Catherine Yeo

Week 1: Intro to GANs

Generative Models

- Discriminative model ⇒ classifier, models $P(\underset{\text{class}}{Y} \mid \underset{\text{features}}{X})$
- Generative model ⇒ creates realistic representation of a class, models $P(X \mid Y)$

  $\underset{\text{noise}}{\xi}, Y \rightarrow X$ Ex: Y = dog X = wet nose, tongue out, etc.

  ↳ Noise helps generate similar but diverse repr.s

- Types
  - Variational Autoencoders (VAE)

    Image → Encoder → Latent Space → Decoder → Reconstruct image

    Encoder: removed after training

    Latent Space: samples point on distr. to add noise

  - Generative Adversarial Networks (GAN) 2014

    Noise → Generator ≈ decoder → Image → Discriminator

    Another image: fake? real?

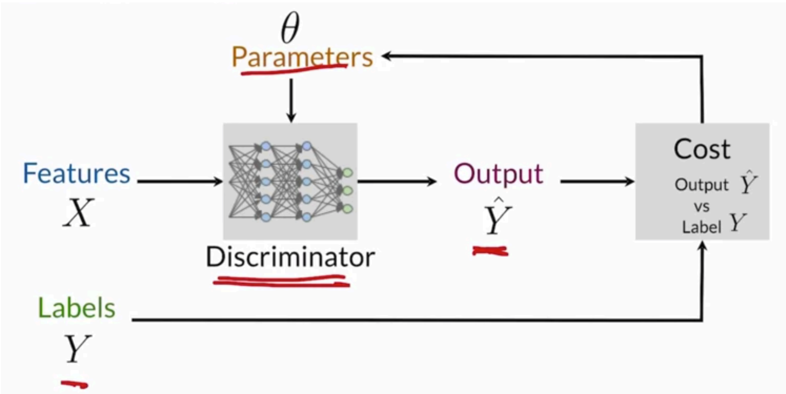
    Generator and Discriminator: always competing to reach realistic output ⇓ ADVERSARIAL (learn from competition)

    Discriminator: removed after training

Inside GANs

- Generator learns to make fakes look real ⇒ fool discriminator
  - Generates examples of class
  - Input ⇒ noise ξ vector to make outputs different
  - Generator wants $\hat{Y}$ to be 1 (real), discriminator wants $\hat{Y}$ to be 0
  - Once done competing, freeze θ params and save generator

- Discriminator learns to distinguish real from fake
  - Classifier (likely neural net)



→ BCE with labels for real (1) and fake (0)

BCE Cost Function

- Binary Cross Entropy

score ≈ 0 when good prediction, $-\infty$ otherwise

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))\right]$$

Prediction

relevant when label=1

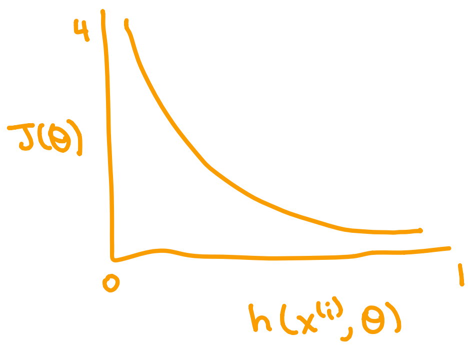
Label

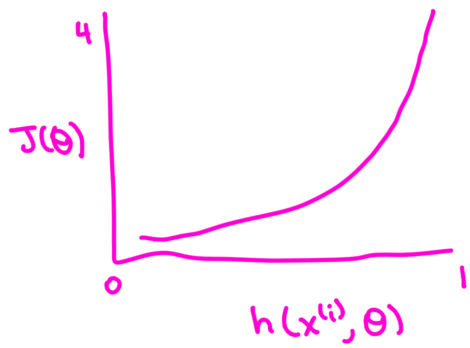
Features

... label = 0

Parameters

Average loss of the whole batch

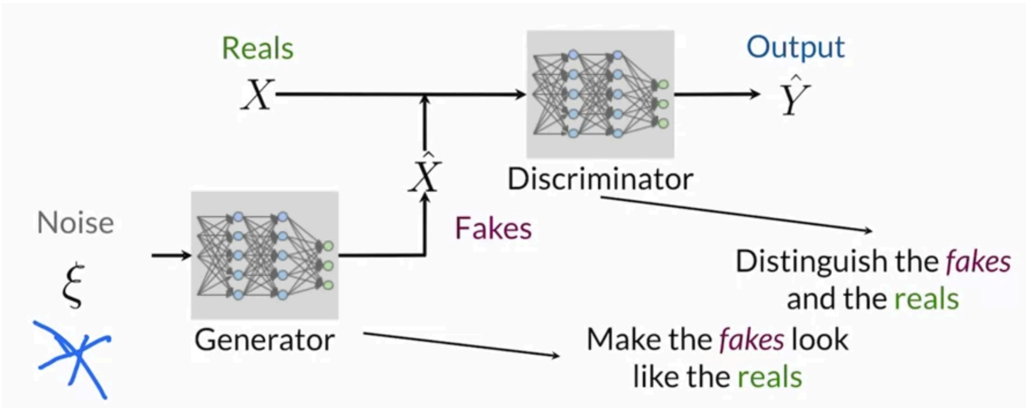




- Performed over mini-batch
- Generator wants to maximize cost ⟺ Discriminator wants to minimize cost

minimax

Recap



Both generator & discriminator should be at similar skill levels.

