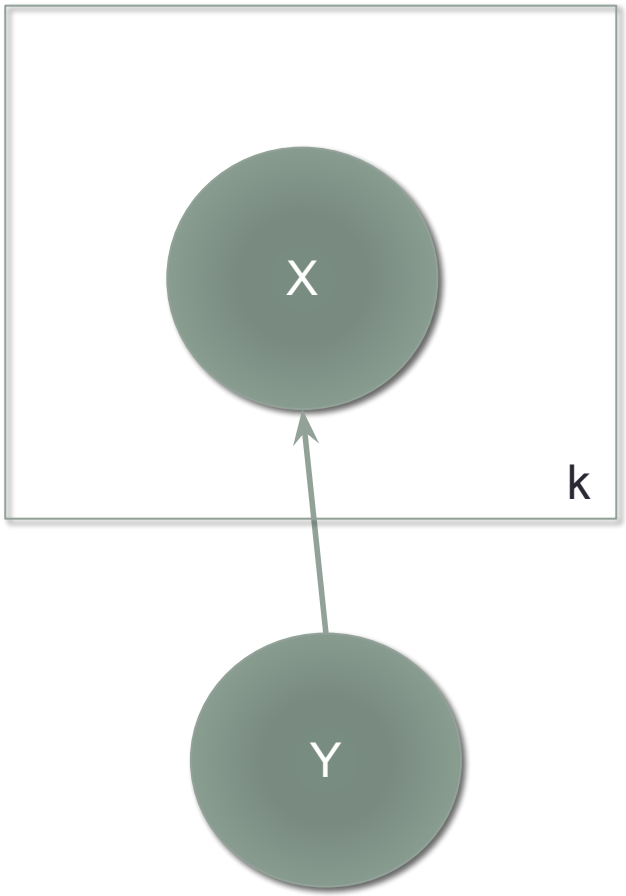
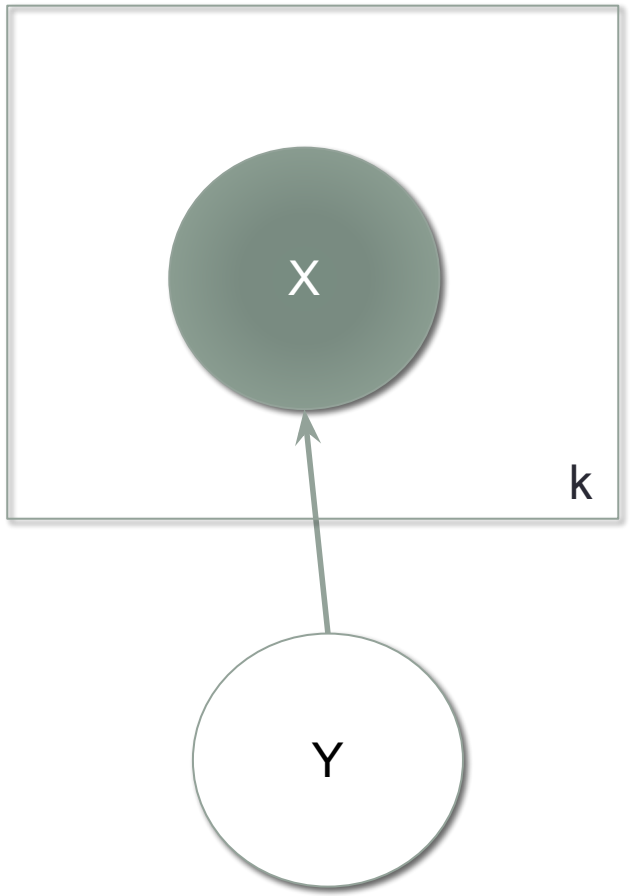


Plate notation

- Summarize by using a square box with number
- Dark circles are observed variables
- White circles are **latent variables**



Topic modeling motivation

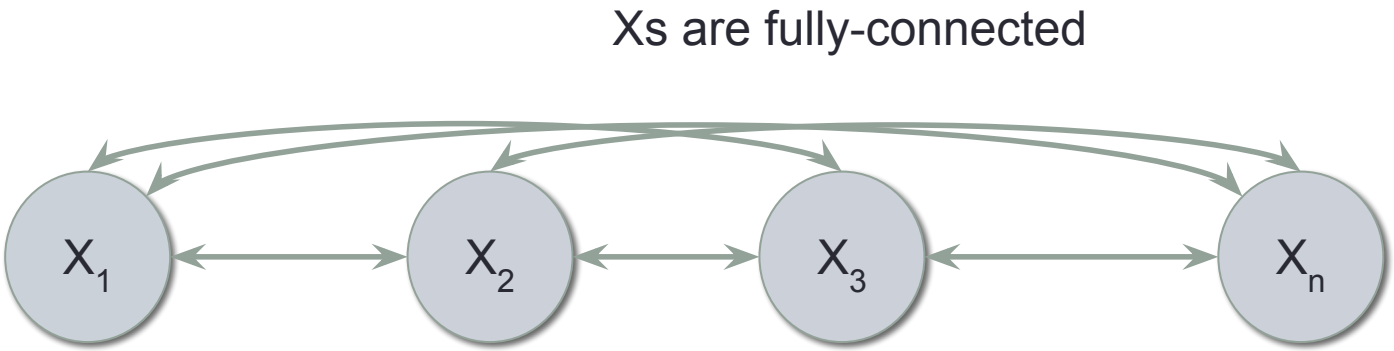
- I want to be able to give probabilities to words/sentences/documents.

- $P(X_1 = \text{Rosses}, X_2 = \text{are}, X_3 = \text{red}, \ldots)$
  - This probability distribution is intractable

- Assume independence
  - $P(X_1)\ P(X_2)\ P(X_3)\ldots$
  - Lost all relational structure. Can we do better?
    - Assume words come from topics. How?

Graphical view of language modeling

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

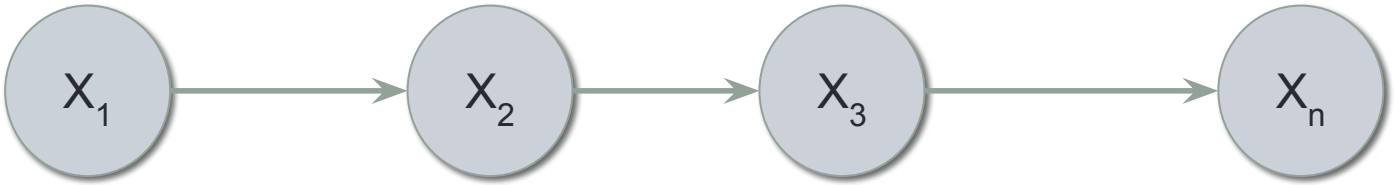


$P(X_1 = \text{Roses}, X_2 = \text{are}, \ldots, X_n = \text{blue})$

Graphical view of language modeling

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

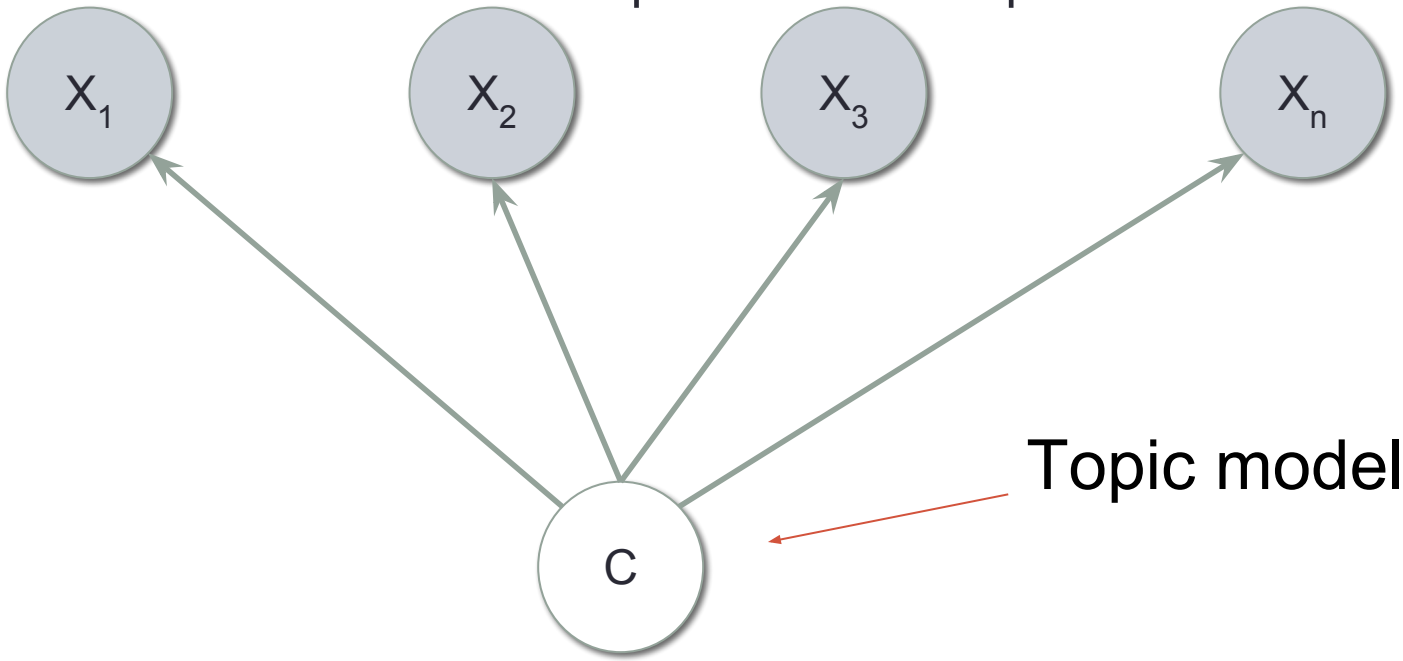
With markov assumption



$P(X_1 = \text{Roses})P(X_2 = \text{are} \mid X_1 = \text{Roses})\ P(X_3 = \text{red}) \ldots P(X_n = \text{blue} \mid X_{n-1} = \text{are})$

Graphical view of language modeling

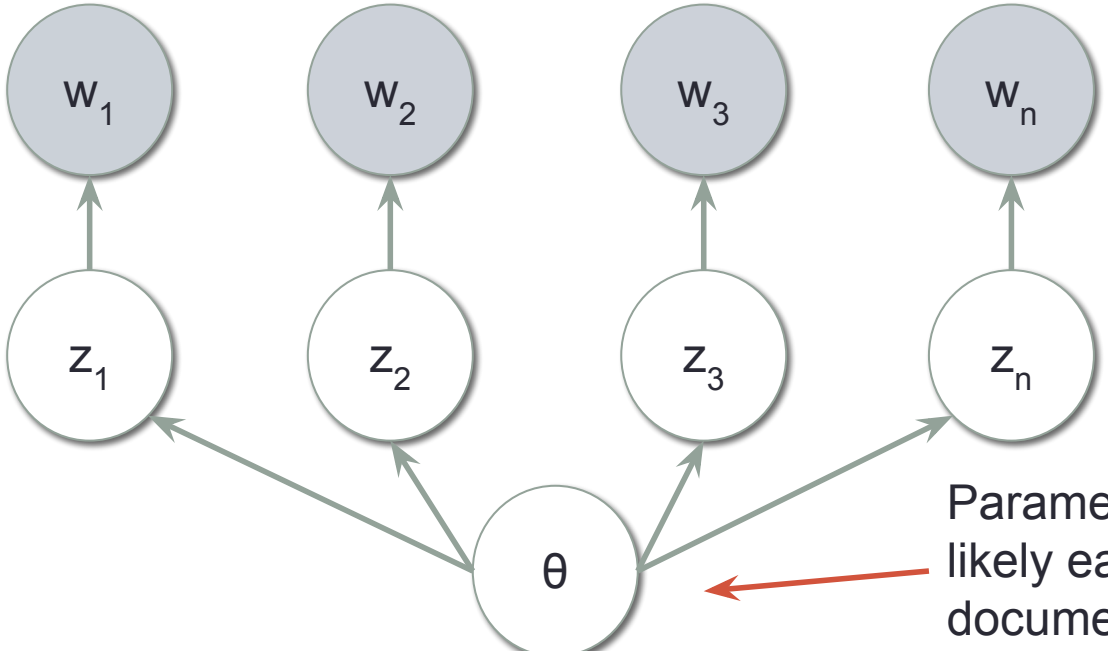
- P("Roses are red. Violets are blue.") = ?
- Let's assume words are generated from a topic
  - Sentence comes from topic C = Shakespeare



$P(X_1 = \text{Roses} \mid c = \text{Shakespeare}) P(X_2 = \text{are} \mid c = \text{Shakespeare}) \ldots$
$P(X_n = \text{blue} \mid c = \text{Shakespeare}) P(c = \text{Shakespeare})$

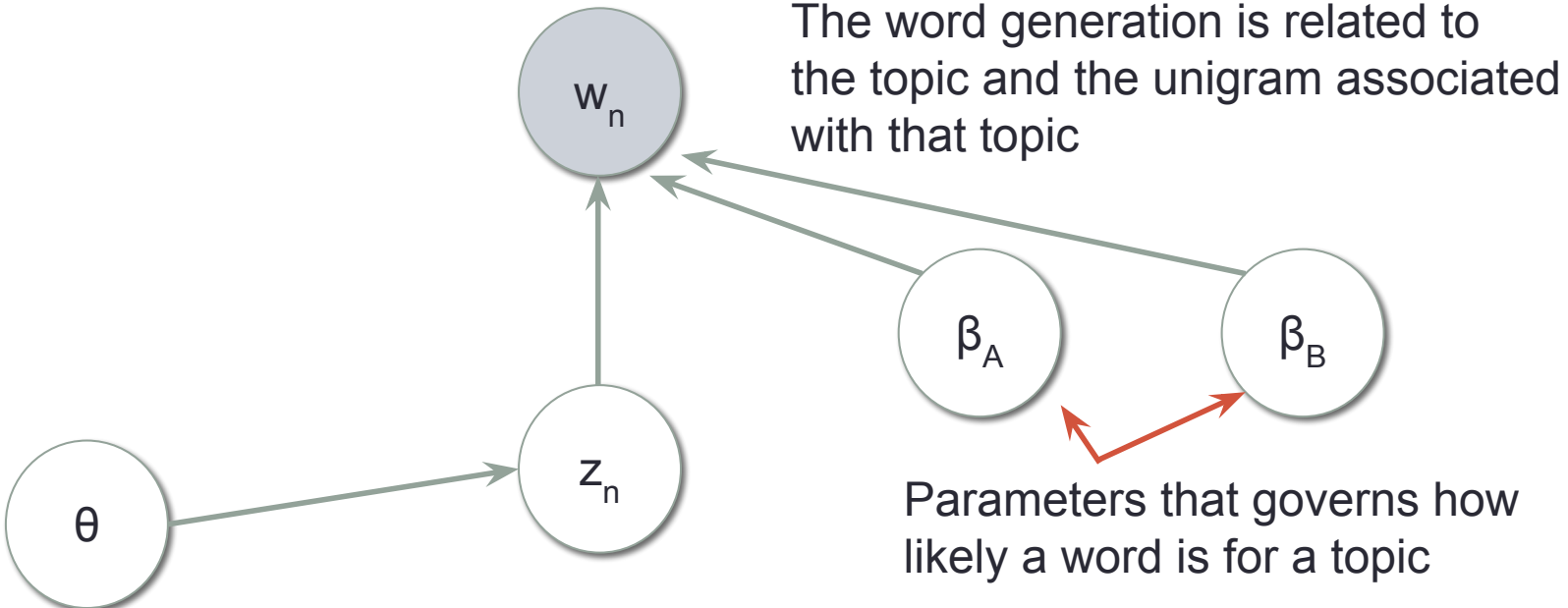
Multiple topics

- Most document have multiple topics. Our model assumes 1 document 1 topic.
  - Let a document be a mixture of topics. Each word has its own topic, z.
    - P(w) = P(z = A) P(w | z = A) + P(z = B) P(w | z = B)
    - P(z = A) + P(z = B) = 1, θ = P(z = A)



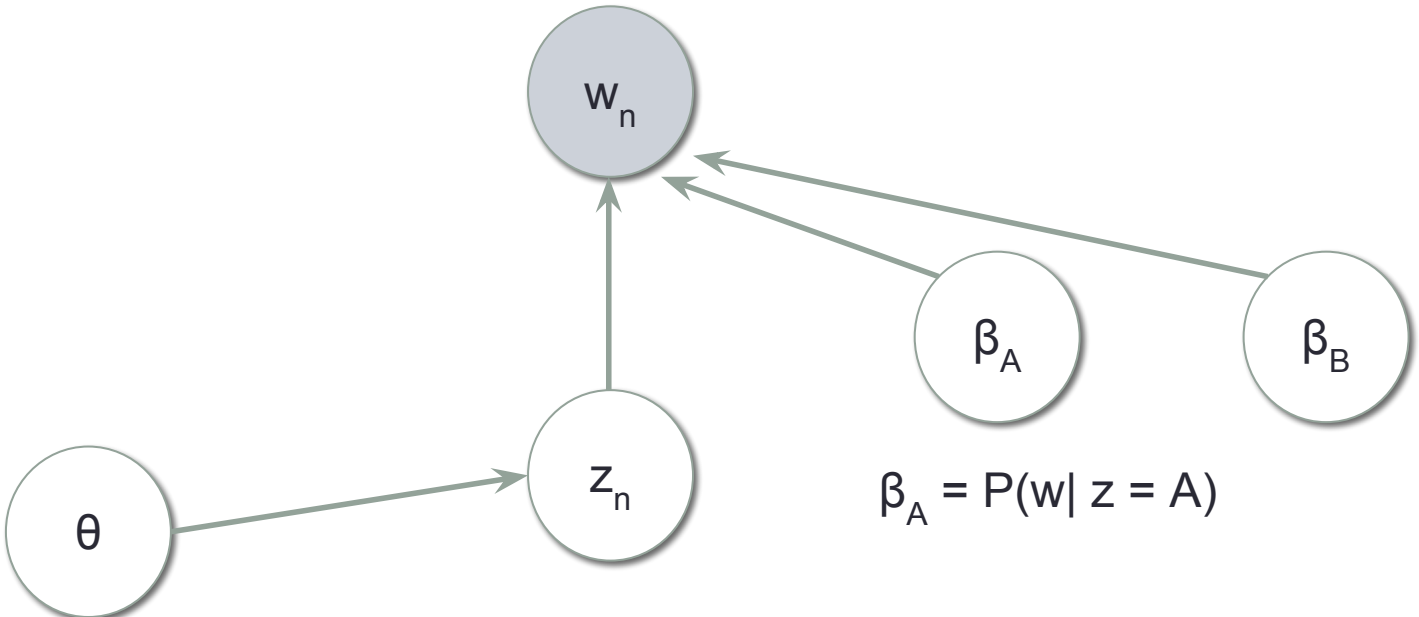
Topic modeling

- Most document have multiple topics. Our model assumes 1 document 1 topic.
  - Let a document be a mixture of topics (language model interpolation). Each word has its own topic, z.
    - P(w) = P(z = A) P(w | z = A) + P(z = B) P(w | z = B)
    - P(z = A) + P(z = B) = 1, $\theta = P(z = A)$ $\beta_A = P(w| z = A)$



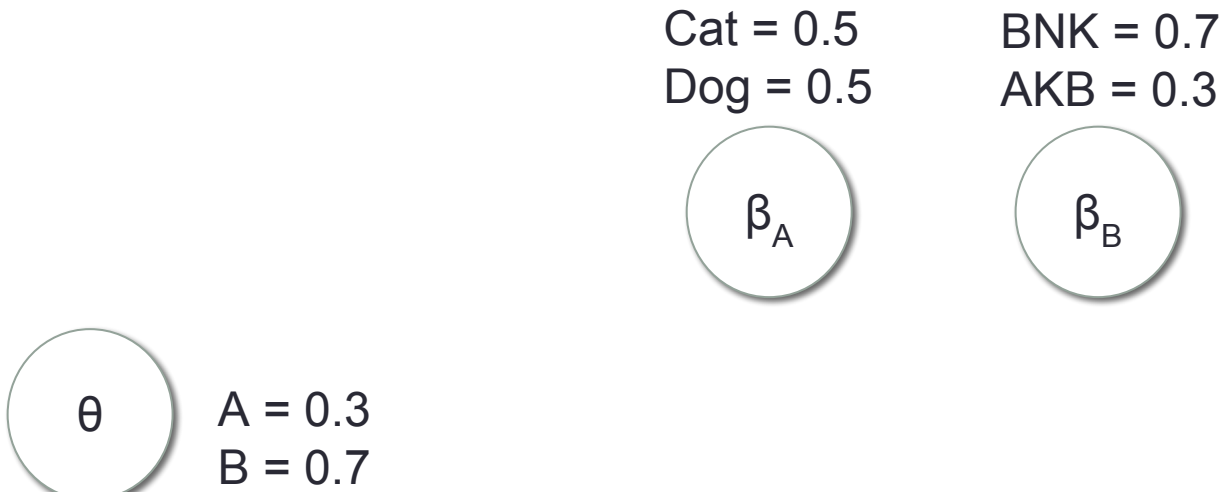
Graphical model and generation

- You can generate a sample from a graphical model by following the arrows



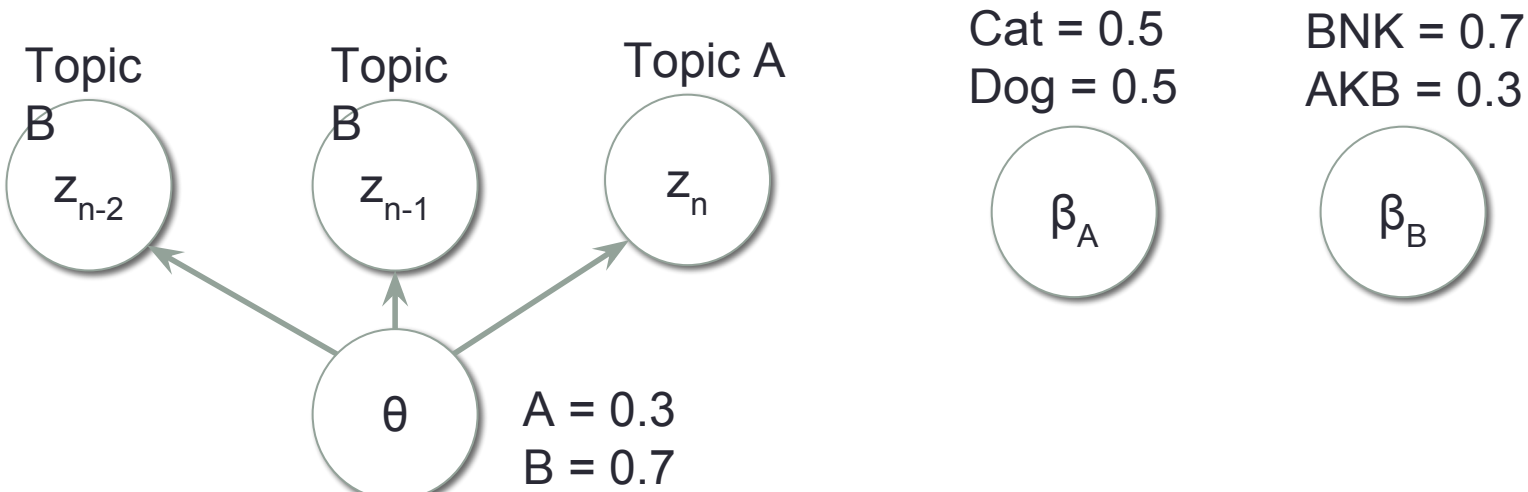
Graphical model and generation

- Given θ, β_A, β_B



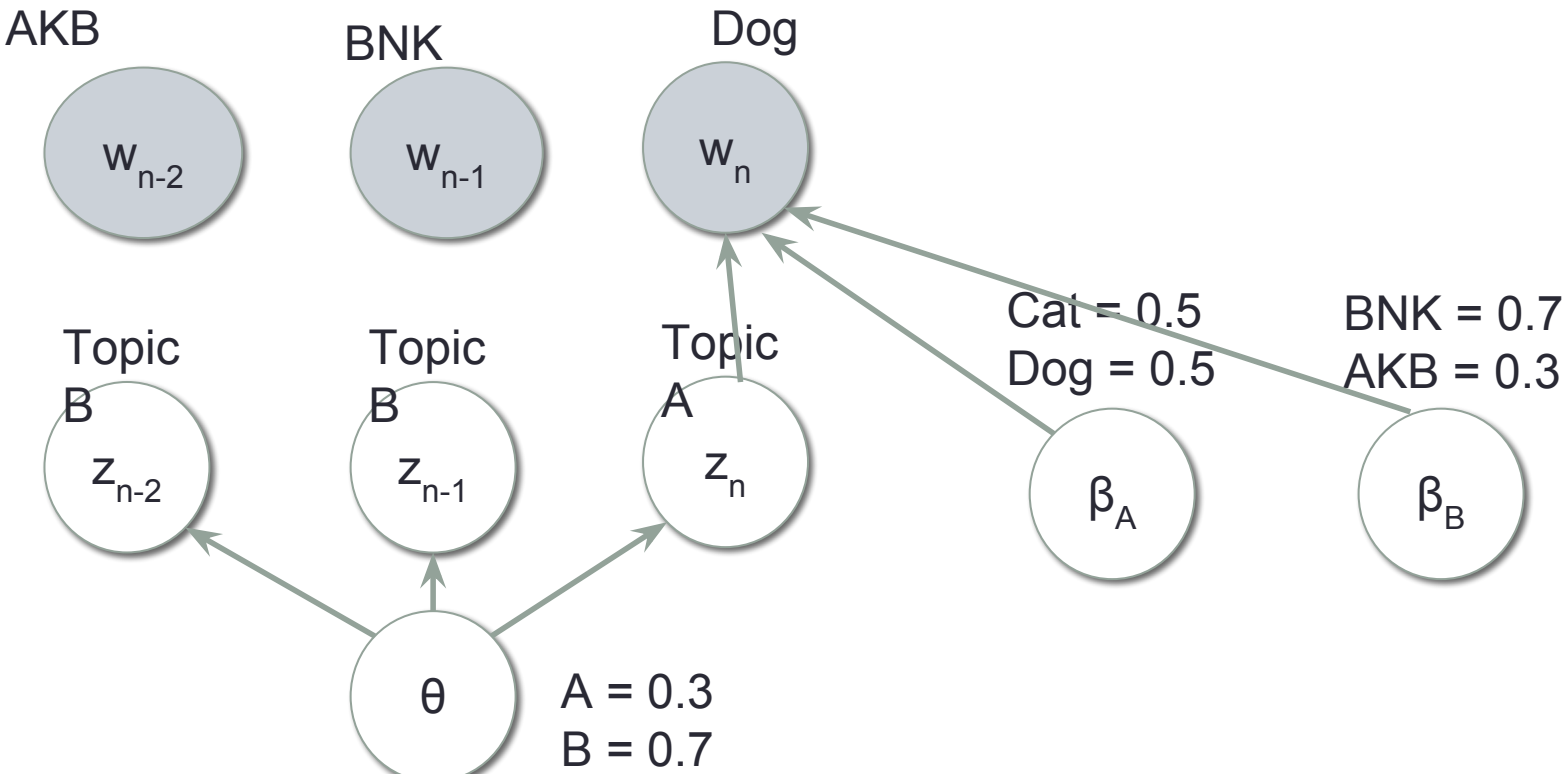
Graphical model and generation

- Given θ, β_A, β_B
- Generate (randomly create) z from θ



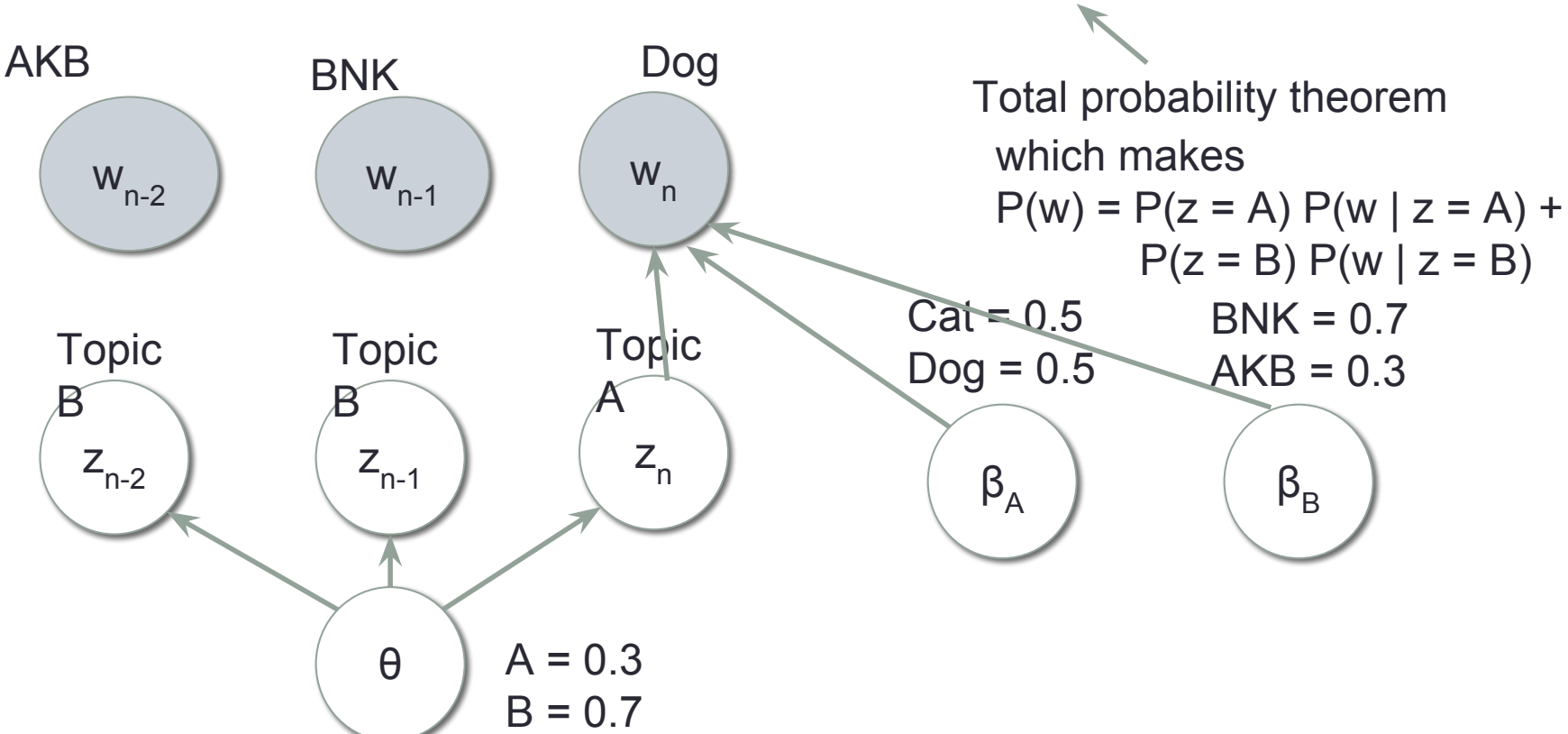
Graphical model and generation

- Given θ, β_A, β_B
- Generate (randomly create) z from θ
- Generate w_n from z_n β_A, β_B

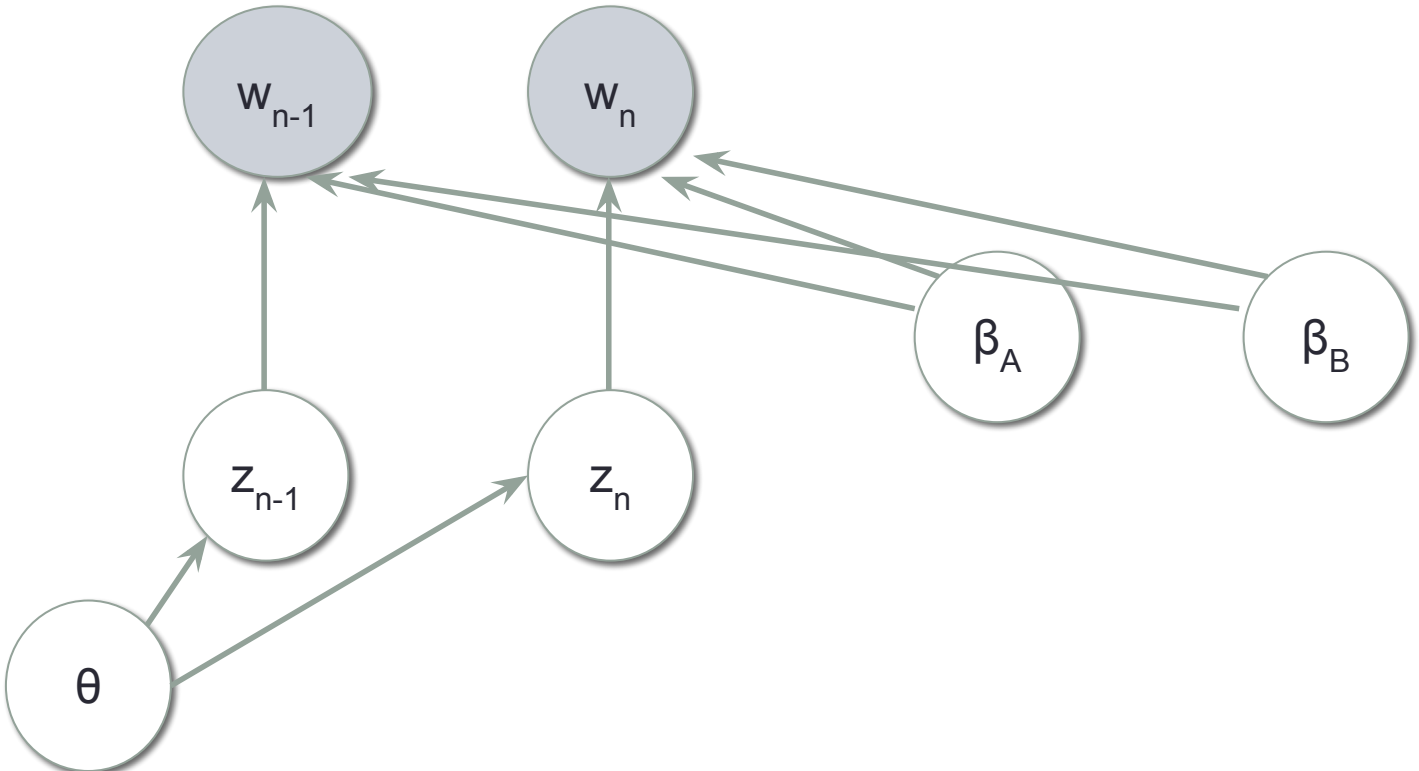


Graphical model and generation

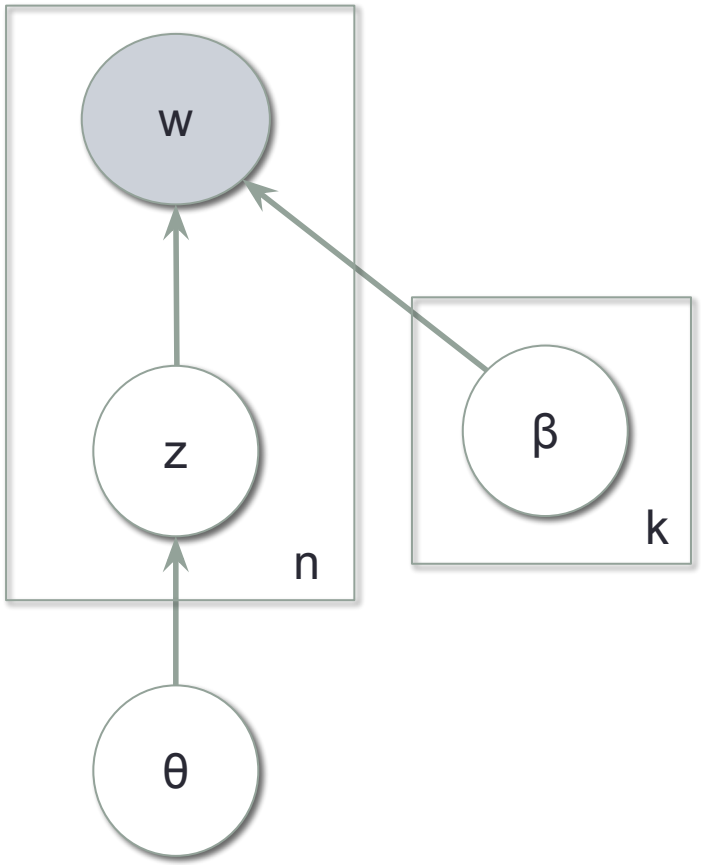
How likely a sentence is likely to be generated follows this generation process
P(AKB,BNK,Dog,B,B,A) = P(B)P(B)P(A)P(AKB|B)P(BNK|B)P(Dog|A)
Note
P(AKB,BNK,Dog) = P(AKB,BNK,Dog,A,A,A) + P(AKB,BNK,Dog,A,A,B)
P(AKB,BNK,Dog,A,B,A) + P(AKB,BNK,Dog,A,B,B) +...



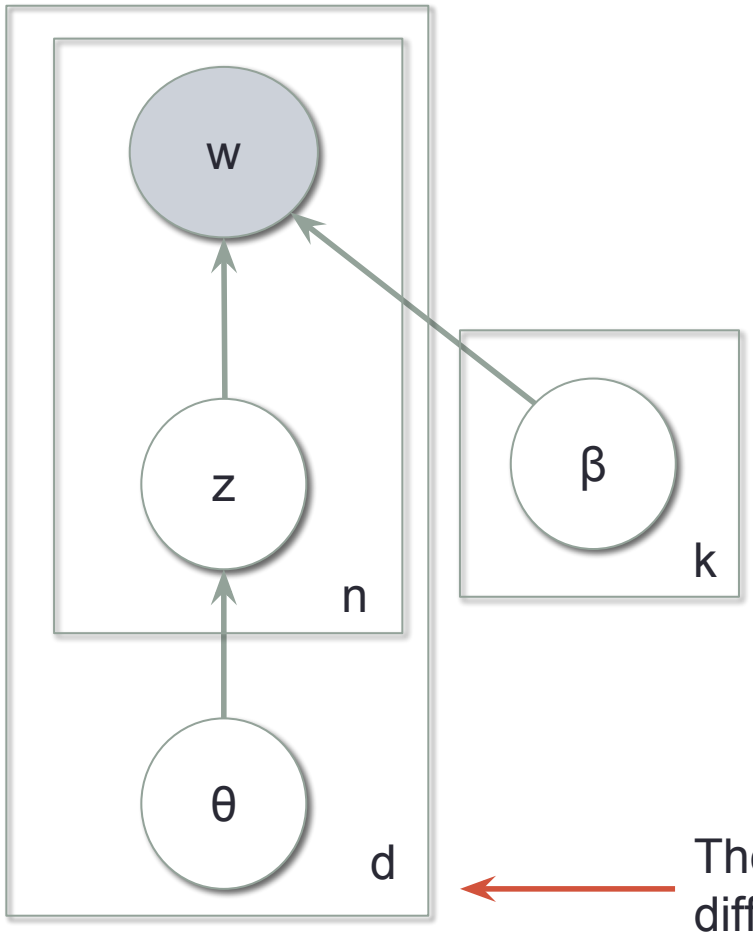
Topic modeling with plate notation



Topic modeling with plate notation

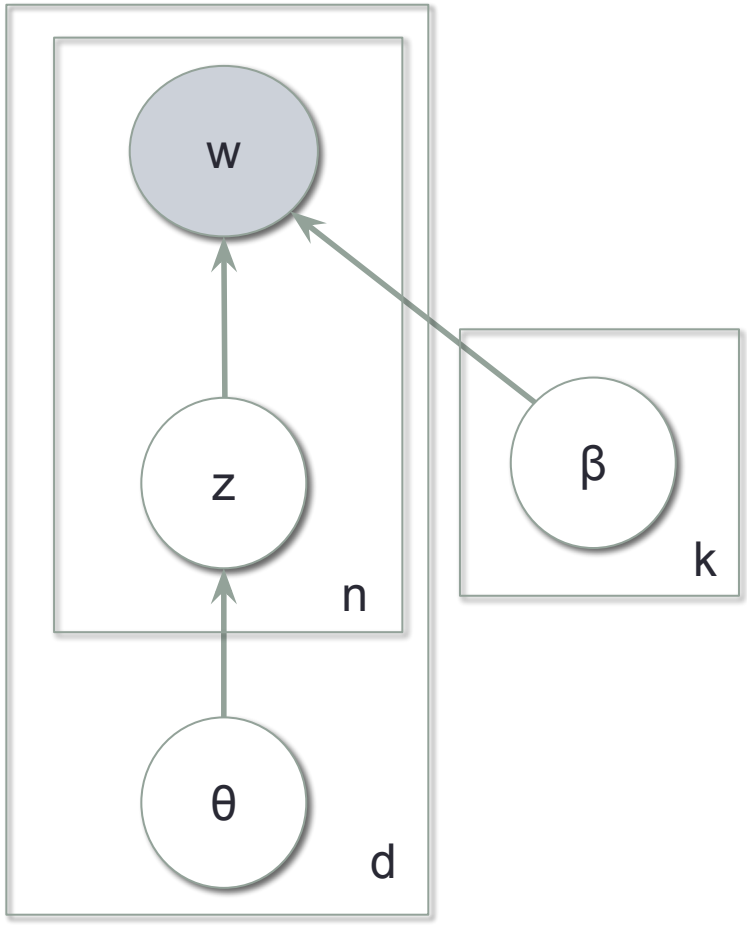


Topic modeling with plate notation



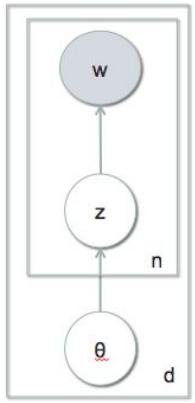
There are d documents each with different topic distributions

Topic modeling with plate notation



Called pLSA
probabilistic Latent Semantic Analysis

Note: if you look at other textbooks, you will see a slightly different picture



Learning topic latent model parameters

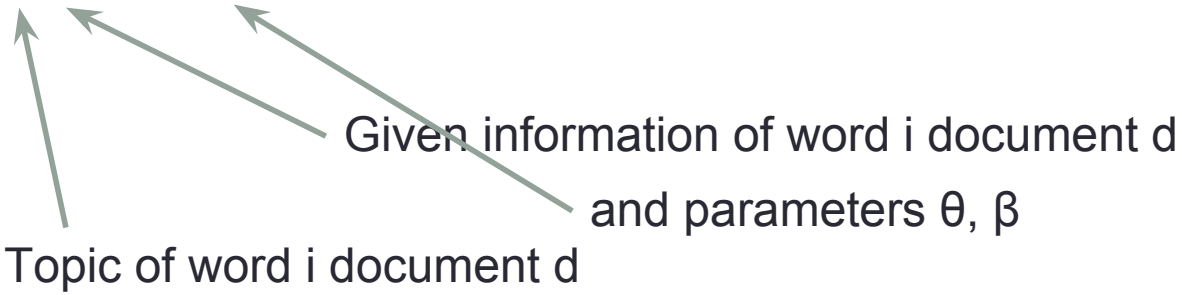
- How to find θ and β?
  - “Cat,Dog,BNK”, “Cat, cat, cat, BNK”, “Dog, dog, BNK, dog, AKB”
- If we know, the latent topic z for each word we can use the counts
  - “Cat_A, Dog_A, BNK_B”
- $P(Cat|A) = \frac{count(Cat\_A)}{count(A)}$
- $P_1(A) = \frac{count(A)}{count(\text{all words in document 1})}$
- But we don’t know the topic z for each word

Expectation maximization (EM)

- A method to iteratively maximize the likelihood of a model on training data
  - Expectation step (E step) – guess latent variables from model parameters (get soft counts)
  - Maximization step (M step) – re-estimate model parameters from latent variables (counts)

E-step

- Find an estimate for the latent variable given parameters θ, β
- $p(z_{di} \mid w_{di}, \theta, \beta)$



E-step

- Find an estimate for the latent variable given parameters θ, β

$$p(z_{di}|w_{di},\theta,\beta) = \frac{p(z_{di},w_{di},\theta,\beta)}{\sum_{z'=1}^{k} p(z'_{di},w_{di},\theta,\beta)}$$

p(w, θ, β)

E-step

- Find an estimate for the latent variable given parameters θ, β

$$p(z_{di}|w_{di},\theta,\beta) = \frac{p(z_{di},w_{di},\theta,\beta)}{\sum_{z'=1}^{k} p(z'_{di},w_{di},\theta,\beta)}$$

$$= \frac{\theta_{z|d}\beta_{w|z}}{\sum_{z'=1}^{k} \theta_{z'|d}\beta_{w|z'}}$$

Index di for z and w dropped for clarity

E-step

- Find an estimate for the latent variable given parameters θ, β

$$p(z_{di}|w_{di},\theta,\beta) = \frac{p(z_{di},w_{di},\theta,\beta)}{\sum_{z'=1}^{k} p(z'_{di},w_{di},\theta,\beta)}$$

$$= \frac{\theta_{z|d}\beta_{w|z}}{\sum_{z'=1}^{k} \theta_{z'|d}\beta_{w|z'}}$$

P(Word is from topic A| word is cat from document 1)
Probability that the word is from each topic. Use as counts

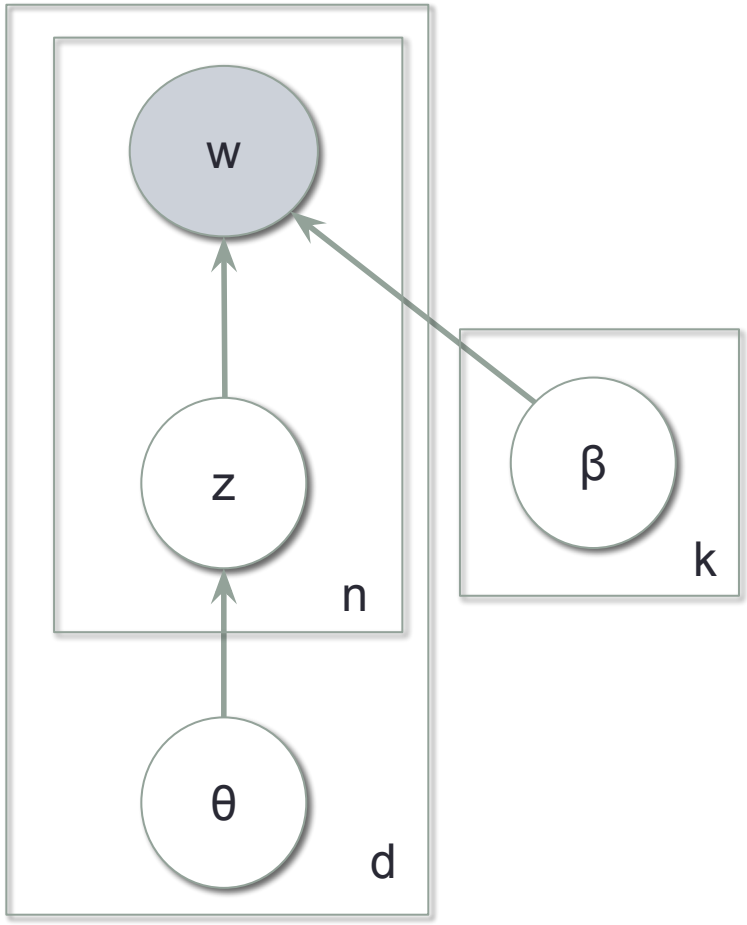
M-Step

- Instead of real counts use $P(z_{di})$ as the topic label
  - $P(Cat|A) = \frac{count(Cat\_A)}{count(A)}$
  - $P_1(A) = \frac{count(A)}{count(all\ words\ in\ document\ 1)}$