PyTorch Tutorial

- PyTorch is imperative (computes on the go) → dynamic, unlike TF (needs to compile) → static
    - Static → TF takes less time
    - TF 2.0 moving towards PyTorch
- Ex: defining a model

```python
import torch
from torch import nn

class LogisticRegression(nn.Module):
    def __init__(self, in):
        super().__init__()
        self.log_reg = nn.Sequential(
            nn.Linear(in, 1),
            nn.Sigmoid()
        )
    def forward(self, x):
        return self.log_reg(x)
```

Custom layers for DL

Define the model as a class

Initialization method with parameters

Definition of the architecture

Forward computation of the model with inputs x

- Ex: training a model

16 input vars

```python
model = LogisticRegression(16)

criterion = nn.BCELoss()

optimizer = torch.optim.SGD(model.parameters(), lr=0.01)

for t in range(n_epochs):
    y_pred = model(x)
    loss = criterion(y_pred, y)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

Initialization of the model

Cost function

Optimizer

Training loop for number of epochs

Forward propagation

Optimization step

WEEK 2: DEEP CONVOLUTIONAL GANs

Activations

- Used for classification b/t layers
- $z_i^{[l]} = \sum_{i=0}^{n} W_i^{[l]} a_i^{[l-1]} + b$

  $a_i^{[l]} = \underbrace{g}_{\text{activation function}}{}^{[l]}(z_i^{[l]})$

  activation function ⇒ must be $\begin{cases} \text{differentiable (for backprop)} \\ \text{non-linear (for layers)} \end{cases}$

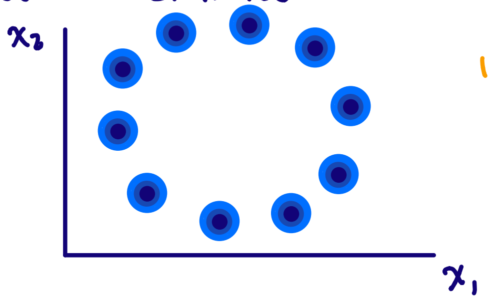
- Common functions
  - ReLU
  - Leaky ReLU
    - ↳ $\max(az^{[l]}, z^{[l]})$
      to solve dying ReLU problem (when deriv. stuck at 0)
  - Sigmoid $\xrightarrow{[0,1]}$ vanishing gradient problem
  - tanh $\xrightarrow{[-1,1]}$ same issues

WEEK 3: WASSERSTEIN GANS WITH GRADIENT PENALTY

→ This week: problems faced by GANs trained with BCE loss

Mode Collapse

- Mode in statistics

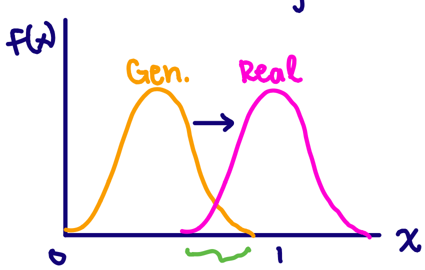


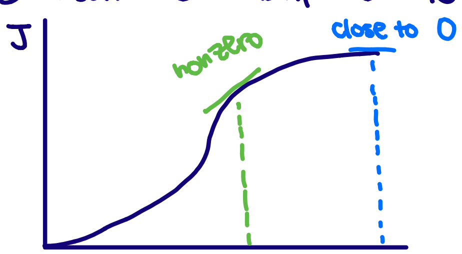
10 modes, 1 per MNIST digit

- **Mode collapse**
  - Ex: take discriminator that is good at identifying all except 1s and 7s.
    - ⇒ generator sees that discriminator misclassifies many 1s and 7s, so it generates many 1s and 7s.
    - ⇒ discriminator misclassifies all 1s
    - ⇒ generator only produces 1s ⇒ MODE COLLAPSE
  - Happens when generator gets stuck in one mode (a local minima)

Problems with BCE Loss

- GANs trained with BCE loss are prone to vanishing gradient problems
- GANs want generated & real distributions to look similar





Vanishing gradient

- Discr. → single output ; Gen. → complex output (e.g. image)
  easy to train ; difficult to train
- When discr. improves too much, approximated function by BCE loss → flat → useless feedback

Earth Mover's Distance (EMD)

- EMD = amount of effort to make generated distr. equal to real distr.
  - ↳ function of distance & amount
- Analogy: distr. 1 = dirt. How hard to move & mold dirt into real distr.?