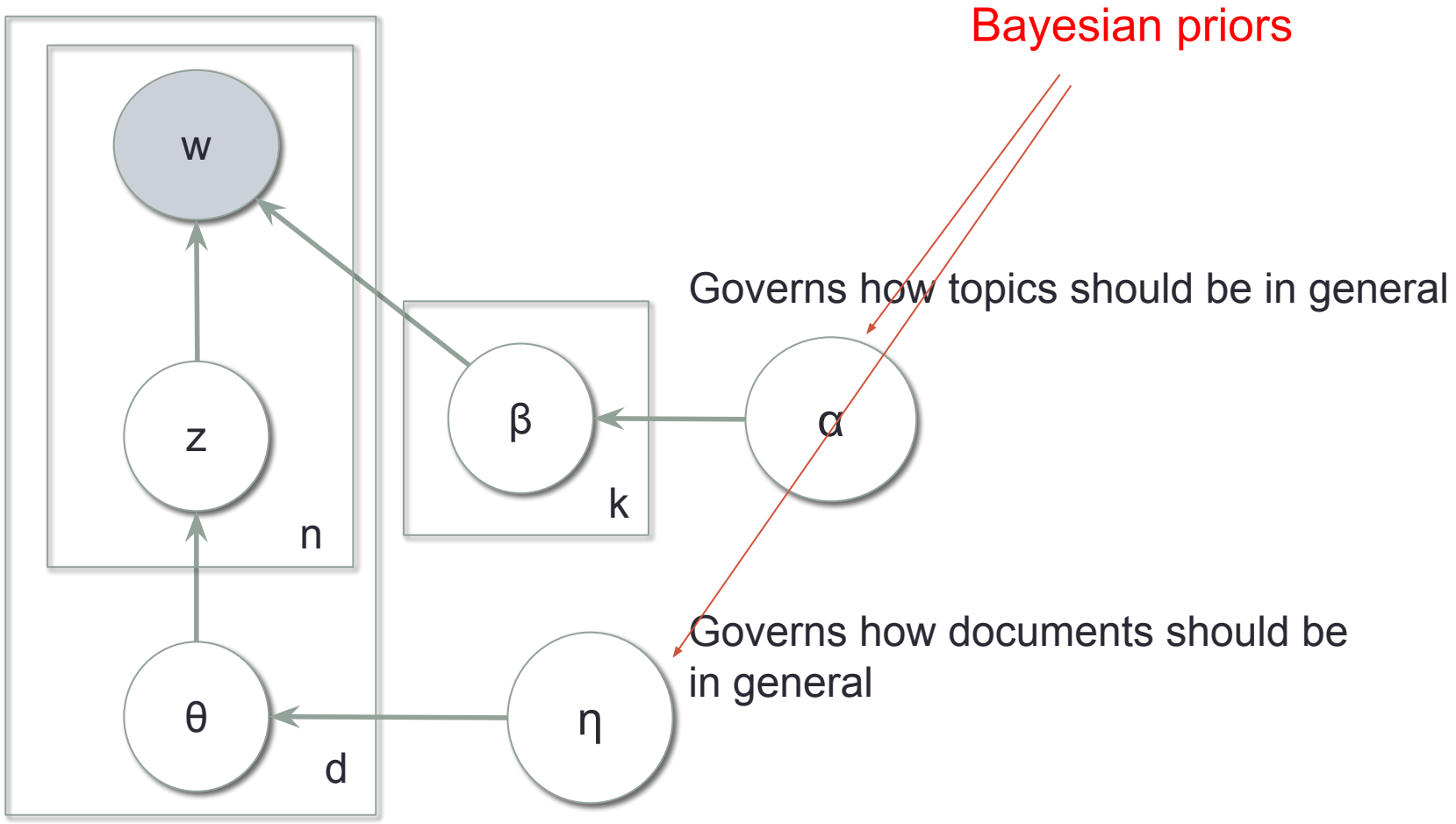
pLSA

- pLSA automatically clusters words into topic unigrams
  - Requires user to specify number of topics
- Automatically learn document representation based on the learned topics
  - DocA = [0.7 0.3] DocB = [0.2 0.8] DocC = [0.5 0.5]
- Overfits easily to data outside of the training set
  - Nothing that ties all document together
  - A document from a document collection should be have topic distributions that are similar
- Solution: LDA (Latent Dirichlet Allocation)

pLSA



LDA


Bayesian priors
w
z
n
β
k
α
Governs how topics should be in general
θ
d
η
Governs how documents should be in general

α and η

- 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 the 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

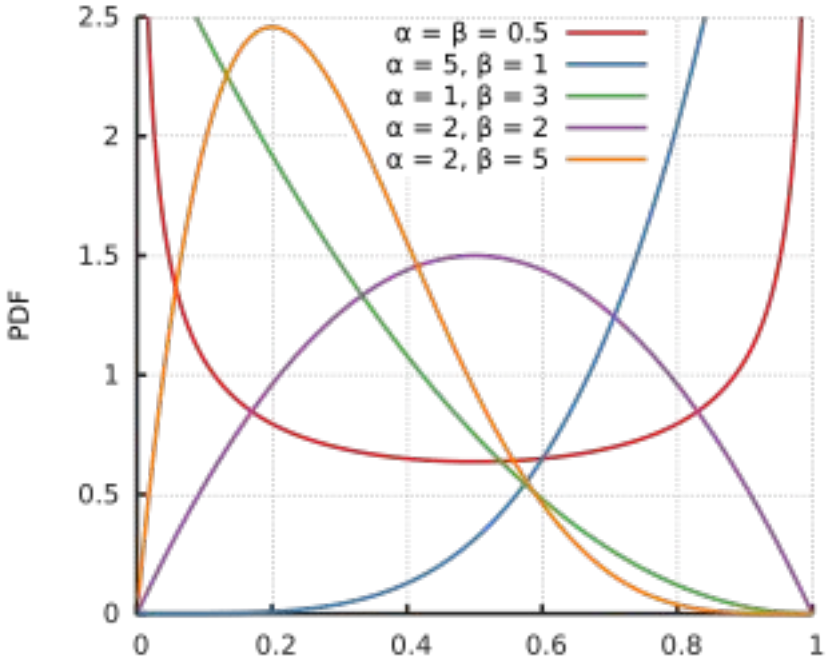
Dirichlet distribution

- 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

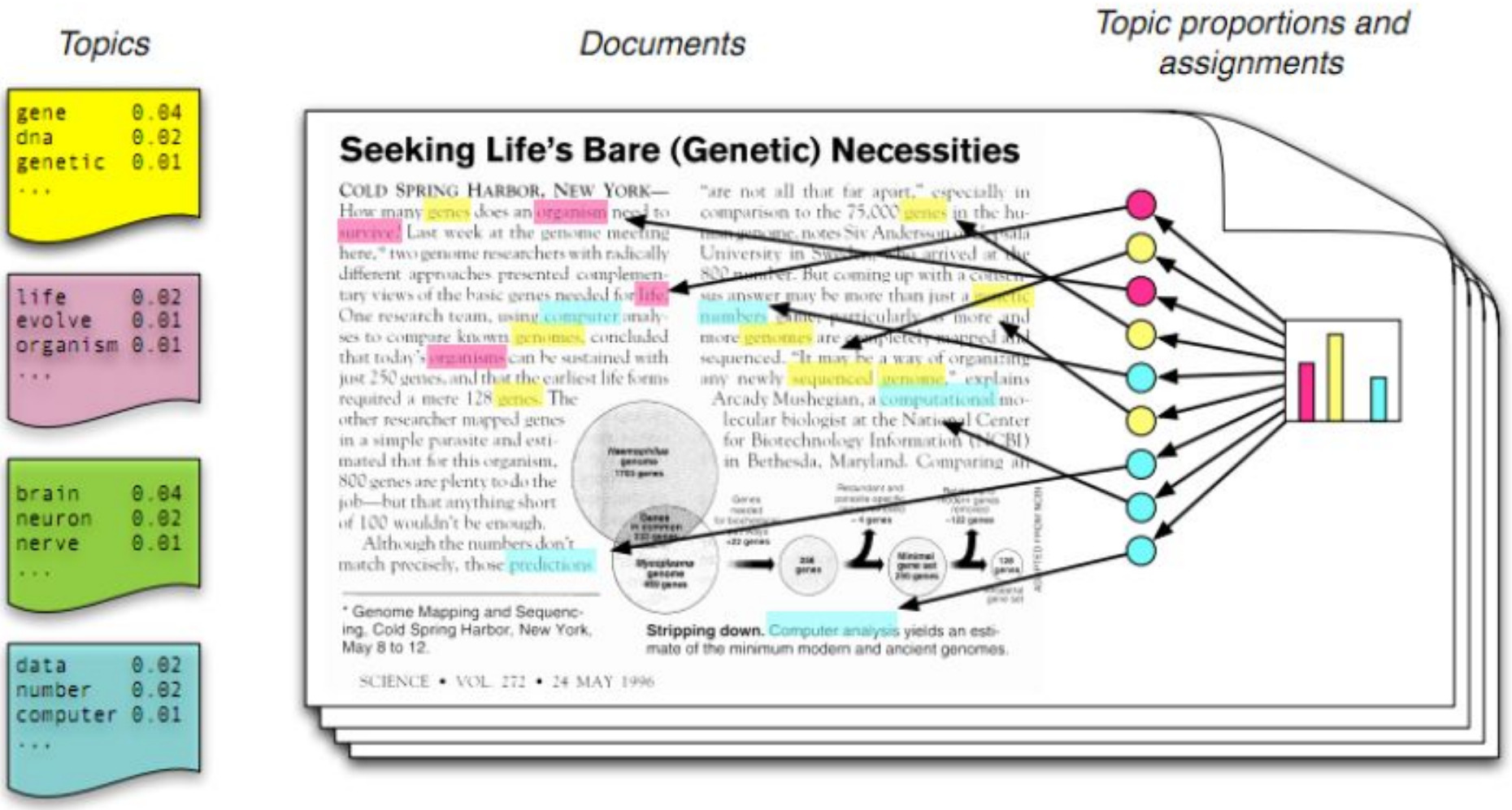


$$p(x; \alpha, \boldsymbol{\beta}) = \frac{1}{\mathrm{B}(\alpha, \beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

Beta distribution
A prior for binomial trails

Parameter learning

- How do we actually learn these parameters and latent variables?
  - Gibbs sampling
  - Variational methods



Introduction to Probabilistic Topic Models, Blei 2011
http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Example usage

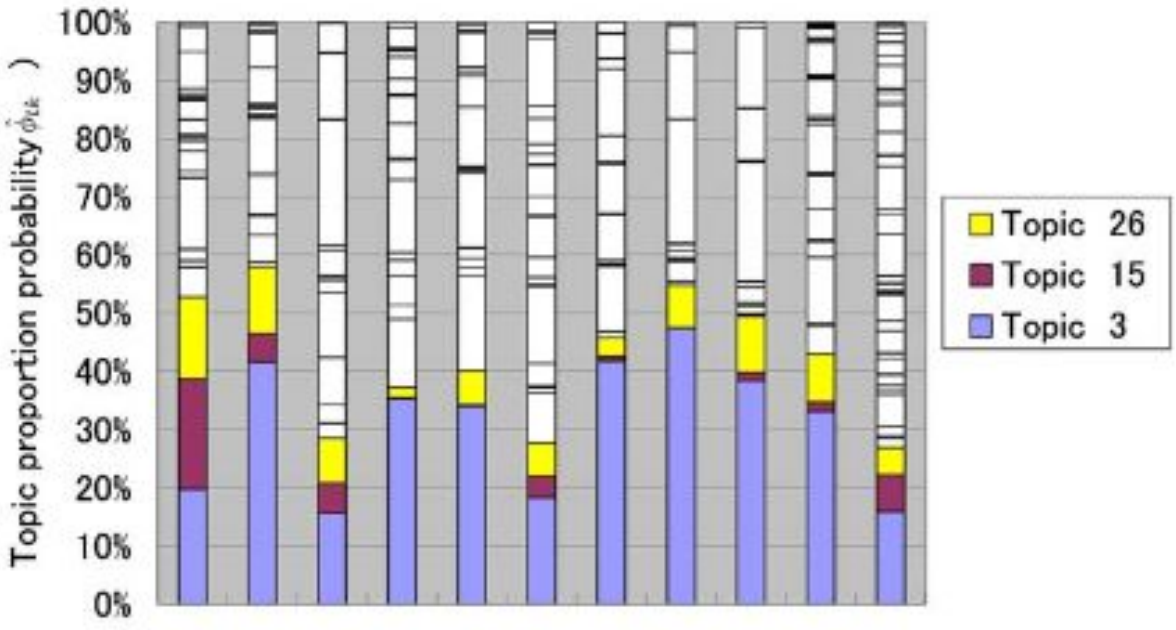
- Lots of lectures. Can we discover the topics? Can we classify the lectures? No topic labels given

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

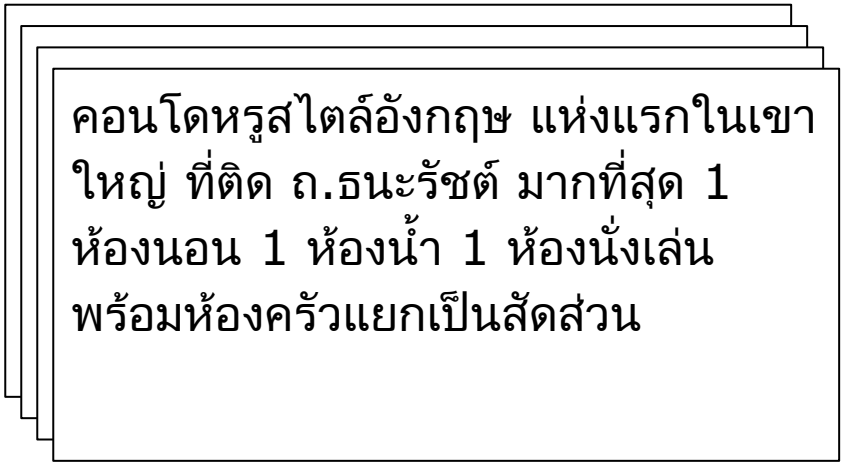
http://www.sciencedirect.com/science/article/pii/S0885230810000641

Topic distribution of each document



Unsupervised topic modeling for real estate

Can we learn real estate characteristics from unstructured data?

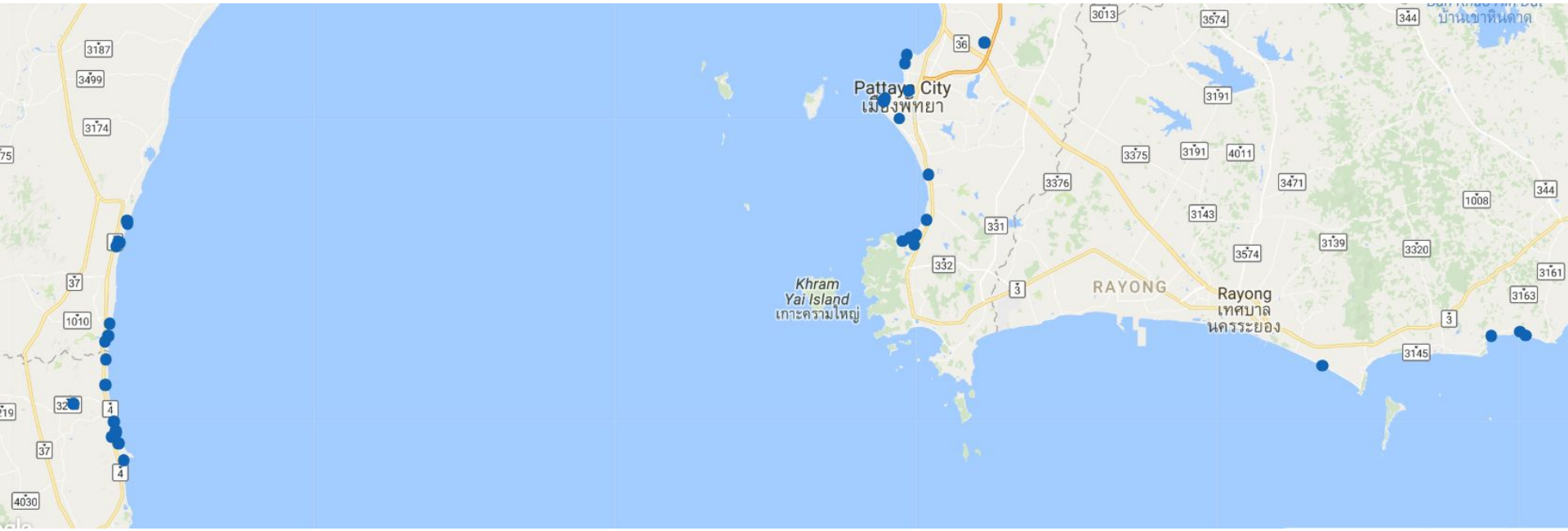


Just give it a bunch of descriptions

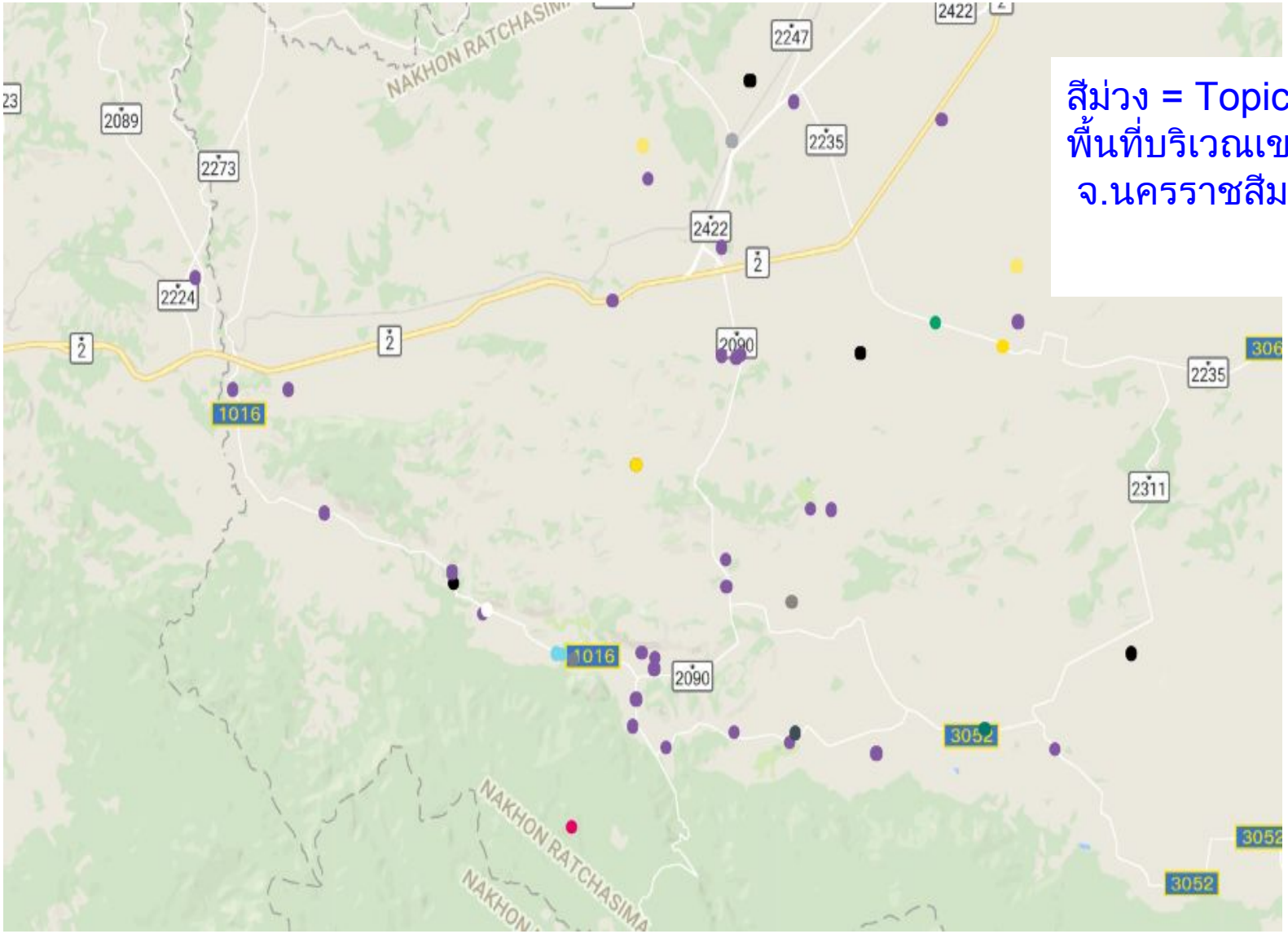
HOME DOT TECH

LDA Examples

Topic 28
0.068*"วิว" + 0.058*"ทะเล" + 0.038*"คอนโด" + 0.029*"หัว" + 0.027*"คอนโดมิเนียม" + 0.025*"มองเห็น" + 0.023*"ทัศนียภาพ" + 0.022*"ชายหาด"



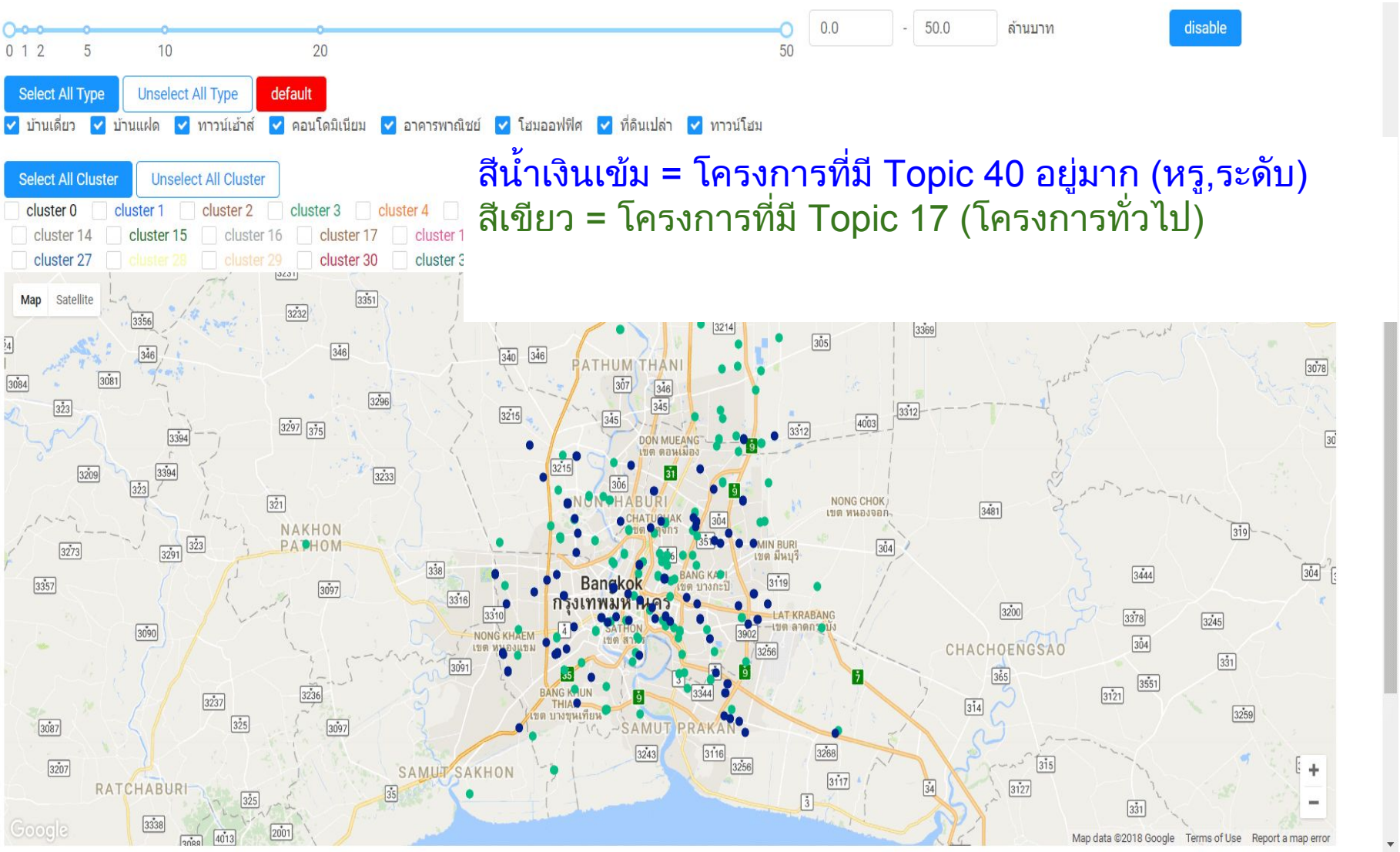
Topic 9
0.071*"ธรรมชาติ" + 0.031*"บรรยากาศ" + 0.028*"ร่มรื่น" + 0.027*"บ้าน" + 0.025*"ท่ามกลาง" + 0.025*"สวน" + 0.025*"สัมผัส" + 0.021*"พื้นที่"



สีม่วง = Topic 9
พื้นที่บริเวณเขาใหญ่
จ.นครราชสีมา

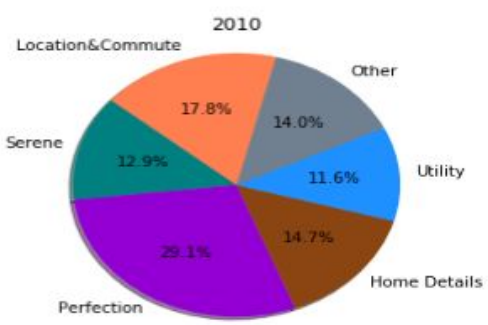
Topic 40
0.115*"ระดับ" + 0.066*"เหนือ" + 0.046*"หรู" + 0.031*"ทำเล" + 0.026*"ชีวิต" + 0.026*"ใช้ชีวิต" + 0.016*"สไตล์" + 0.016*"สะท้อน"

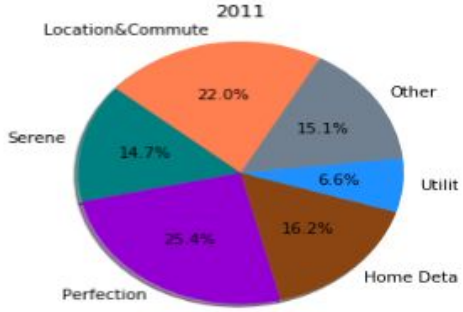
Topic 17
0.077*"พื้นที่" + 0.060*"ออกแบบ" + 0.045*"โล่ง" + 0.039*"โปร่ง" + 0.038*"ใช้สอย" + 0.020*"ประโยชน์" + 0.018*"ห้อง" + 0.017*"อาคาร"

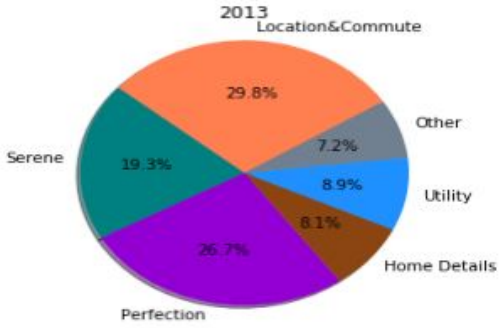


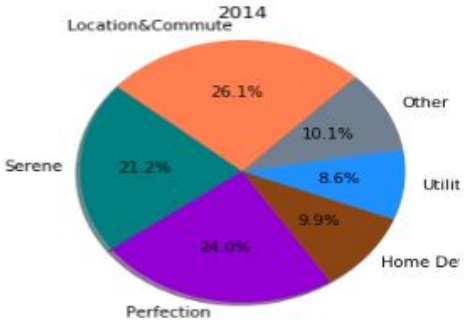
สีน้ำเงินเข้ม = โครงการที่มี Topic 40 อยู่มาก (หรู,ระดับ)
สีเขียว = โครงการที่มี Topic 17 (โครงการทั่วไป)

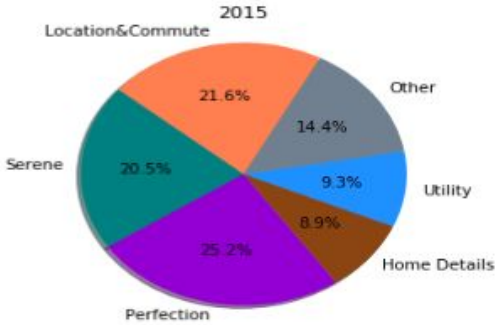
Time and advertisement trends

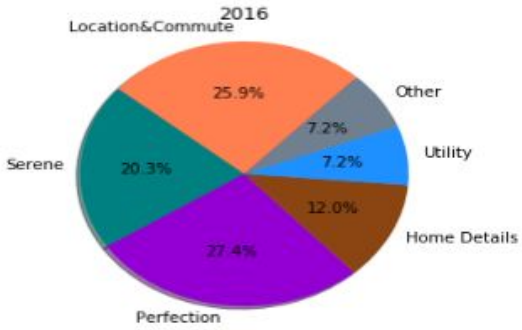


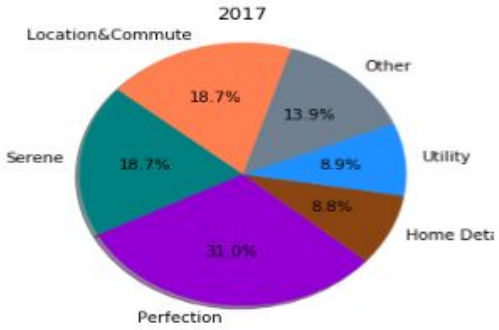


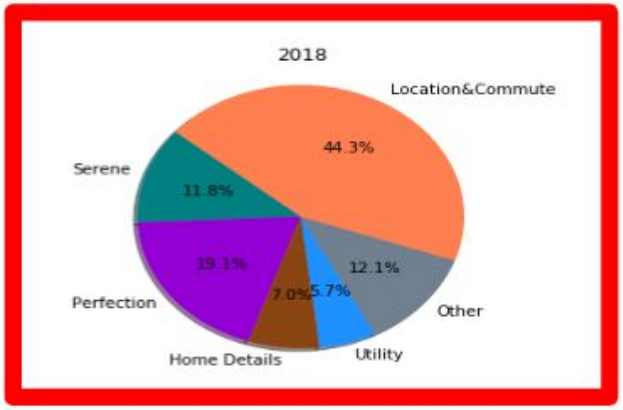






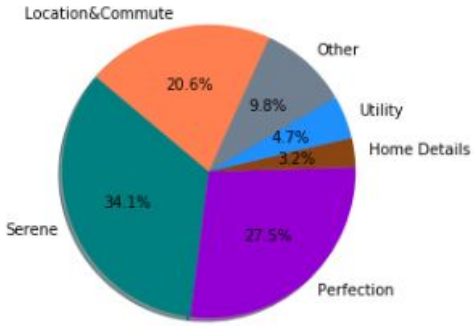


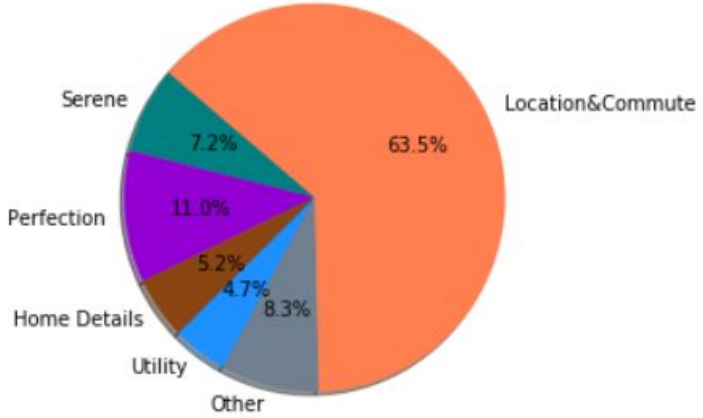


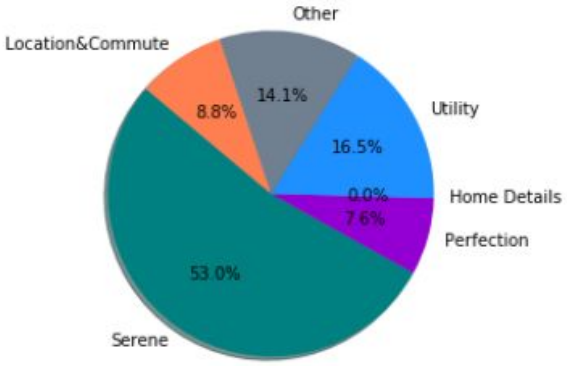


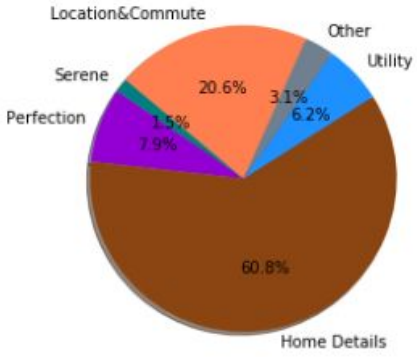
Advertisement niche of each developer

Each real estate developer has its own style







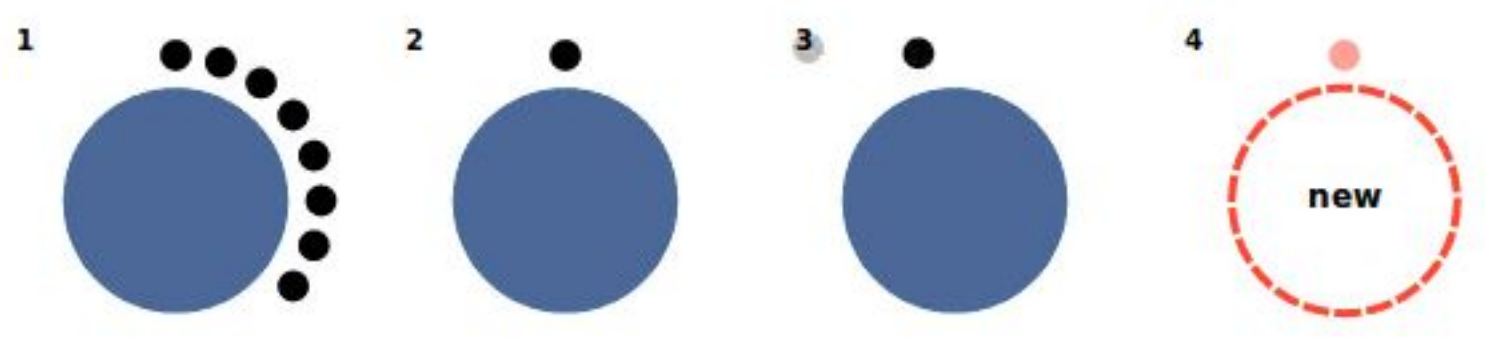


Discovering K

- 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.
- Hyperparameter c controls how often to start a new class.

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

Unsupervised Learning

- Clustering
  - Introduction to probabilistic graphical models
  - LDA
- Autoencoders
  - Denoising Autoencoder
  - VAE
    - Evidence lower bound
- Generative Adversarial Networks
- Random Cool stuff

“Unsupervised” Learning

- Given an image -> Tell you right away it’s a cat without anyone labeling anything?

  NO

- What does it do exactly?

  Can we somehow utilize unlabeled data to help with some task we care about?

First Unsupervised Learning Idea

- Suppose we have a magic function f that maps an image to two numbers

f

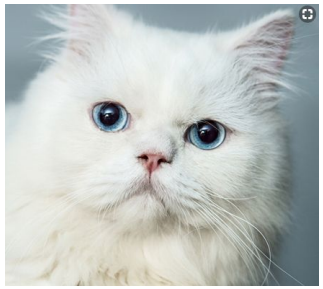
(4.1, 5.0)

First Unsupervised Learning Idea

- Suppose we have a magic function f that maps an image to two numbers

(4.1, 5.0)

(8.1, 8.0)

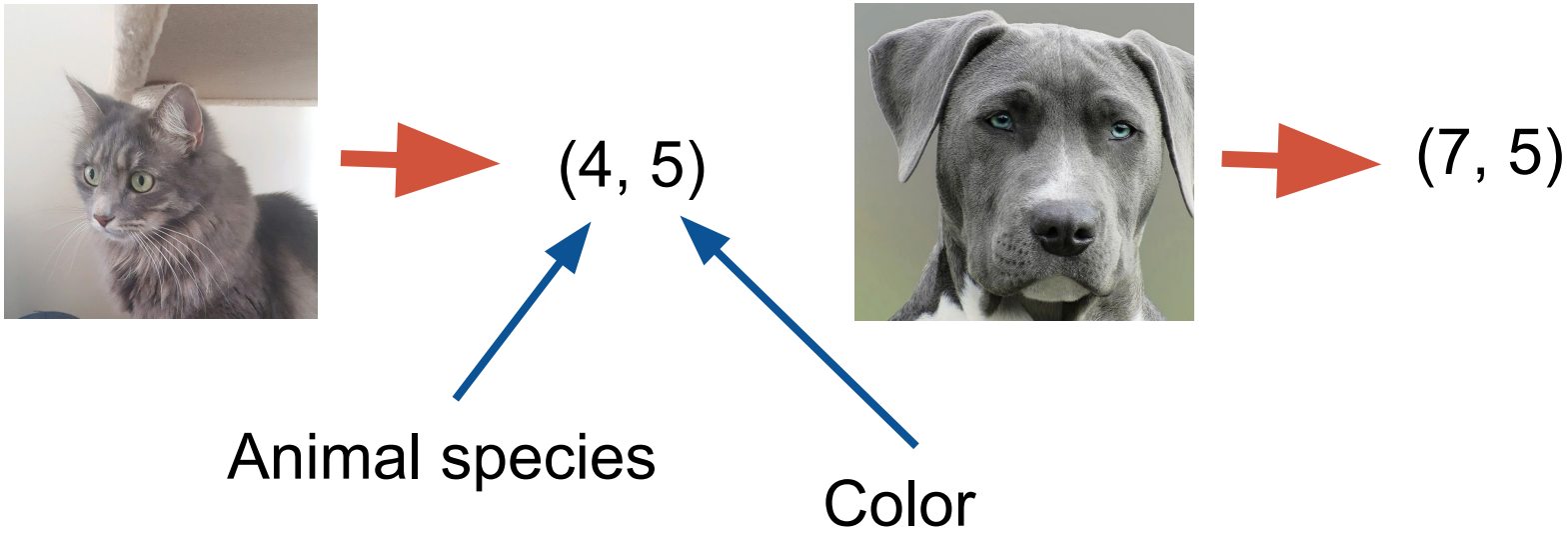


(4.2, 9.1)

(7.9, 4.9)

First Unsupervised Learning Idea

- Ideal magic function f

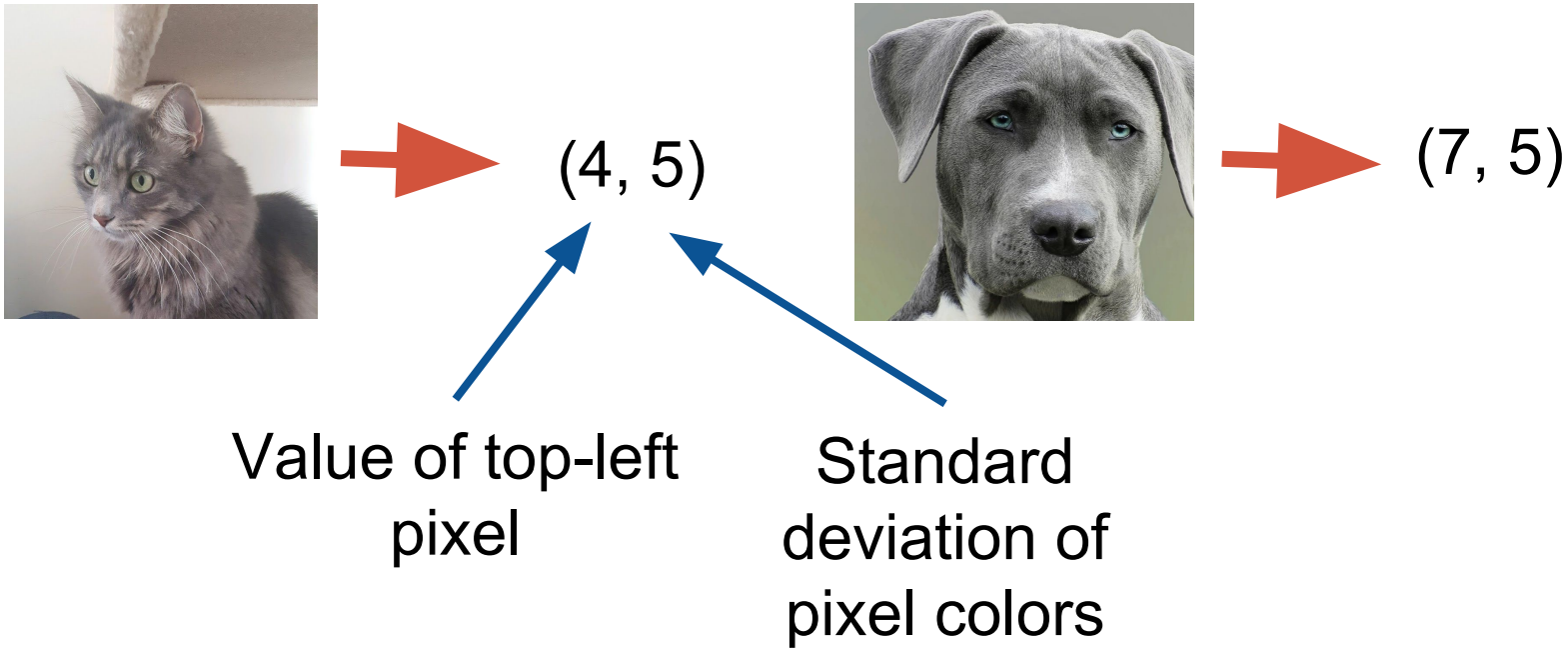


Getting this f would be so nice!

Given f, we can recognize animals with a linear classifier!

First Unsupervised Learning Idea

An alternative magic function f

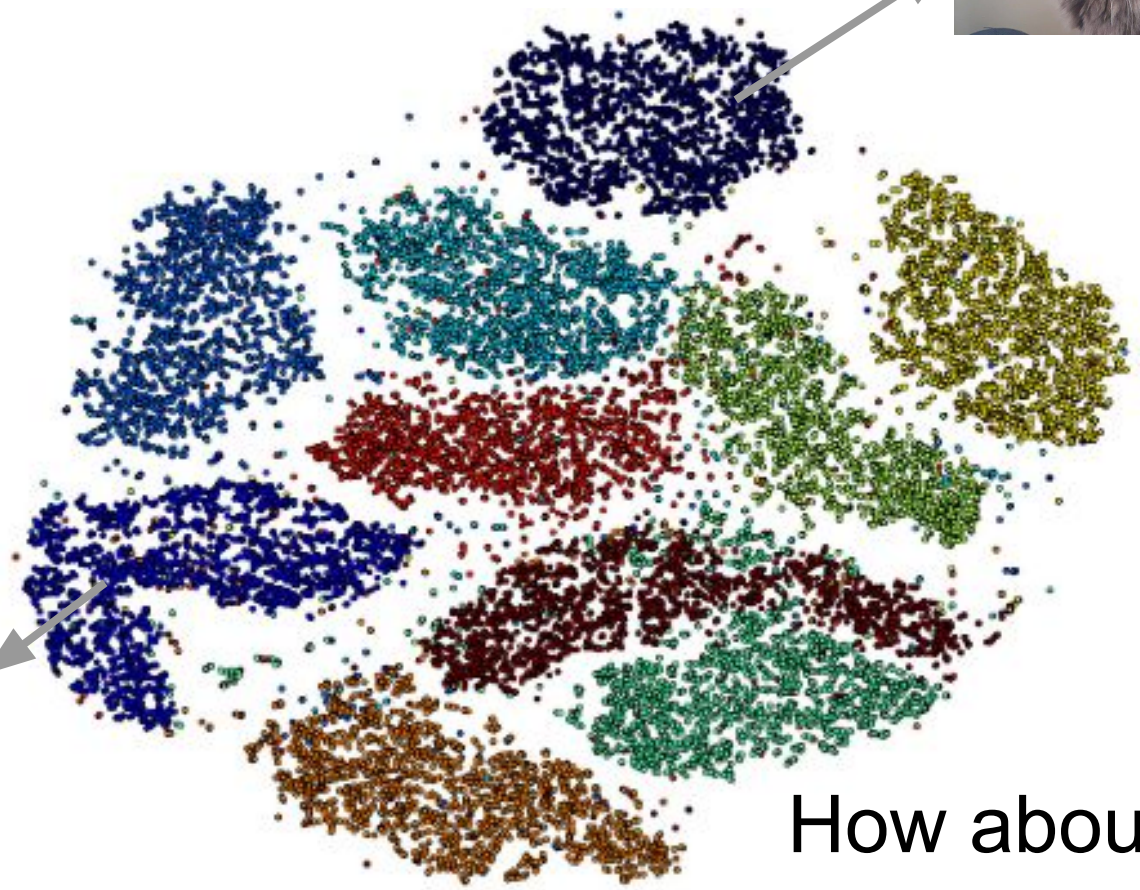


This f isn't so useful

Task: We try to learn the most "useful" f() from unlabeled data.

How to define usefulness? [subjective: Natural, human tasks]

One solution: Clustering



How about K-Mean?
PCA?

How about K-mean with L2?

We need a distance metric: How about pixelwise L2 norm?

$$\sum_p |I_p^1 - I_p^2|^2$$

In vector form:

$$\left\| \quad \right\|^2$$

How about K-mean with L2?

$\| \quad - \quad \|^2 < \| \quad - \quad \|^2$




White cat is more similar to white dog than to orange cat!

Pixel-wise L2 doesn't work that well. Too low-level comparison.
How can we come up with a useful distance metric? [DNN]

How about PCA?