- Gradient far from 0 when distributions are very different

Wasserstein Loss (W-Loss)

- BCE loss simplified:

$$\min_{\underset{\text{discr.}}{d}} \max_{\underset{\text{gen.}}{g}} -\left[E(\log(d(x))) + E(1-\log(d(g(z))))\right]$$

- W-Loss approximates EMD → c = critic (W-Loss's version of discr.)

$$\min_{\underset{\text{discr.}}{d}} \max_{\underset{\text{gen.}}{g}} E(\underbrace{c(x)}_{\text{real}}) - E(\underbrace{c(g(z))}_{\text{fake}})$$

- Doesn't have to have sigmoid layer b/c no longer capped b/t 0 and 1
- Comparison

BCE Loss	W-Loss
Discriminator outputs between 0 and 1	Critic outputs any number
$-[\mathbb{E}(\log(d(x))) + \mathbb{E}(1-\log(d(g(z))))]$	$\mathbb{E}(c(x)) - \mathbb{E}(c(g(z)))$

hence solve vanishing gradient

- Condition on W-Loss: critic needs to be 1-Lipschitz continuous (1-L cont.)

$$\|\nabla f(x)\|_2 \leq 1$$

→ norm of gradient ≤ 1 at every point

  - How to enforce 1-L continuity:
    - Weight clipping: forces weights of critic to a fixed interval
      - ↳ limits critic's learning ability
    - Gradient penalty: add λ reg (regulariz. term) to W-Loss
      - ↳ penalizes critic when gradient norm > 1

· Ta-da ~

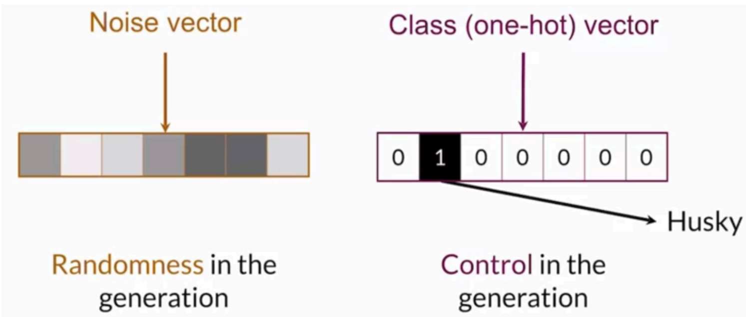
interpolated image

$$\min_{g} \max_{c} \; E(c(x)) - E(c(g(z))) + \lambda E(\|\nabla c(\hat{x})\|_2 - 1)^2$$

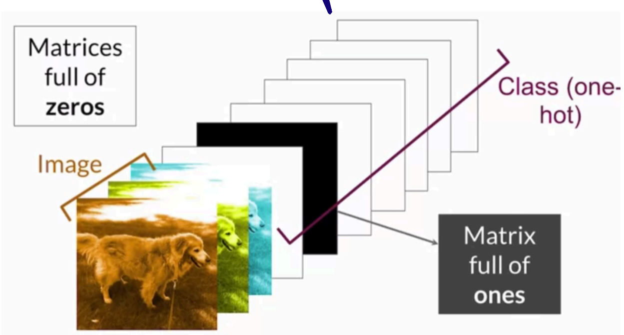
Week 4: Conditional GAN & Controllable Generation

Conditional Generation

- Unconditional generation: you get outputs from a random class
- Conditional generation: you get what you ask for from the class you specify
  ↳ training data needs to be labelled
- Inputs
  - Generator input: one-hot concatenated to noise vector



  - Discriminator input: one-hot matrices as a channel



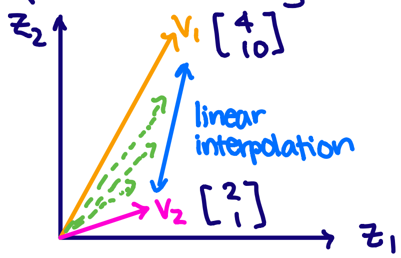
Controllable Generation

- Controlling features after model has been trained
  - Tweak input noise vector Z

Controllable	Conditional
Examples with the **features that you want**	Examples from *the classes you want*
Training dataset **doesn't need to be labeled**	Training dataset *needs to be labeled*
Manipulate the z vector input	*Append a class vector* to the input

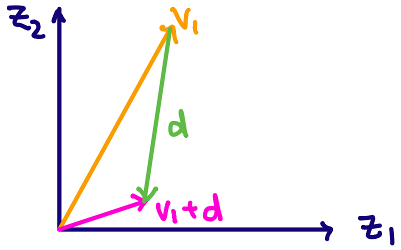
Vector Algebra in Z-Space

- Interpolation using Z-space



$g(v_1) \xleftrightarrow{\text{intermediate images}} g(v_2)$

- Move in Z-space to modify features by finding vector directions



$g(v_1) \rightarrow$

$g(v_1+d) \rightarrow$

  - Finding that direction using classifier gradients
  - Only update noise vector

Challenges

- Feature correlation ⇒ may lead to too many (correlated) features modified
- Z-space entanglement ⇒ not possible to control single output features
  - Happens when z doesn't have enough dimensions, so z-values don't correspond to clear mappings on images
- Disentanglement
  - If disentangled, every z element corresponds to a feature
    ↳ Latent factors of variation
  - Changes to 1 feature do not affect others
  - Methods
    - Add labels to data
    - Use a reg. term