(4.1, 5.0)

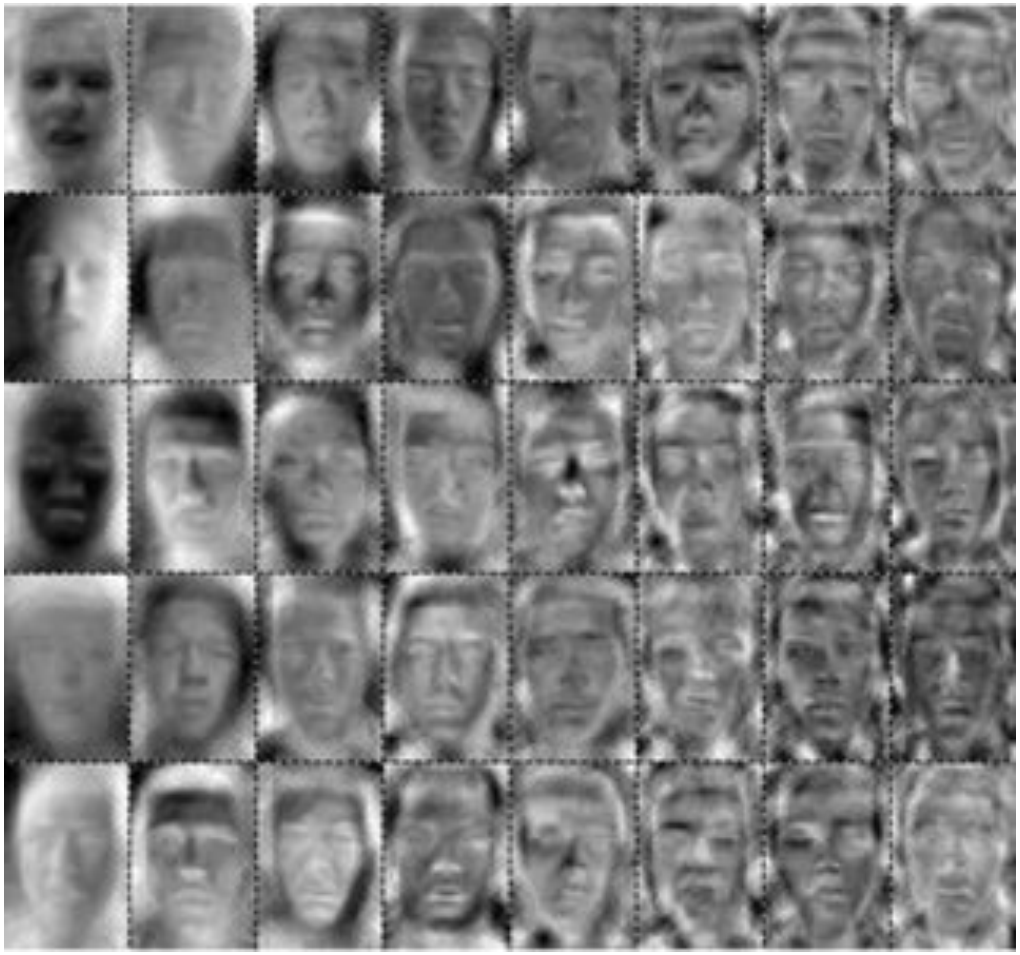
PCA

Dimensionality reduction
from (h x w) dim to 2 dim

Will this work?

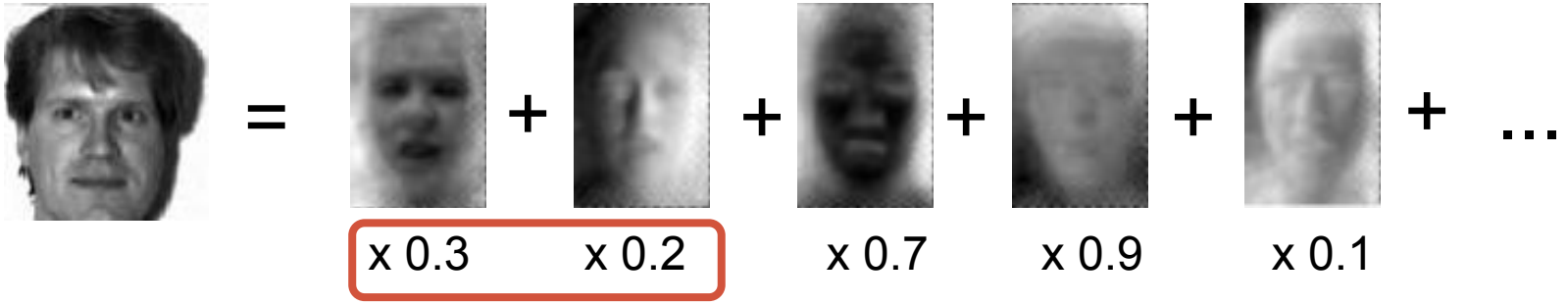
Historically, can be used to do simple face recognition.

PCA: Eigenfaces



PCA basis

PCA: Eigenfaces



Problems:

1. PCA is a linear operation (as well as LDA or fisherface). Many interesting attributes are highly non-linear, e.g., identity, emotion, and cannot be factored linearly.
2. Faces have to be aligned and centered for this to work
3. Illumination dominates!

How about DNN?

We know that DNN can model highly complex mapping.
How about modeling f with a neural net?

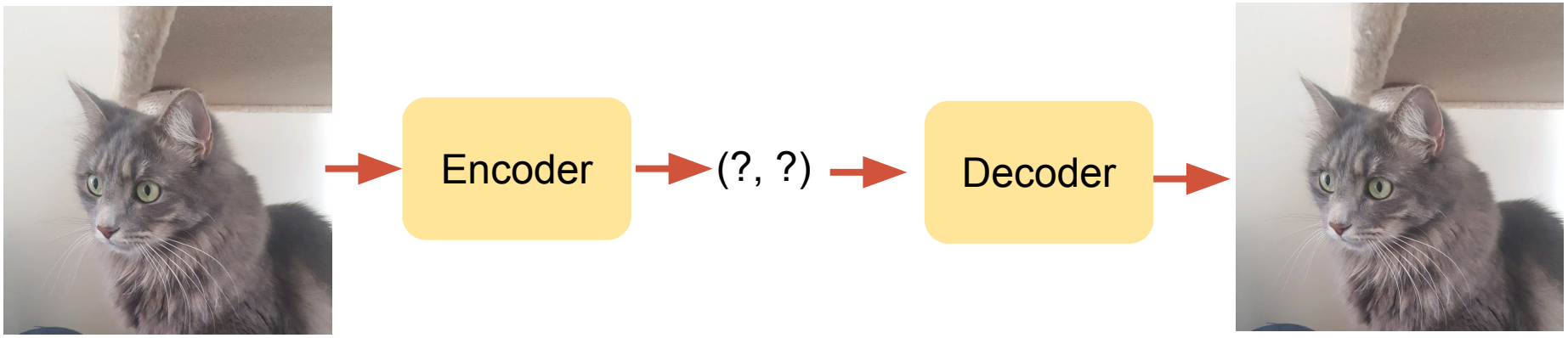
Deep Neural Network

(4.1, 5.0)

But how to train it? We don't know the target, groundtruth two numbers.

Autoencoder

Auto means self!



Input

X

Output

X'

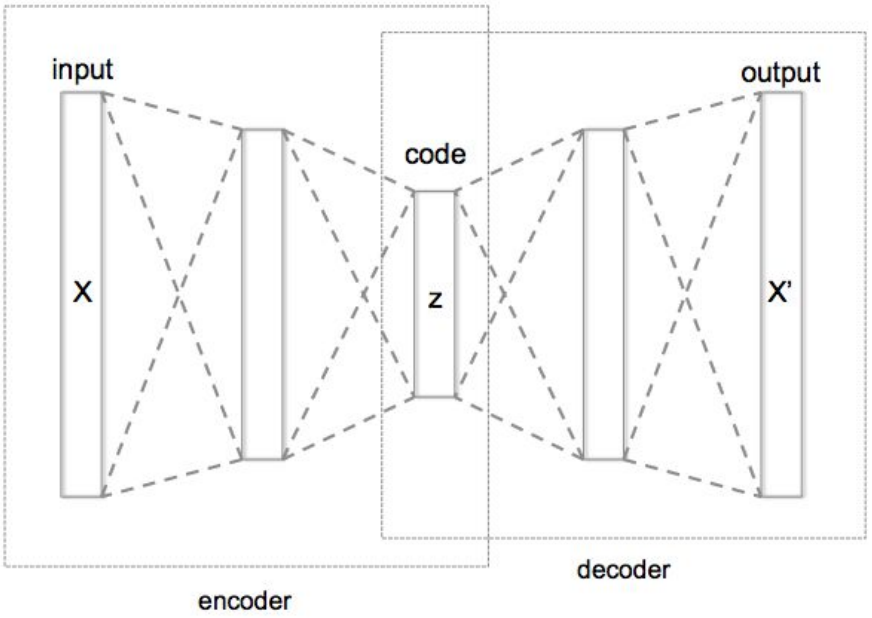
Now we can train the network with L2 Loss: $\|X - X'\|^2$

Autoencoder

Auto means self!

Input

X

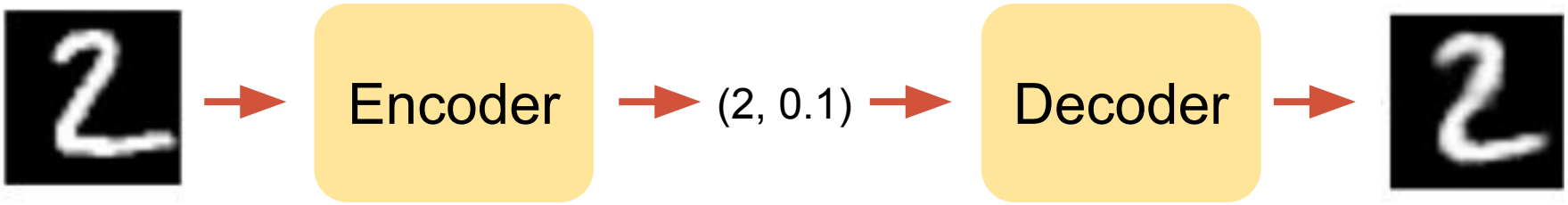


Output

X'

Now we can train the network with L2 Loss: $\|X - X'\|^2$

Autoencoding Game



Both Encoder and Decoder see the entire dataset.

Encoder: What two numbers should I send you so that you can re-draw my input as accurate as possible?

Encoder: Maybe the first number would directly correspond to what digit it is. The second, perhaps, the thickness of the stroke?

Turns out DNN does surprisingly well at encoding important attributes.

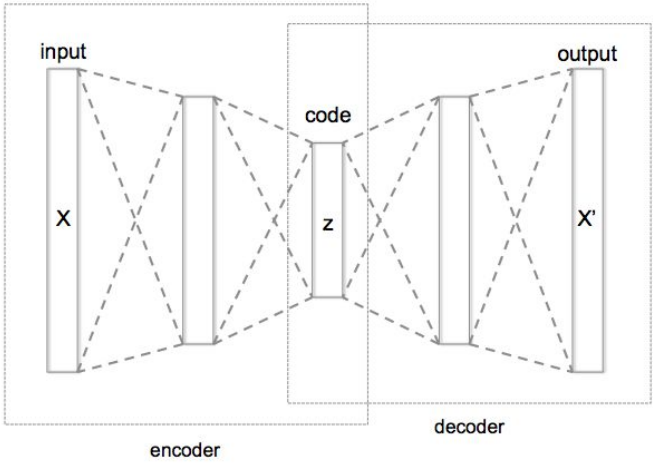
Autoencoder

Can we pick an arbitrary dimension for the code?

- When z has the same dimension as the input and output, NN cheats and outputs an identity mapping -> useless.

Input

X

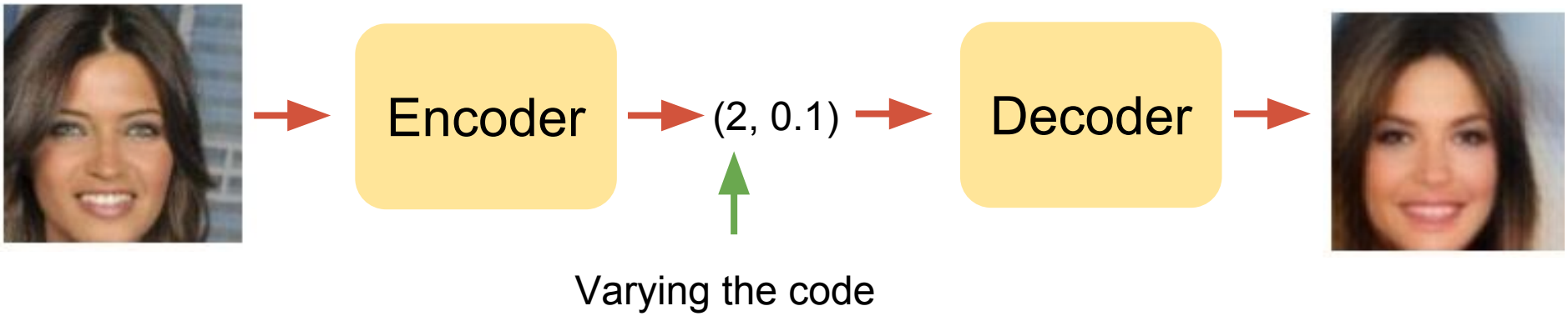


Output

X'

- When z dim << input dim, NN has to work hard!

Inspecting attributes learned from AE



Deep Feature Consistent Variational Autoencoder [Hou et al. 2016]

Attributes from autoencoder

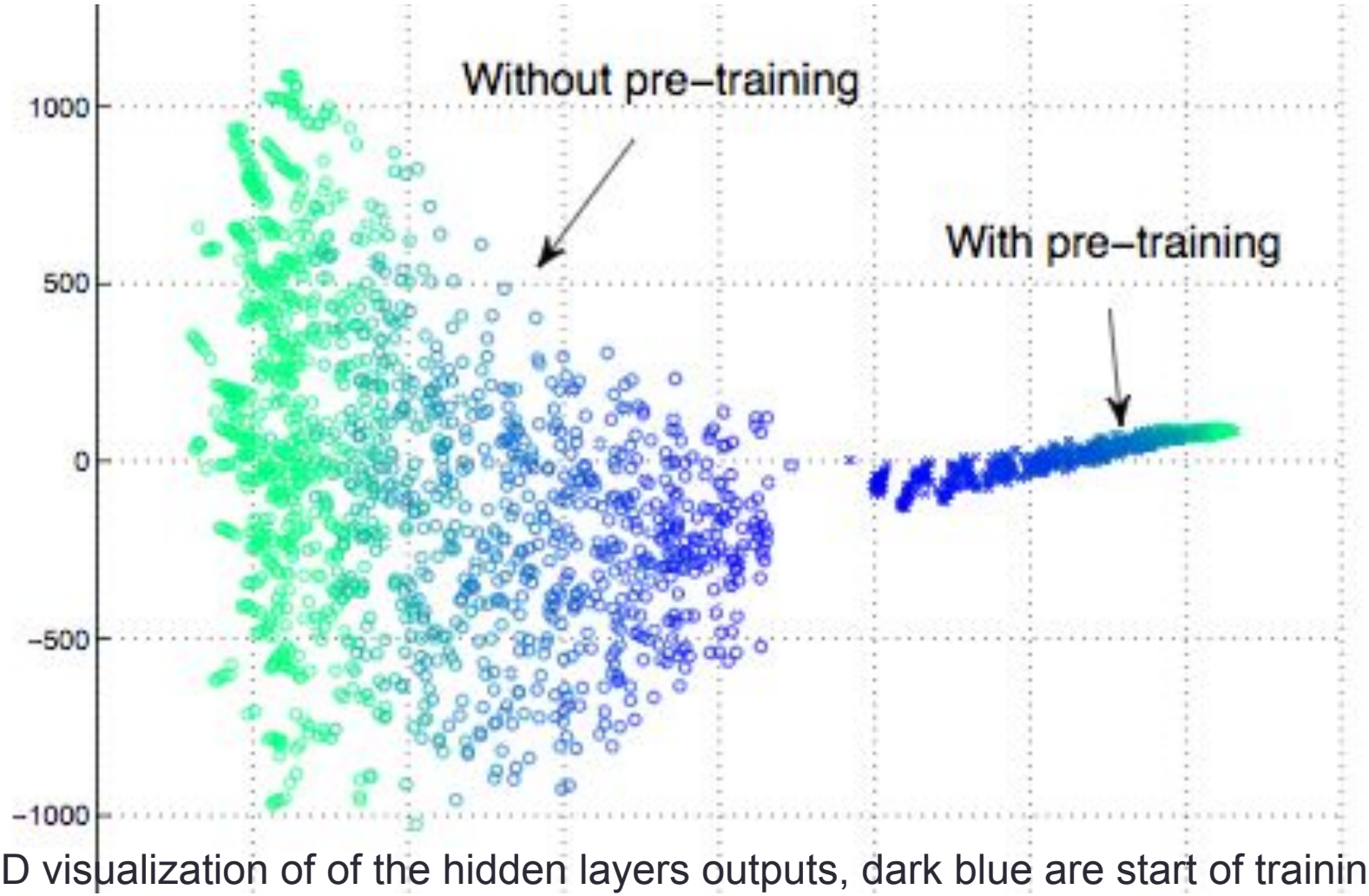
Deep Feature Consistent Variational Autoencoder [Hou et al. 2016]

Utilizing autoencoder for supervised tasks

Many ways to help with a supervised task such as recognition.

1. Append the input with the code from the encoder
2. Stick a classifier on top of the encoder
3. Used for pretraining a network

Pre-training effects



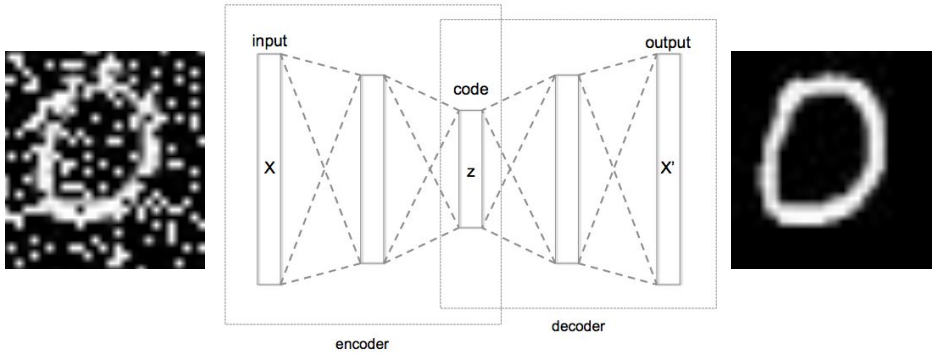
2D visualization of of the hidden layers outputs, dark blue are start of training

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

Denoising autoencoder

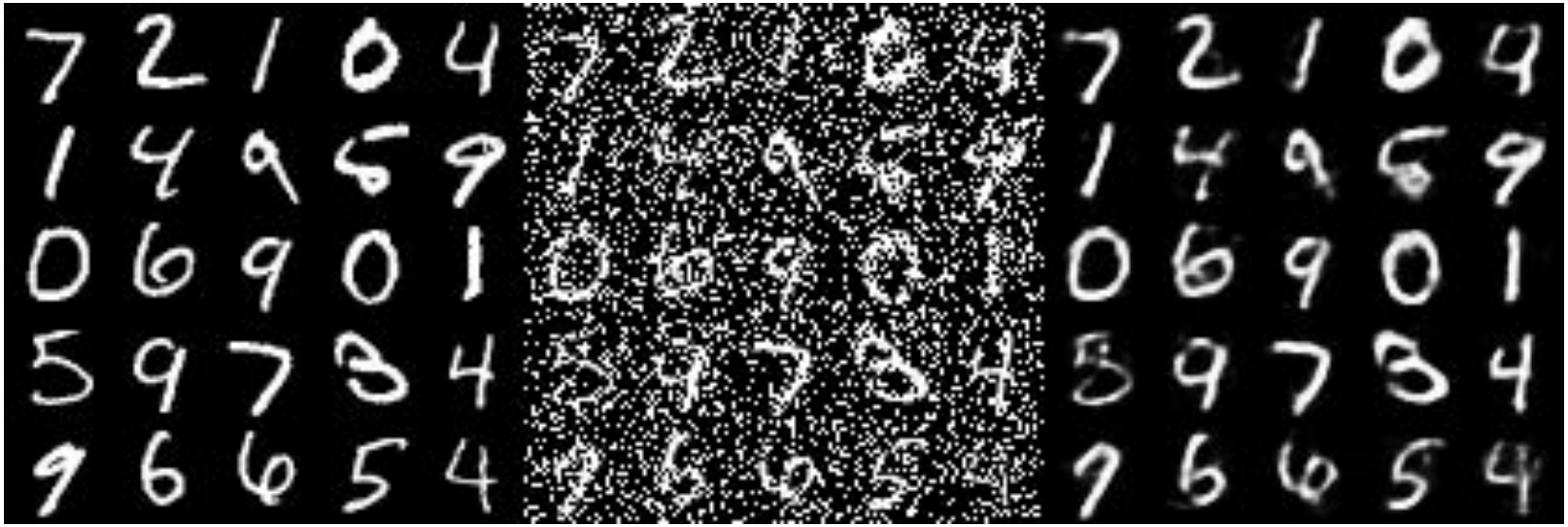
- Motivation: Making AE more robust -- really learn what's important

- 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

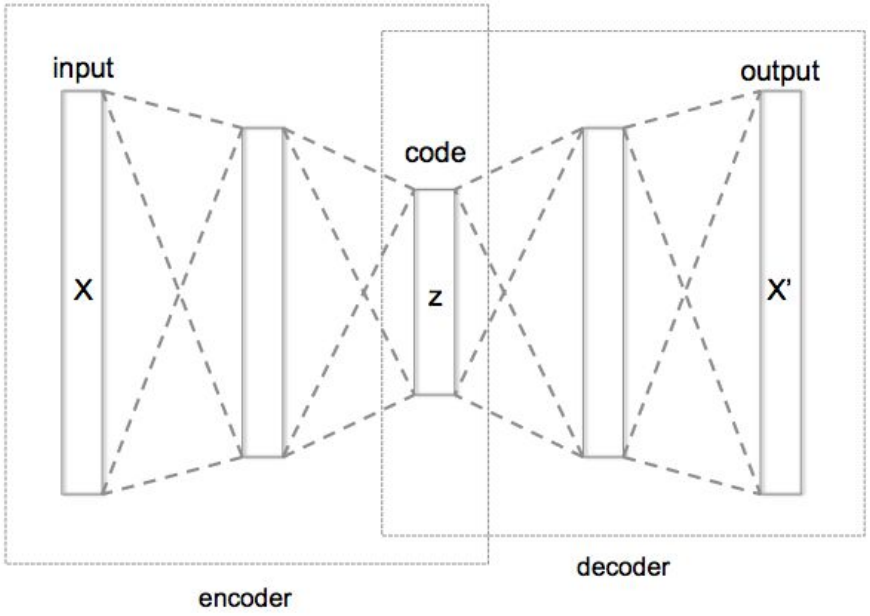


https://www.doc.ic.ac.uk/~js4416/163/website/autoencoders/denoising.html

Other uses

- Besides denoising, can also be used to make your “code” more robust to the type of corruption you introduced.
- Finding similar images. Combining with clustering algos.
- Generative models / generating photos

Autoencoders



- Automatically learn features
- Non-linear dimensionality reduction / data compression

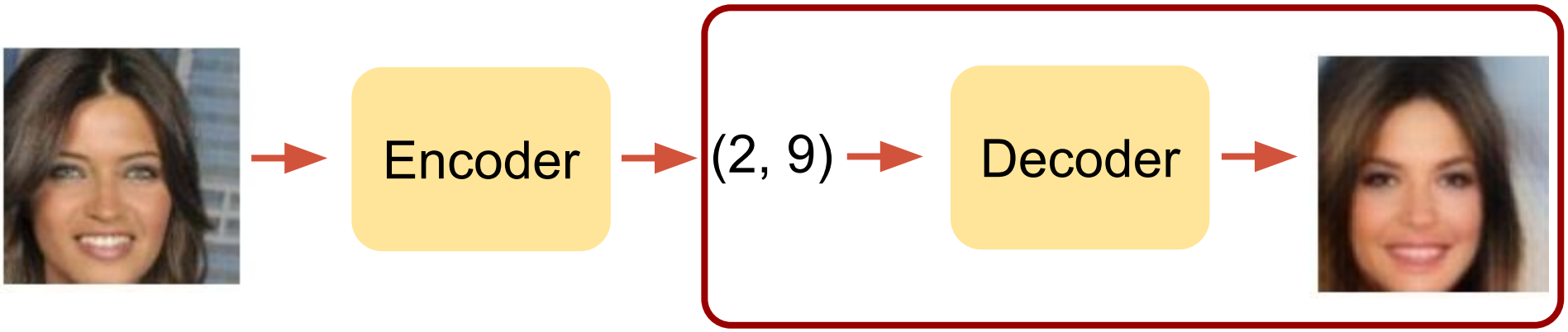
Variational Autoencoder

Original and by-product motivations:

1. Allows approximation of intractable posterior.

2. Generate photos from the decoder

Motivation #2 -- Generating Photos



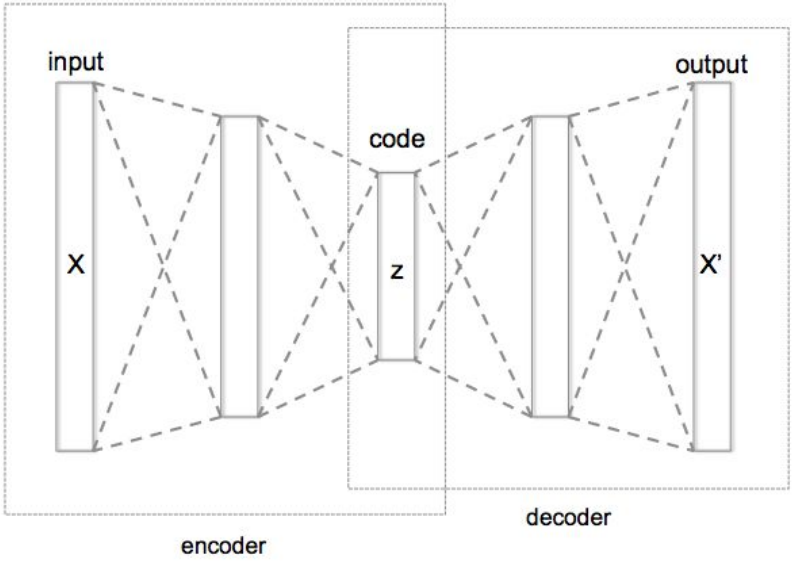
Use this part to generate an image from random code.

But how do you generate the two numbers? Uniform? Gaussian? Encoder could output two numbers with some crazy distribution.

VAE allows us to force the latent code to follow some distribution like a unit normal distribution.

Motivation #1 -- Solving Intractable Inference

Autoencoder can be viewed from a probabilistic perspective.



However, a standard AE isn't quite "probabilistic."

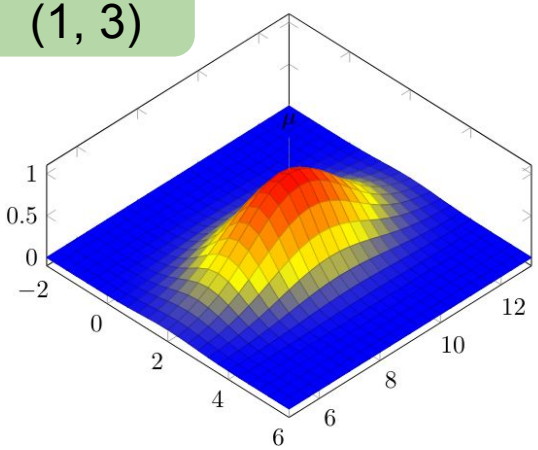
I.e., the encoder deterministically outputs z given x. Rather we want a distribution of the code (posterior p(z | x)).

Standard AE

Encoder → (2, 9) → Decoder

Probabilistic AE

Encoder → Means (2, 9) / Std. (1, 3) → Sample → (2.1, 9.2) → Decoder



Autoencoding Variational Bayes: Bayesian Review

One important inference problem : Posterior inference

$$p(\mathbf{z} \mid \mathbf{x})$$

$\mathbf{x}$ Observed data

$\mathbf{z}$ Latent variable (hidden code)

Bayesian Review

One important inference problem : Posterior inference

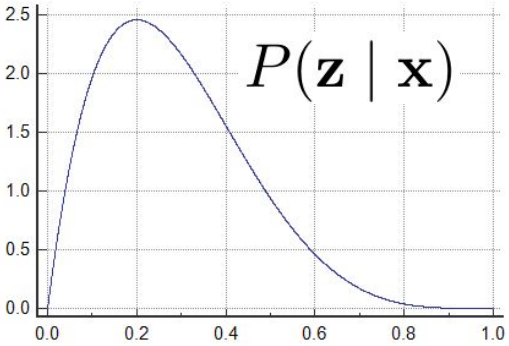
$$p(\mathbf{z} \mid \mathbf{x})$$

$\mathbf{x}$ Observed data

2 heads out of 10 coin flips
(Binomial Dist)

$\mathbf{z}$ Latent representation
(hidden code)

The coin fairness
z = 0.5, fair coin
z = 1, always head
z = 0, always tail

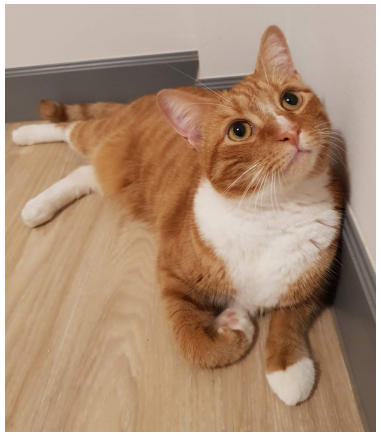


Bayesian Review

One important inference problem : Posterior inference

$$p(\mathbf{z} \mid \mathbf{x})$$

$\mathbf{x}$ Observed data



$\mathbf{z}$ Latent representation (hidden code)

Class label cat, color, can fly?, ...

$p(\mathbf{z} \mid \mathbf{x})$

Tells you **not only** what animal is the most likely, **but also** how “orange cat” it is, how “white dog” it is, etc -- the probability of all hidden code

Variational Bayesian Review

One important inference problem : Posterior inference

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})} = \frac{p(\mathbf{z}, \mathbf{x})}{\int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}}$$

Hard to compute because z can be high-dimensional

E.g. **z** = {z_1, z_2, z_3, ...}

z_1 : class label {cat, dog, rabbit, rat}

z_2 : color, a real number

z_3 : ability to fly {yes, no}

$$\int d\mathbf{z} = \sum_{z_1} \int_{z_2} \sum_{z_3} \ldots dz_2$$

Bayesian Review

One important inference problem : Posterior inference

$$p(\mathbf{z} \mid \mathbf{x}) = \frac{p(\mathbf{z}, \mathbf{x})}{p(\mathbf{x})} = \frac{p(\mathbf{z}, \mathbf{x})}{\int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}}$$

A solution:

Let's approximate the true posterior $p(\mathbf{z} \mid \mathbf{x})$

with an approximate, easier-to-work-with $q(\mathbf{z} \mid \mathbf{x})$

and solve this as an optimization problem.

Approximate True Posterior

Let's approximate the true posterior $p(\mathbf{z} \mid \mathbf{x})$ with an approximate, easier-to-work-with $q(\mathbf{z} \mid \mathbf{x})$ and solve this as an optimization problem.

Approximate True Posterior

Let's approximate the true posterior $p(\mathbf{z} \mid \mathbf{x})$ with an approximate, easier-to-work-with $q(\mathbf{z} \mid \mathbf{x})$ and solve this as an optimization problem.

Best $q^*(\mathbf{z} \mid \mathbf{x})$ is the one closest to $p(\mathbf{z} \mid \mathbf{x})$

A solution: $q^*(\mathbf{z} \mid \mathbf{x}) = \underset{q}{\operatorname{argmin}} \operatorname{KL}\left[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})\right]$

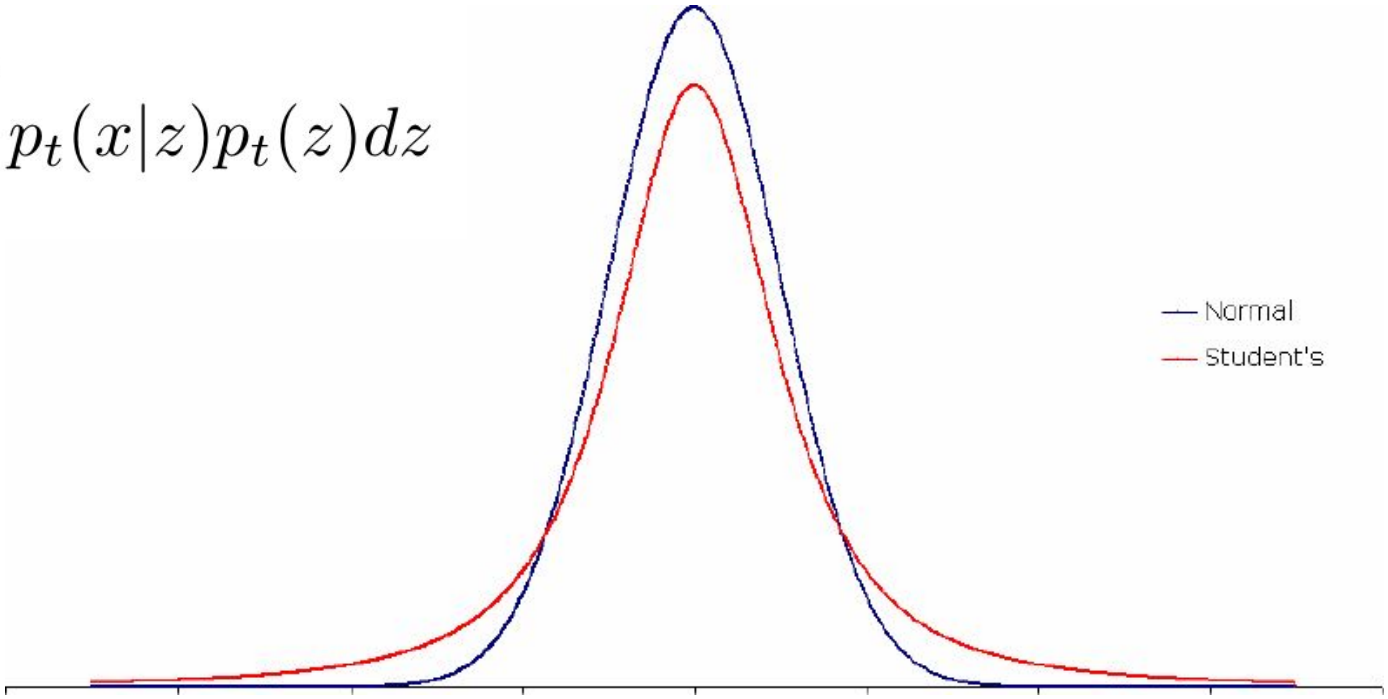
- But why not $\operatorname{KL}\left[p(\mathbf{z} \mid \mathbf{x}) \| q(\mathbf{z} \mid \mathbf{x})\right]$ or $D\left(p(\mathbf{z} \mid \mathbf{x}), q(\mathbf{z} \mid \mathbf{x})\right)$?
- How is this related to autoencoder?

Digression: Model evidence

Model evidence (or marginal likelihood) is simply $p(\mathbf{x})$

$$p_g(x) = \int p_g(x|z)p_g(z)dz$$

$$p_t(x) = \int p_t(x|z)p_t(z)dz$$



Approximate True Posterior

Let's approximate the true posterior $p(\mathbf{z} \mid \mathbf{x})$ with an approximate, easier-to-work-with $q(\mathbf{z} \mid \mathbf{x})$ and solve this as an optimization problem.

Because doing this (i.e., using KL (q || p)):

$$q^*(\mathbf{z} \mid \mathbf{x}) = \underset{q}{\operatorname{argmin}} \, \mathrm{KL}\left[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})\right]$$

Corresponds to maximizing the evidence lower bound (ELBO)!

Proof

This KL is hard to compute

$$q^*(\mathbf{z} \mid \mathbf{x}) = \operatorname*{argmin}_q \mathrm{KL}\left[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})\right]$$

$$\mathrm{KL}\left[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})\right] = \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x})] - \mathbb{E}_q[\log p(\mathbf{z} \mid \mathbf{x})]$$

(by definition)

$$= \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x})] - \mathbb{E}_q[\log p(\mathbf{z}, \mathbf{x})] + \log p(\mathbf{x})$$

(expanding $P(z \mid x)$, $E_q[\log p(x)] = \log p(x)$)

$$= \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})] + \log p(\mathbf{x})$$

Constant w.r.t q. So, we can safely kill it

$$\min_q \mathrm{KL}\left[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})\right] \iff \min_q \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

Questions: 1. What's this new term? 2. Is it easier to evaluate?

Proof

$$\min_q \mathrm{KL}\left[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})\right] \iff \min_q \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

Let's interpret this new term:

From previous slide, we have:

$$\log p(\mathbf{x}) = \mathrm{KL}\left[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})\right] - \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

We know KL >= 0, always!

$$\log p(\mathbf{x}) \geq \boxed{-\mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]}$$

(Lower bound of model evidence)

$$\min_q \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})] \iff \max_q -\mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$
$$\iff \max_q \mathrm{ELBO}$$

But how to compute this ELBO term then?

Maximizing ELBO

$$q^*(\mathbf{z} \mid \mathbf{x}) = \operatorname*{argmin}_q \mathrm{KL}\left[q(\mathbf{z} \mid \mathbf{x}) \| p(\mathbf{z} \mid \mathbf{x})\right]$$

$$\max_q \mathrm{ELBO} = \max_q -\mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

$$\iff \min_q \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x}) - \log p(\mathbf{z}, \mathbf{x})]$$

$$\iff \min_q \mathbb{E}_q[\log q(\mathbf{z} \mid \mathbf{x})] - \mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{z})]$$

$$\iff \min_q \mathrm{KL}[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})]$$

We're free to pick

Standard AE

Encoder → (2, 9) → Decoder

Variational AE

Encoder → Means, Std. → Sample → (2.1, 9.2) → Decoder

$q(\mathbf{z} \mid \mathbf{x})$

*Output as a sample from P(x | z).

$$\min_q \mathrm{KL}[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})]$$

Encoder

Means
(2, 9)

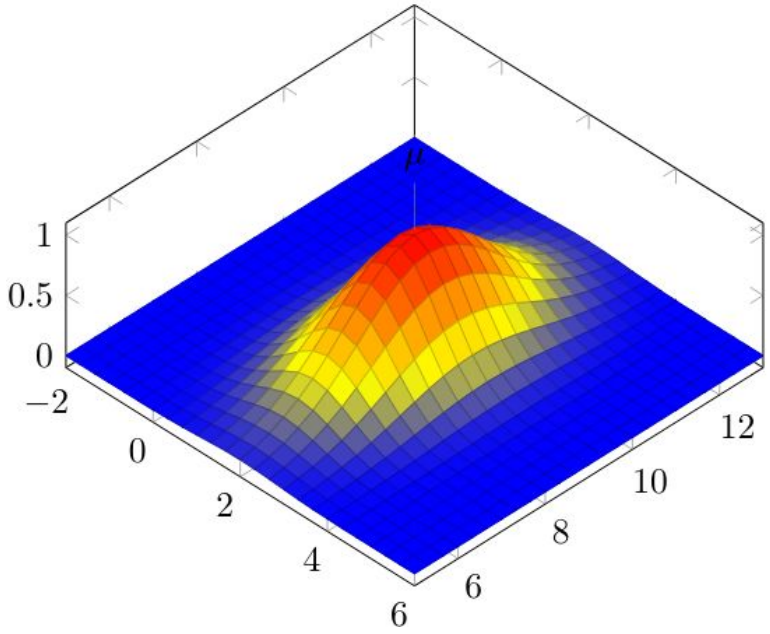
Std.

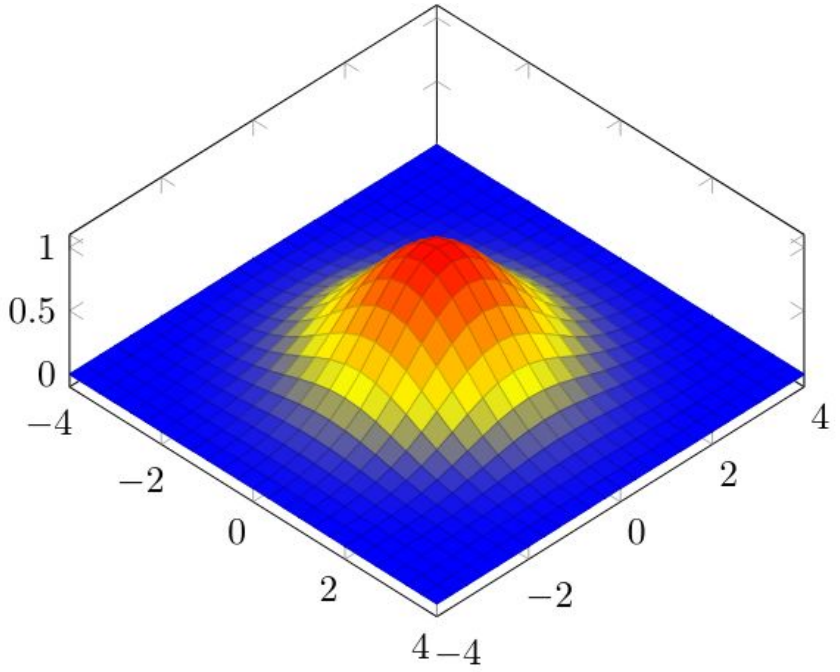
Sample

(2.1, 9.2)

$q(\mathbf{z} \mid \mathbf{x})$

Decoder





$q(\mathbf{z} \mid \mathbf{x})$ is forced to be close to $p(\mathbf{z}) = \mathcal{N}(0, I)$

$$\min_q \mathrm{KL}[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})]$$

What is this second term?

Second Term: Data Log Likelihood

$$\min_q \mathrm{KL}[q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z})]$$

If we also model $p(\mathbf{x} \mid \mathbf{z})$ as a Gaussian distribution:

$$p(\mathbf{x} \mid \mathbf{z}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-u)^2}{2\sigma^2}}$$

(Let's consider a single pixel case
u is the output from the decoder
x is the ground-truth pixel)

$$\propto \exp\left(-(x-u)^2\right)$$

$$-\log[\exp\left(-(x-u)^2\right)] = (x-u)^2$$ (with constants hidden)

Becomes the standard L2 error

Yet another reason why assuming Gaussian is nice

Encoder → (2, 9) → Decoder →

X X'

Standard AE Loss $\|X - X'\|^2$

Encoder → Means, Std. → Sample → (2.1, 9.2) → Decoder →

X X'

Variational AE Loss $\approx \|X - X'\|^2 + \mathrm{KL}[q(\mathbf{z} \mid \mathbf{x}) \| \mathcal{N}(0, I)]$

Precisely, $\min_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$

Last Problem

$$\min_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$$

When we need to compute the gradient with respect to the network parameters (I.e., ϕ, θ) for SGD, we're struck because we can't back-prop through the sampling for computing $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}$

Example:

$$z = y^2 \qquad z \sim \mathcal{N}(y, 1) \qquad z \sim q_\phi(z \mid x)$$

$$\frac{dz}{dy} = 2y \qquad \frac{dz}{dy} = ?? \qquad \frac{dz}{d\phi} = ??$$

Reparameterization Trick

$$z \sim \mathcal{N}(\mu, \sigma)$$

$$\frac{\partial z}{\partial \mu} = ??$$

$$\frac{\partial z}{\partial \sigma} = ??$$

$$z = \epsilon_i \times \sigma + \mu$$

$$\epsilon \sim \mathcal{N}(0, 1)$$

$$\frac{\partial z}{\partial \mu} = 1$$

$$\frac{\partial z}{\partial \sigma} = \epsilon_i$$

$$\min_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]$$

$$\iff \min_{q_\phi(\mathbf{z}|\mathbf{x})} \mathrm{KL}[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z})] - \mathbb{E}_{p(\epsilon)}[\log p_\theta(\mathbf{x} \mid \hat{\mathbf{z}})]$$

$$\hat{\mathbf{z}} = \epsilon \cdot \sigma + \mu, \quad p(\epsilon) \sim \mathcal{N}(0, I)$$

X

Encoder → (2, 9) → Decoder

X'

Standard AE Loss $\|X - X'\|^2$

X

Encoder → Means, Std. → Sample → (2.1, 9.2) → Decoder

X'

Variational AE Loss $\approx \|X - X'\|^2 + \mathrm{KL}[q(\mathbf{z} \mid \mathbf{x}) \| \mathcal{N}(0, I)]$

Disentangling Representation: β-VAE

We want compact meaningful representation.

But also disentangled representation

$$\mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta\, D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

Skin color

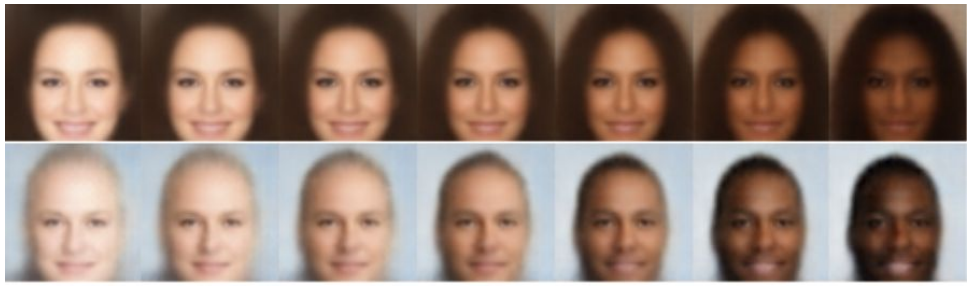
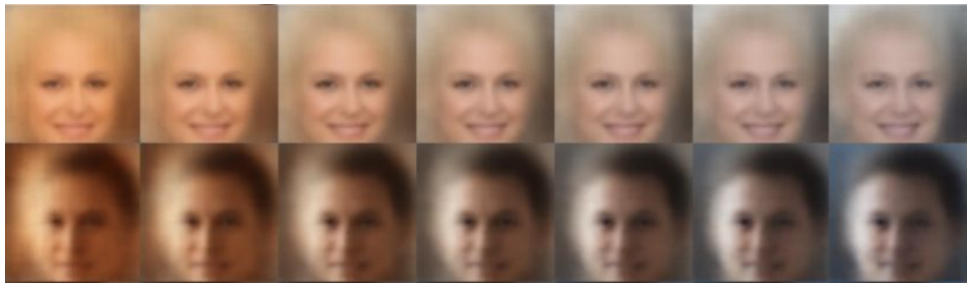


Image Saturation



Head Angle

Age / Gender

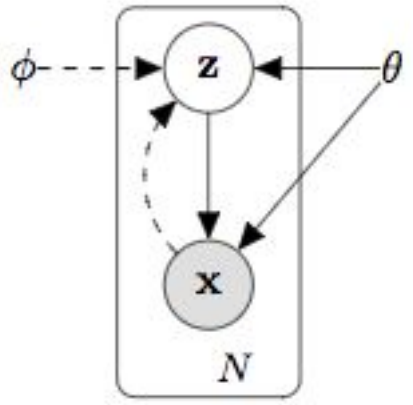
Variational Autoencoders (VAE)

- 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 mimic 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 maximize

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

- 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/

Generating Images

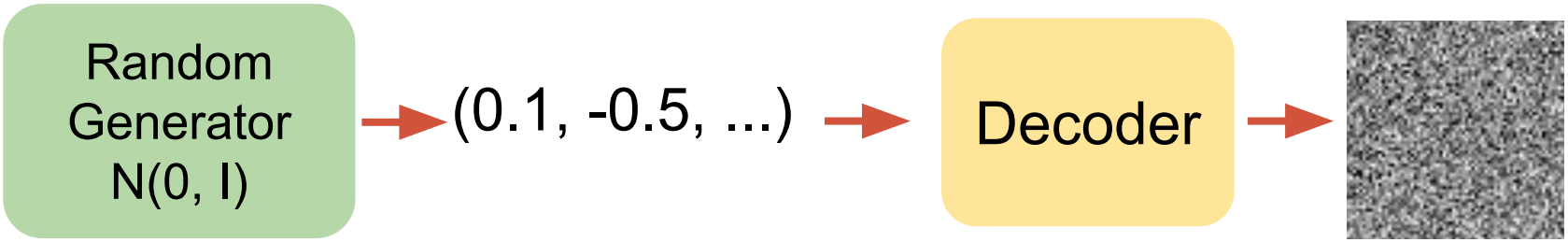
VAE can be used to generate images by feeding numbers into the decoder.

Are there other ways?

Given a collection of, say, faces, how can we generate new faces?

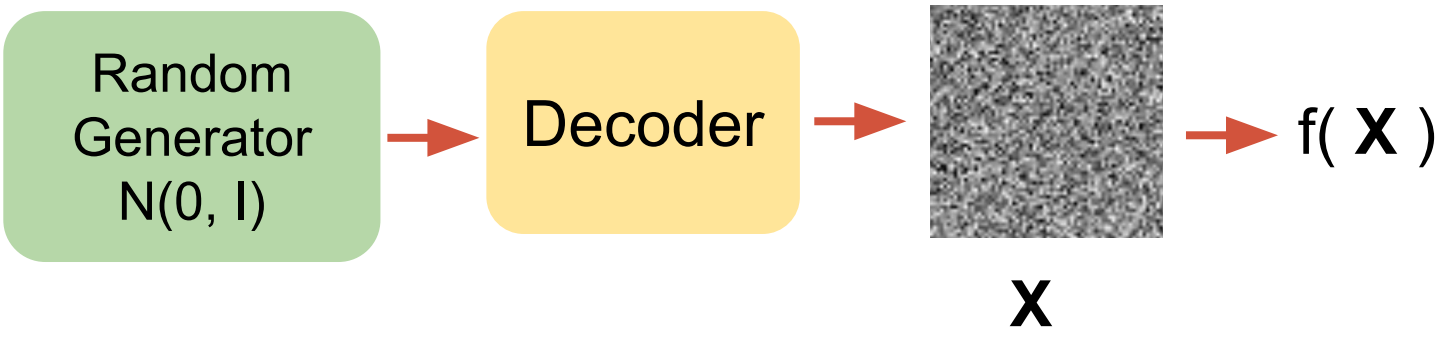
Maybe let's start with a decoder, because we need something that outputs an image anyway.

Generating Realistic Faces



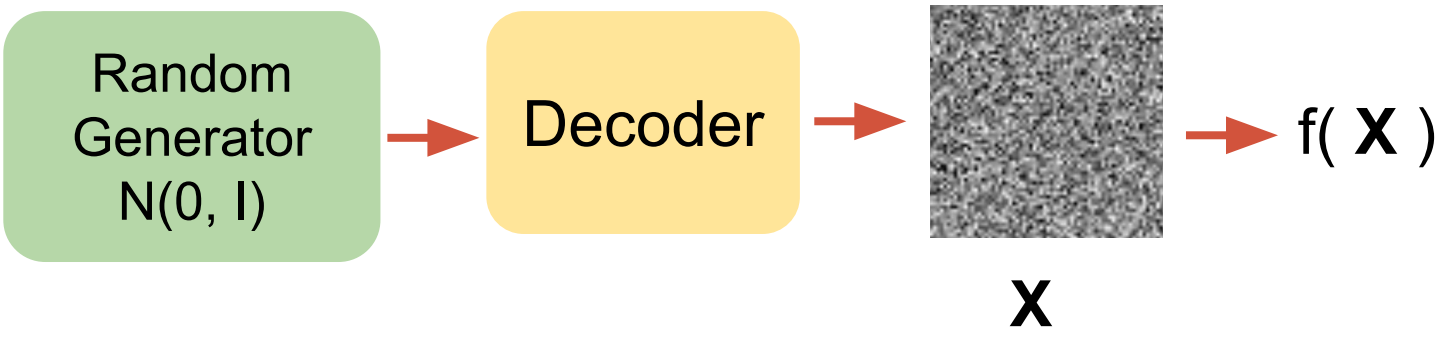
What kind of loss should we use?

Generating Realistic Faces



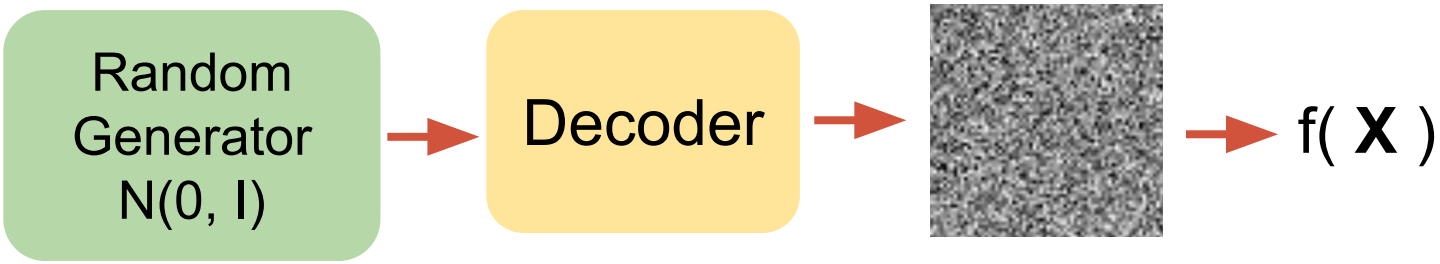
We'll be done if we can come up with a smooth function f, that takes in the output image x and outputs [0, 1] where:

Generating Realistic Faces



Given this f, we can compute its gradient with respect to the decoder parameters and use SGD to train. Done!

Generating Realistic Faces

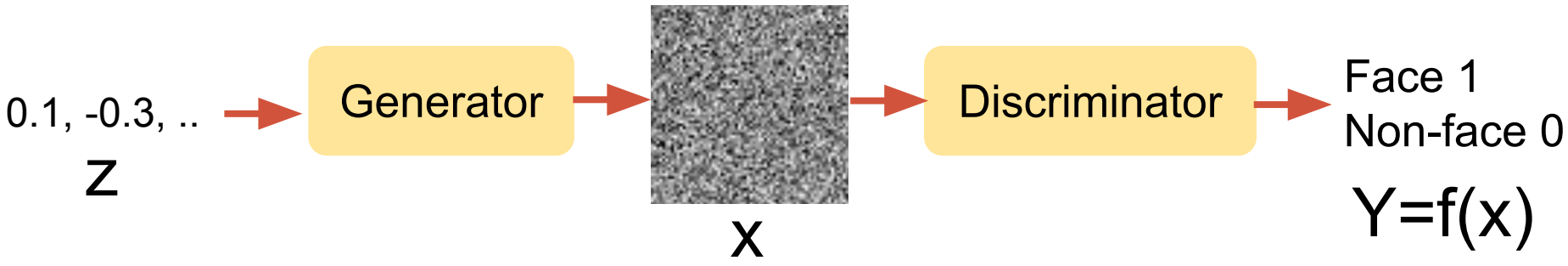


Solution: We will use another network as f. This network will predict the probability of x being a face.

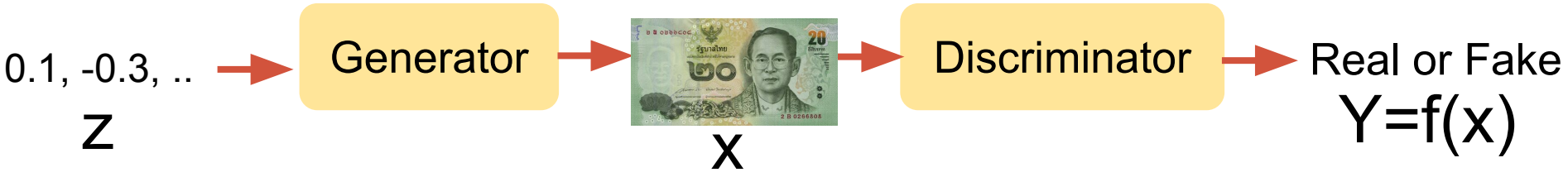
Training this f is a simple supervised learning!

All face photos -> positive examples
All non-face photos -> negative examples

Generating Realistic Faces



Generative Adversarial Networks (GAN)

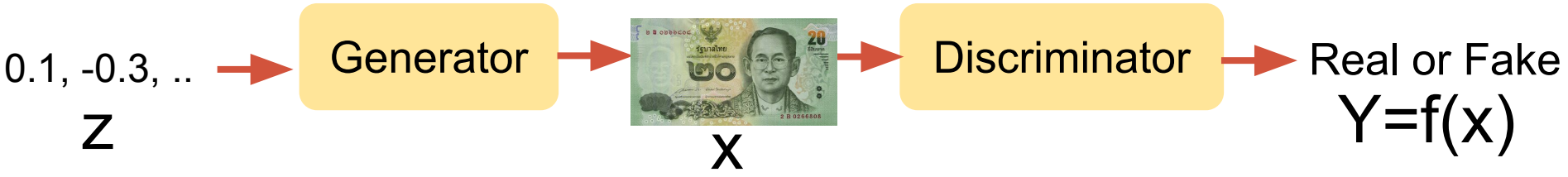


- Consider a money counterfeiter
  - He wants to make fake money that looks real
  - There's a police that tries to differentiate fake and real money.

- 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

Generative Adversarial Networks (GAN)



- Generator (Money Faker):
  - Maximize Y

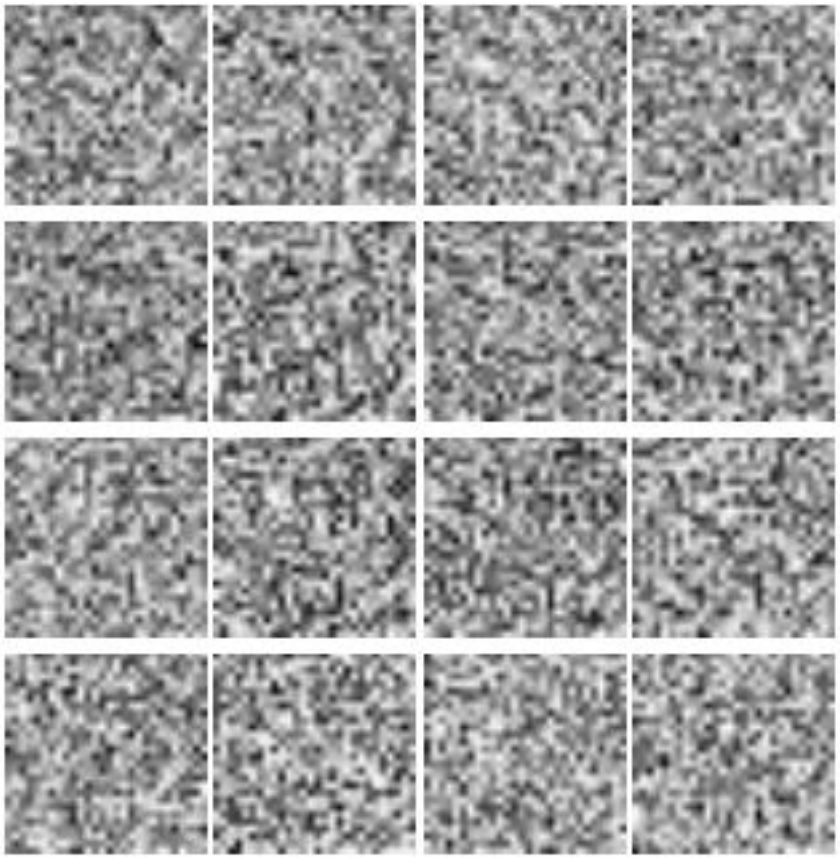
$$\min_G \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}))]$$

- Discriminator (Police):
  - For real images => Maximize Y
  - For generated images from the faker => Minimize Y

$$\max_D \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}))] + \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}}[\log(D(\mathbf{x}))]$$

GAN example

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



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

Progressive GANS

Progressive Growing of GANs for Improved Quality, Stability, and Variation [Karras et al. 2017]

GANs Loss Formulations

$$\max_D \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}))] + \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log(D(\mathbf{x}))]$$

Discriminator

$$\min_G \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}))]$$

Generator

GAN	Discriminator Loss	Generator Loss
MM GAN	$\mathcal{L}_{\text{D}}^{\text{GAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_{\text{G}}^{\text{GAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$
NS GAN	$\mathcal{L}_{\text{D}}^{\text{NSGAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$	$\mathcal{L}_{\text{G}}^{\text{NSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_{\text{D}}^{\text{WGAN}} = -\mathbb{E}_{x\sim p_d}[D(x)] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$	$\mathcal{L}_{\text{G}}^{\text{WGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
WGAN GP	$\mathcal{L}_{\text{D}}^{\text{WGANGP}} = \mathcal{L}_{\text{D}}^{\text{WGAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_g}[(\|\|\nabla D(\alpha x + (1-\alpha\hat{x})\|\|_2 - 1)^2]$	$\mathcal{L}_{\text{G}}^{\text{WGANGP}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
LS GAN	$\mathcal{L}_{\text{D}}^{\text{LSGAN}} = -\mathbb{E}_{x\sim p_d}[(D(x) - 1)^2] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})^2]$	$\mathcal{L}_{\text{G}}^{\text{LSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[(D(\hat{x} - 1))^2]$
DRAGAN	$\mathcal{L}_{\text{D}}^{\text{DRAGAN}} = \mathcal{L}_{\text{D}}^{\text{GAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_d+\mathcal{N}(0,c)}[(\|\|\nabla D(\hat{x})\|\|_2 - 1)^2]$	$\mathcal{L}_{\text{G}}^{\text{DRAGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$
BEGAN	$\mathcal{L}_{\text{D}}^{\text{BEGAN}} = \mathbb{E}_{x\sim p_d}[\|\|x - \text{AE}(x)\|\|_1] - k_t\mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \text{AE}(\hat{x})\|\|_1]$	$\mathcal{L}_{\text{G}}^{\text{BEGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \text{AE}(\hat{x})\|\|_1]$

Another problem: Mode collapsing

Are GANs Created Equal? A Large-Scale Study [Lucic et al. 2018]

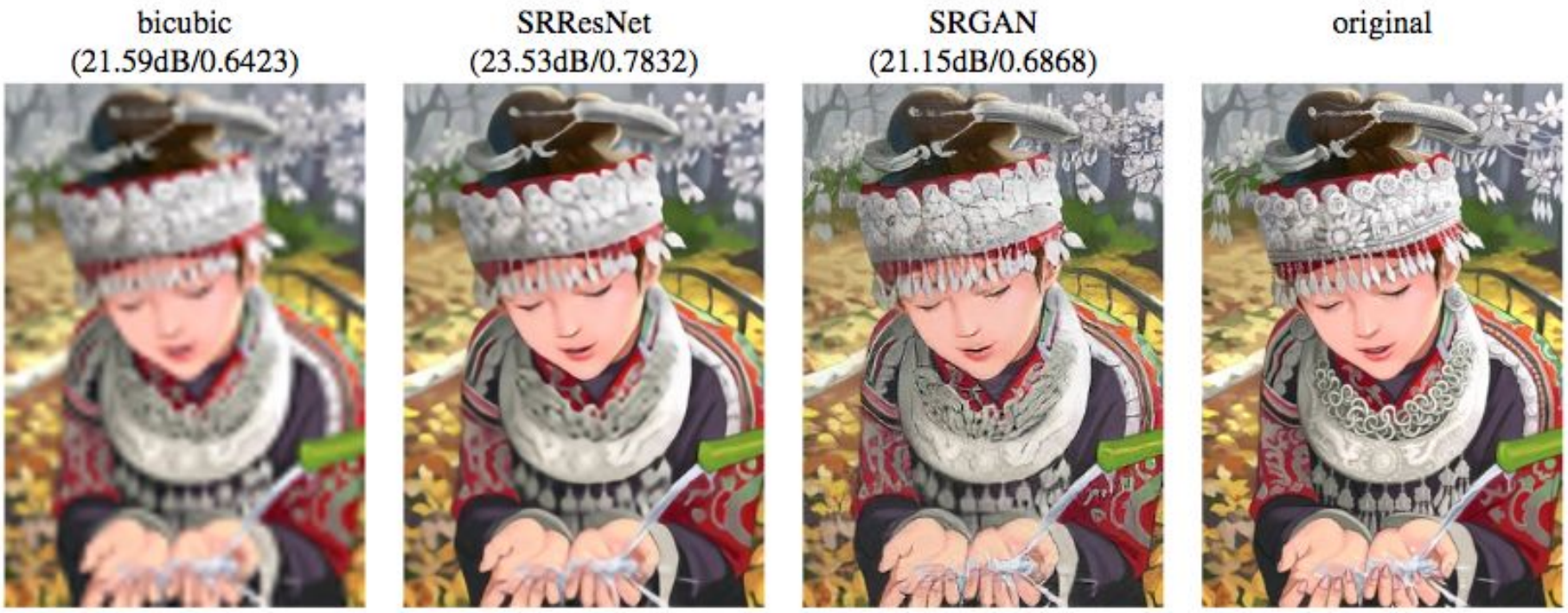
Other uses

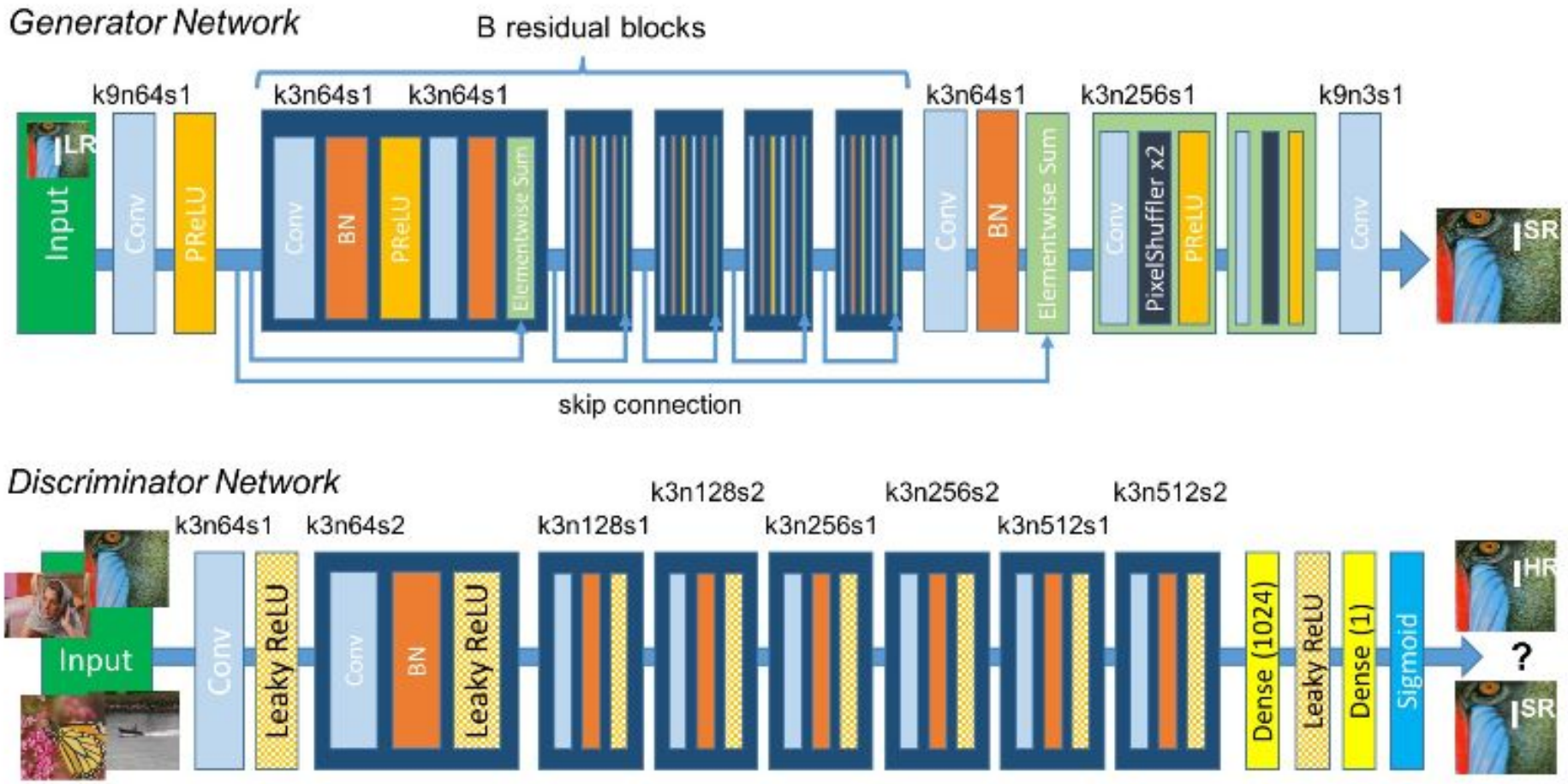
- StackGAN: text to photo

StackGAN: Text to Photo-realistic Image Synthesis with Stacked GANs [Zhang et al. 2017]

SRGAN

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

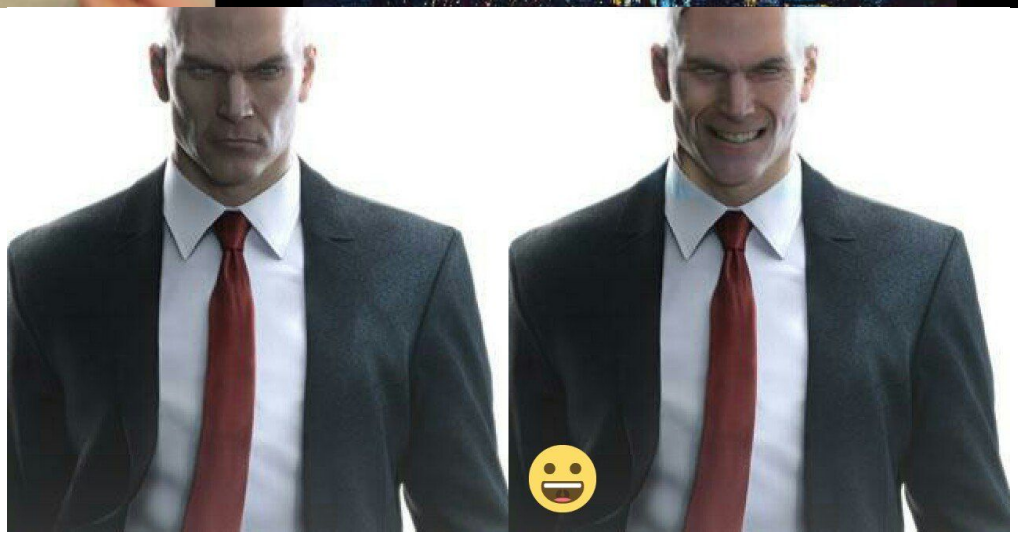



Generator Network
B residual blocks
k9n64s1
k3n64s1
k3n64s1
k3n64s1
k3n256s1
k9n3s1
LR
Input
Conv
PReLU
Conv
BN
PReLU
Elementwise Sum
Conv
BN
Elementwise Sum
Conv
PixelShuffler x2
PReLU
Conv
SR
skip connection
Discriminator Network
k3n64s1
k3n64s2
k3n128s1
k3n128s2
k3n256s1
k3n256s2
k3n512s1
k3n512s2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

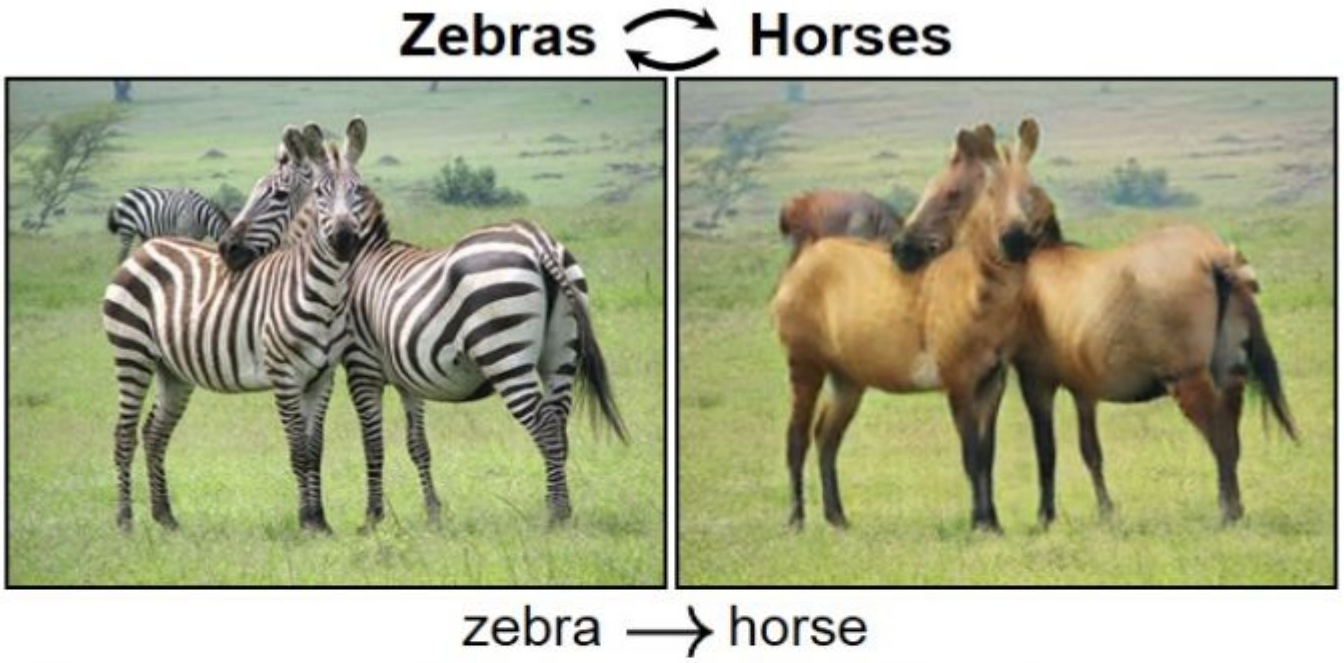
Content-aware Filling

Globally and Locally Consistent Image Completion [Iizuka et al., 2017]

Style transfer with cycleGAN

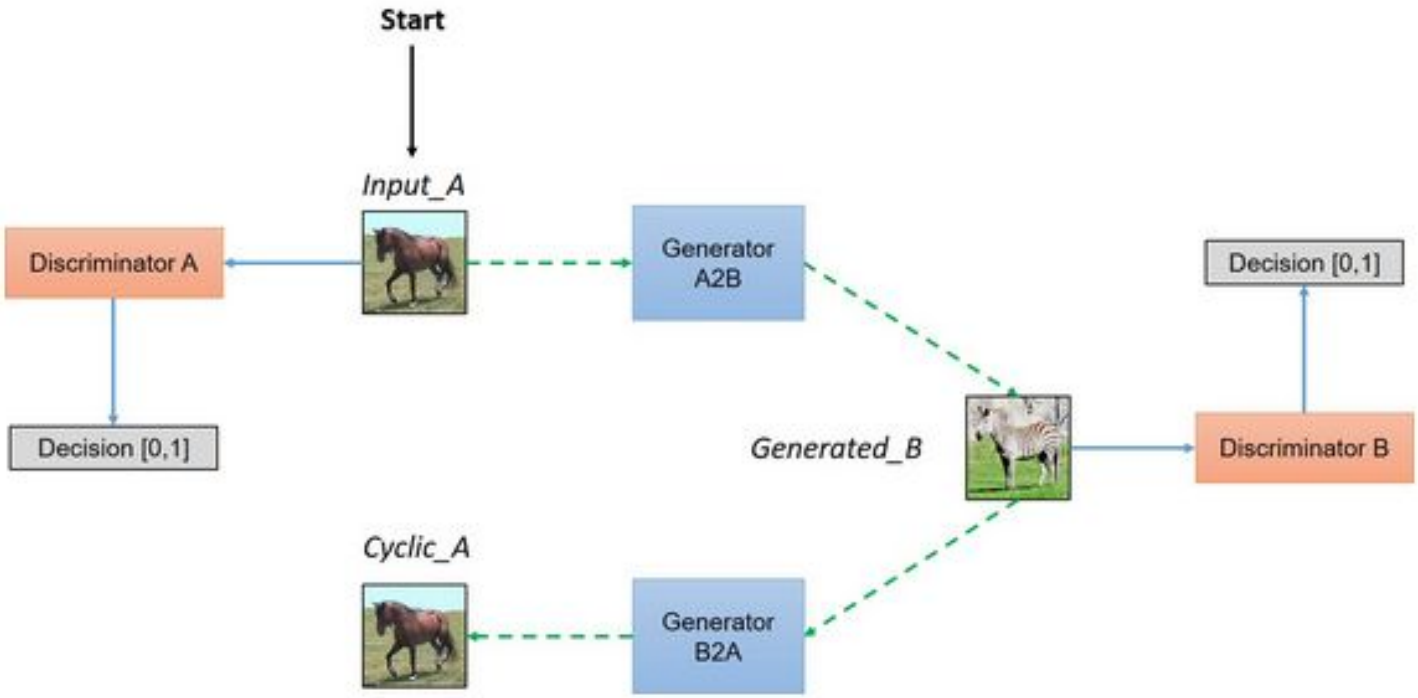


https://hardikbansal.github.io/CycleGANBlog/



Cycle consistency

Note, you don’t need a paired dataset



Cycle consistency

Machine translation with no parallel text

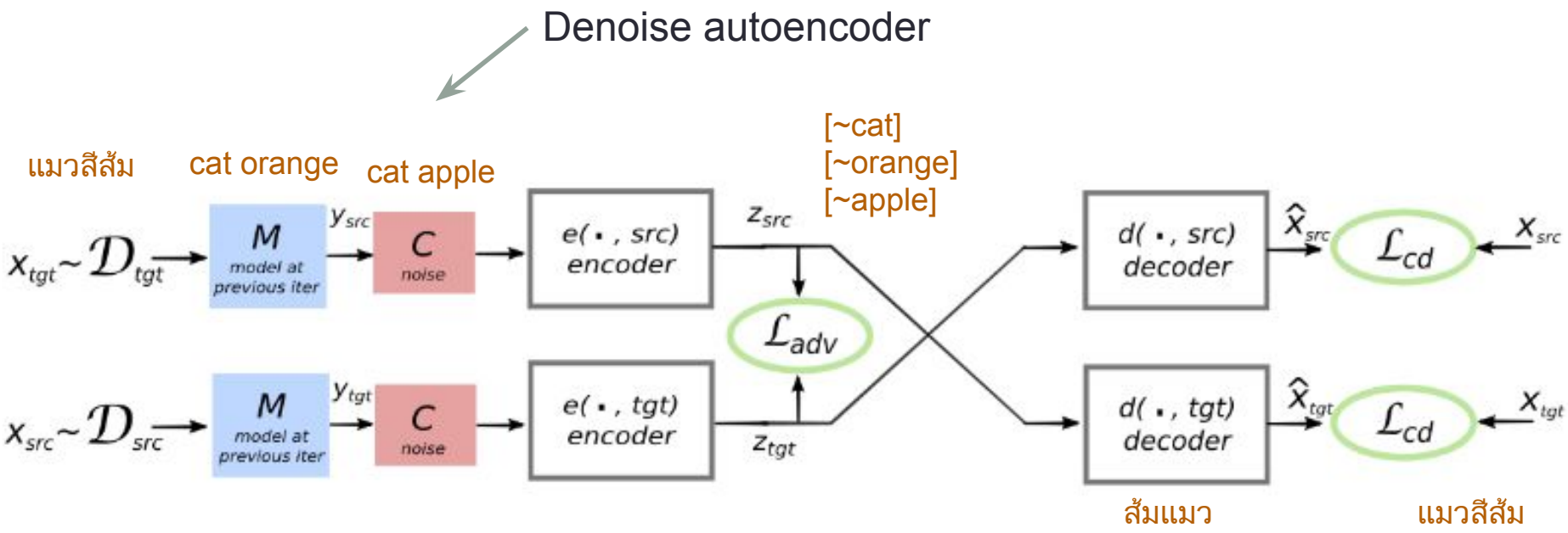
- MT usually requires parallel text

  This is a cat

  นี่คือแมว

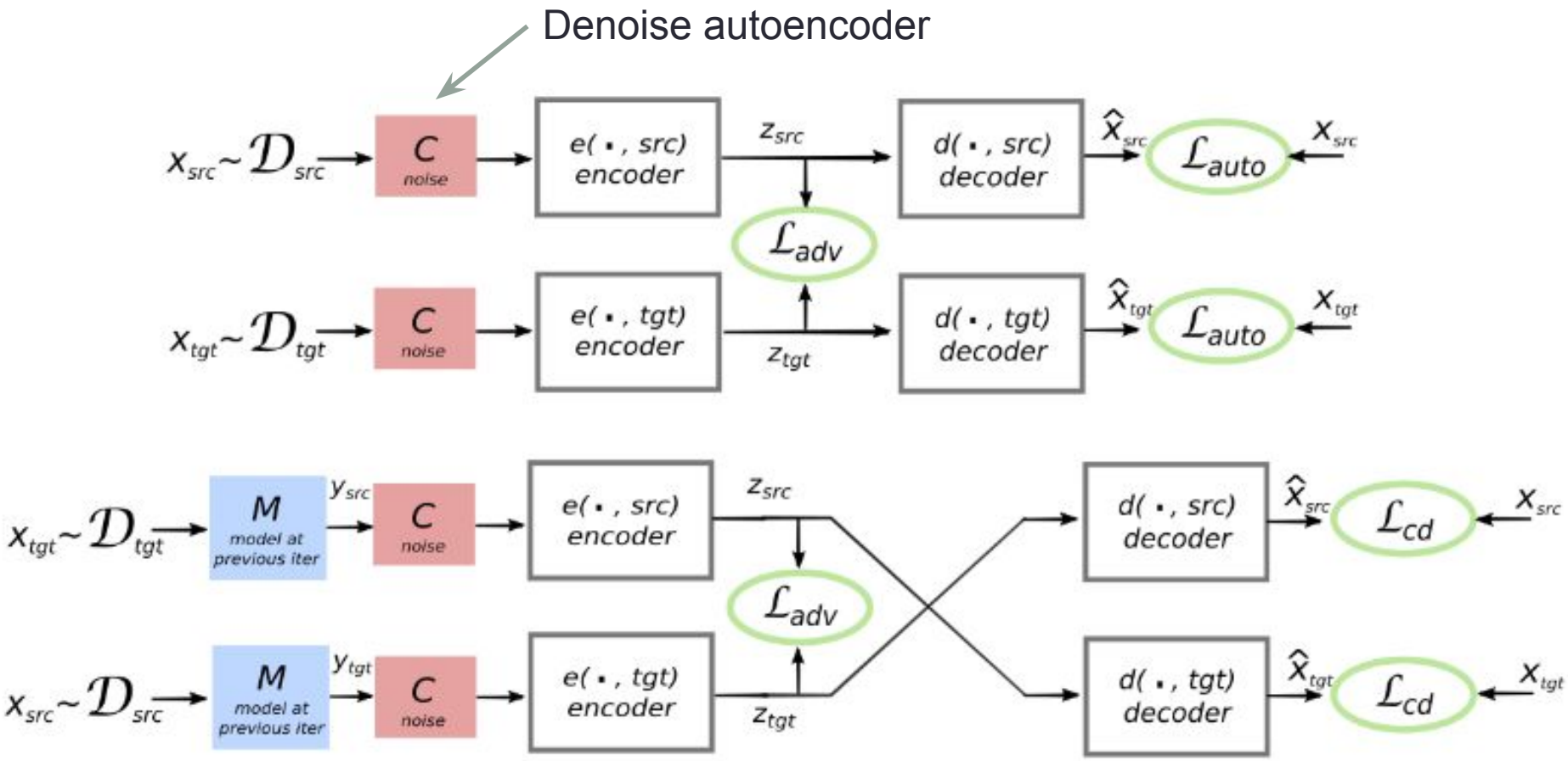
- Most of the time we don't have parallel text. Just text.
- Can we still do MT?
  - Use GANs + Autoencoders

https://arxiv.org/abs/1711.00043



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

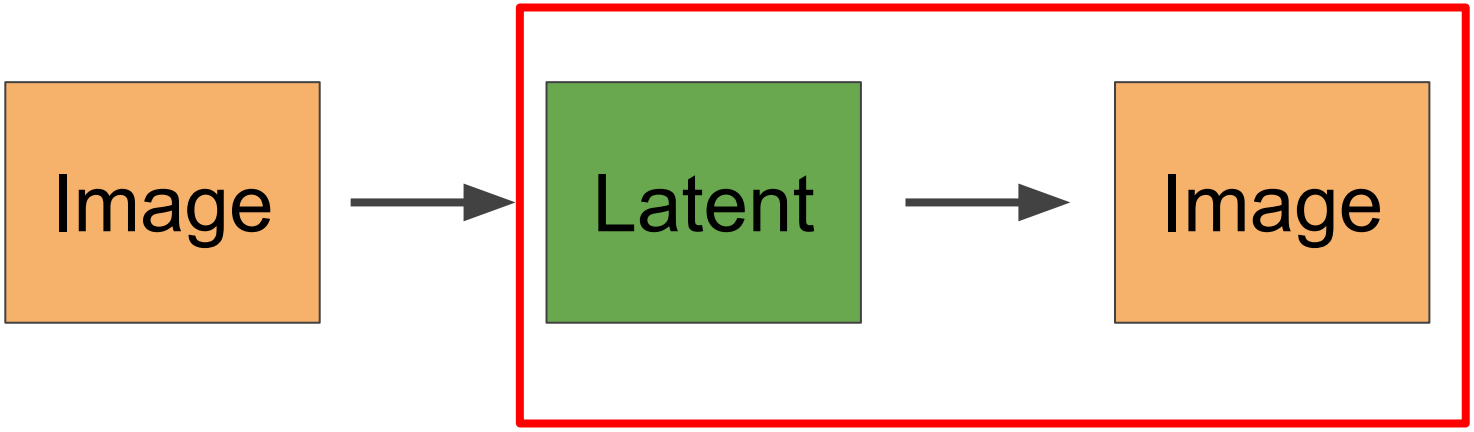
Results

	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

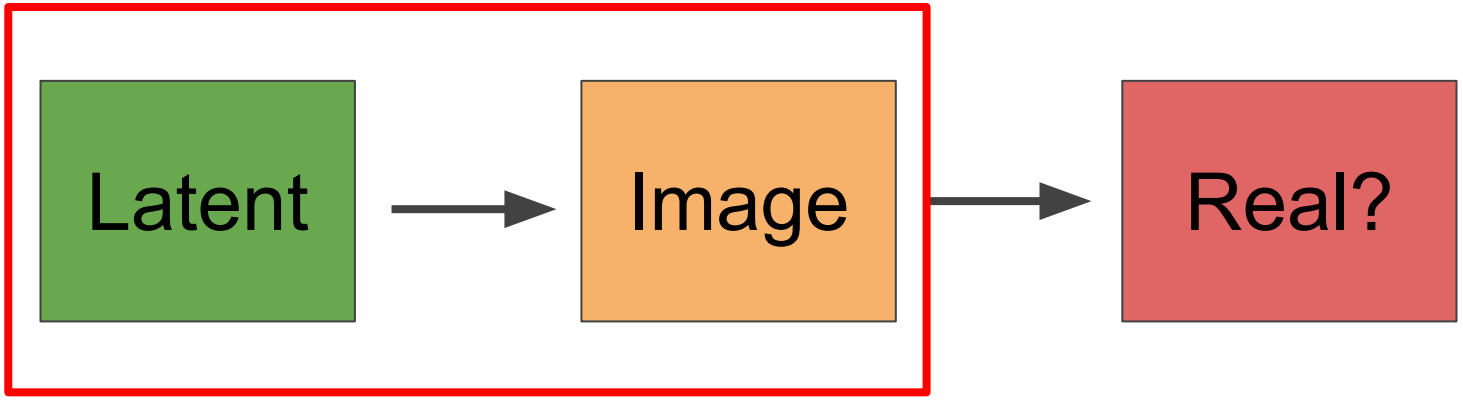
https://arxiv.org/abs/1711.00043

Autoencoder Frontier

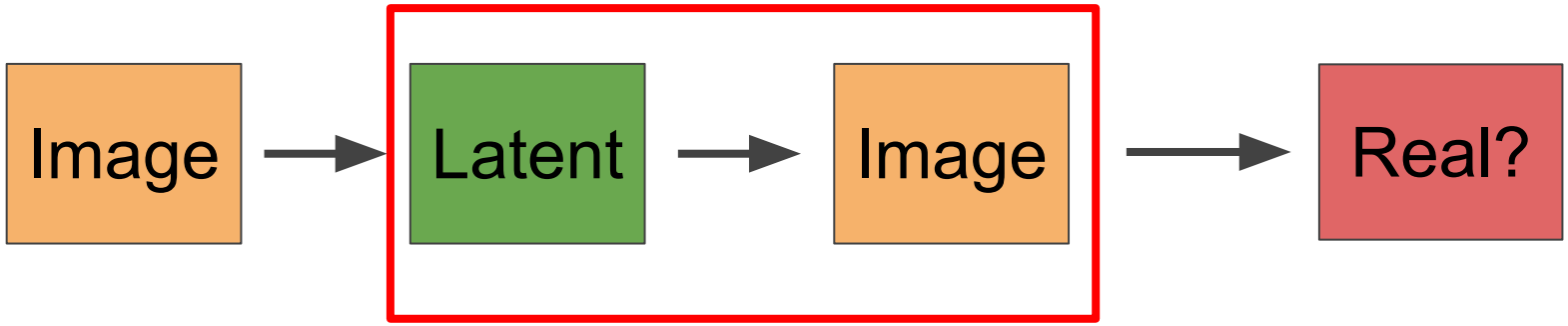
Auto-Encoder



GAN

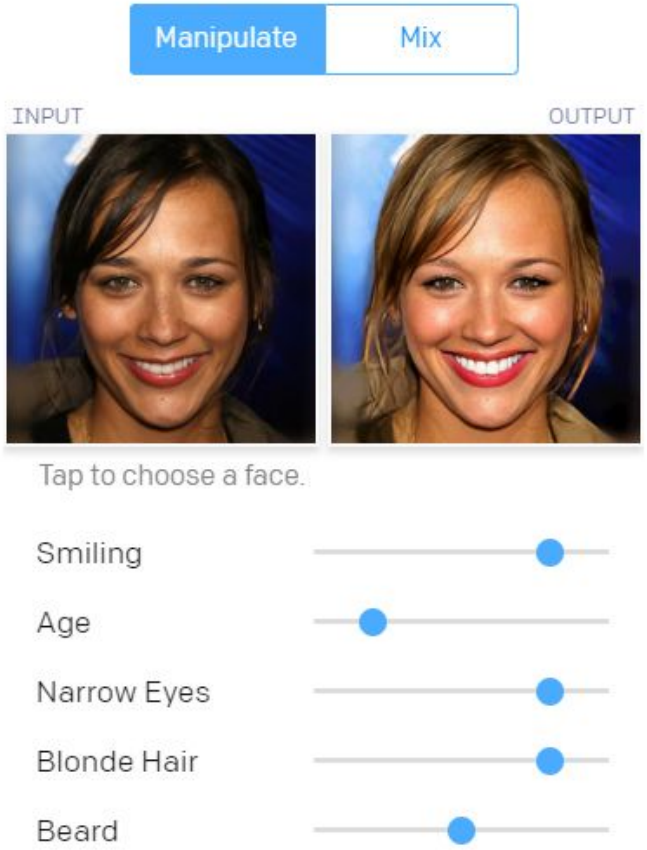


Auto-Encoder + GAN

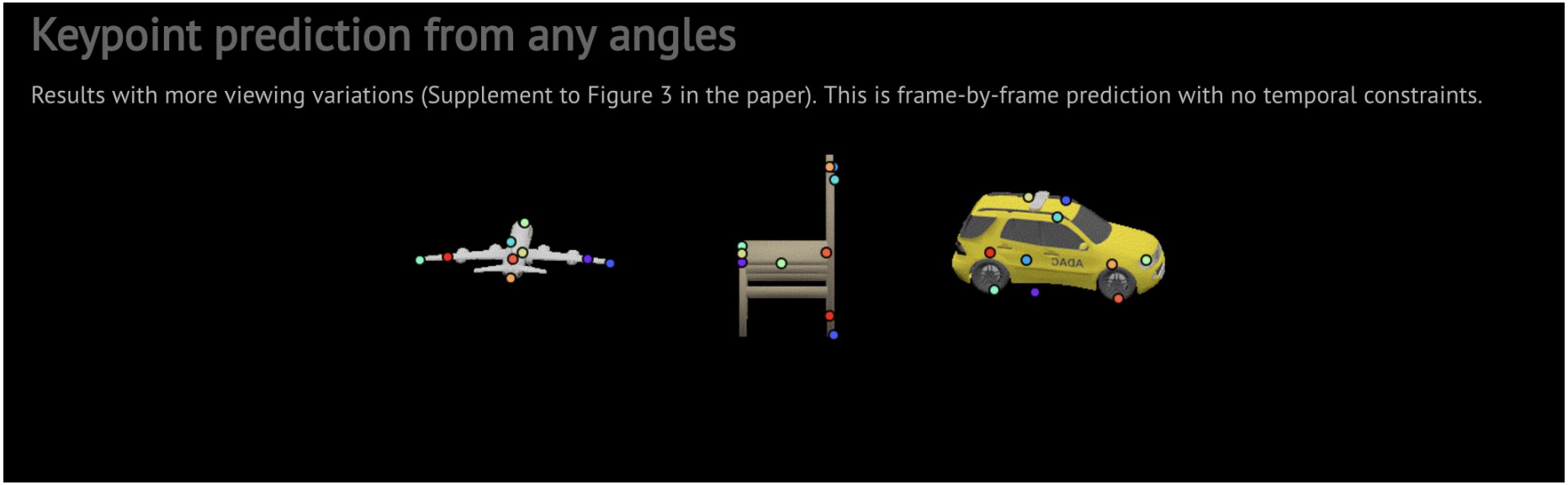


Glow

https://blog.openai.com/glow/



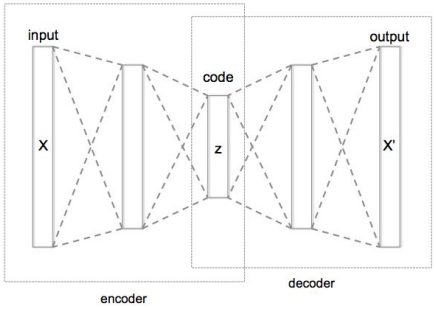
Unsupervised Keypoint Discovery

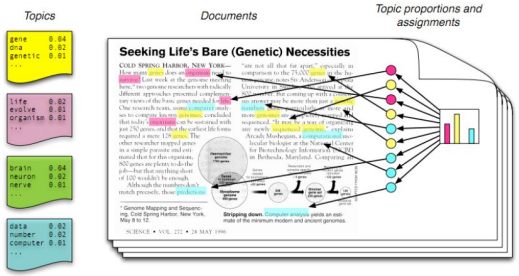


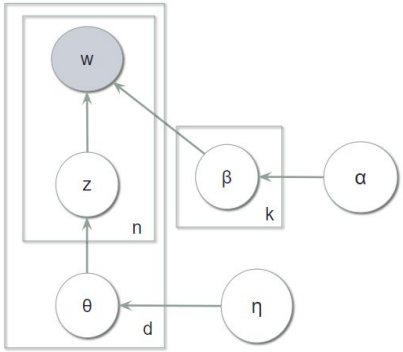
https://keypointnet.github.io/

Unsupervised Learning Summary

- Clustering
  - Introduction to probabilistic graphical models
  - LDA
- Autoencoders
  - Denoising Autoencoder
  - VAE
    - Evidence lower bound
- Generative Adversarial Networks
- Random Cool stuff







Reminder

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

Next next class: project presentation with TWO industry judges :)