# GANs -Andrew

* Week 1 Build Basic GAN
  + Course1: Generative model

Discriminative Models:

Used for classification, features -> class, P(Y|X)

Generative Models:

Learn to produce realistic objects, Noise + class -> features

e.g. Variational Autoencoders (VAE):

Encoder -> latent space -> decoder

e.g. Generative Adversarial Networks (GAN):

generator <- discriminator (fake vs real)

* + Course2: Real life GANs

Application:

Create new images never existed (StyleGAN2)

Image translation (GauGAN)

Animation of static picture

3D object (3D\_GAN) medicine, design

Companies:

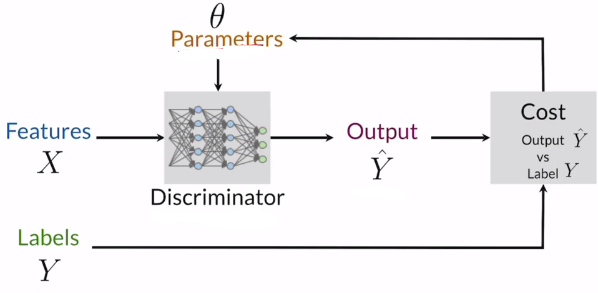
Adobe, IBM, Google, Byte dance, Disney

* + Course3: Institution behind GANs



* + Course4: Discriminator

Classifier, Neural Network, P(real/fake|features)



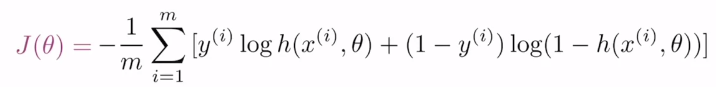
* + Course5: Generator

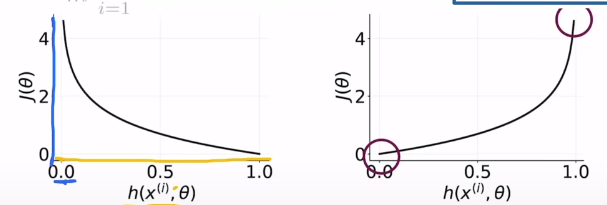
Noise vector -> NN -> features -> discriminator -> cost function

P(feature|class) -> approximate distribution of real objects

* + Course6: BCE Cost Function

Binary cross entropy: y-label, h-prediction, m-length of presentation,

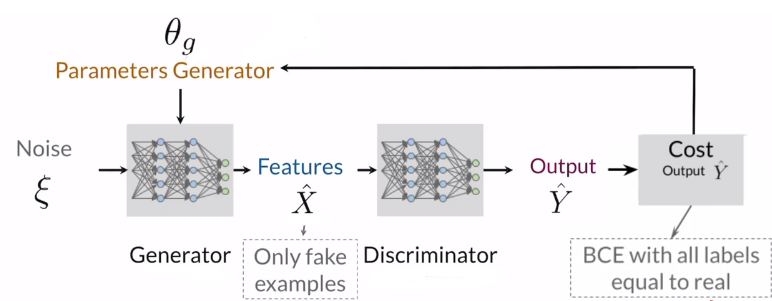




Two parts: one for label 0, and the other for 1.

Close to 0 when prediction and label are similar, otherwise infinity.

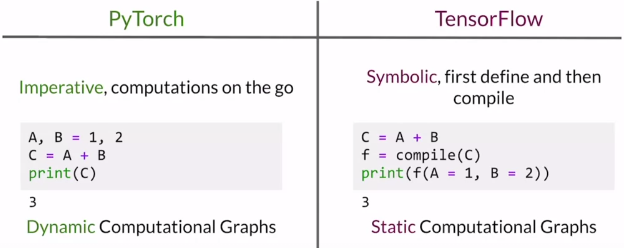
* + Course7: Putting it all together



Train in an alternating fashion

Two models should be at similar skill level

* + Course8: Intro to Pytorch



* Week 2 Deep convolutional GAN
  + Course1: Activations (basic properties)

Non-linear differential function:

Differentiable for backpropagation

Nonlinear for complex features, otherwise linear regression

* + Course2: Common Activations

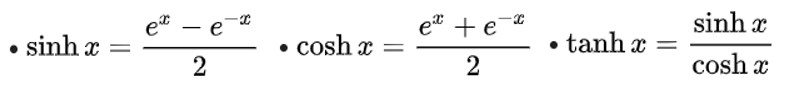
Sigmoid, ReLU, Leaky ReLU, Tanh

Sigmoid: vanishing gradient and saturation problem (0, 1)

ReLU: rectified linear unit

Leaky ReLU: add negative weight; avoid dying ReLU problem

Tanh(hyperbolic): keeps sign of input (-1, 1)



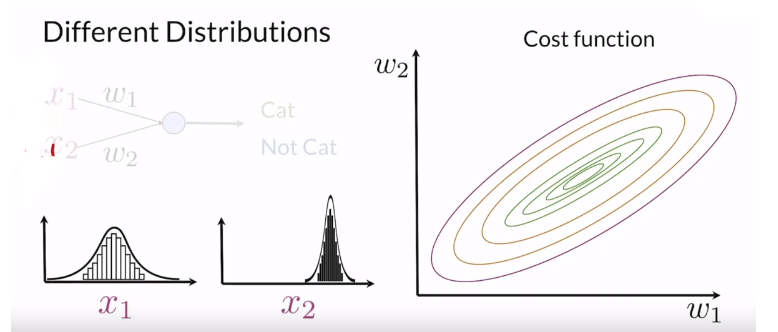
* + Course3: Batch Normalization

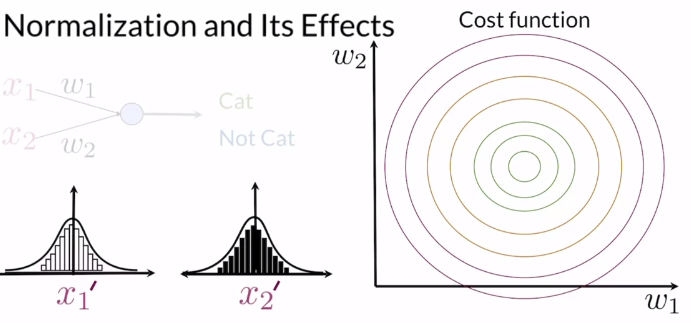
How it helps:

Reduce Covariant shift: change distribution and speed up training

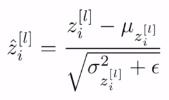
Around mean at 0 and std. at 1

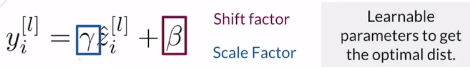
Smooth cost function





Procedure:

 μ: mean, σ: std



use the batch mean and standard deviation during training and the running statistics (that was computed over the entire training set) for testing. The running values are fixed after training.

* + Course4: Review of Convolution

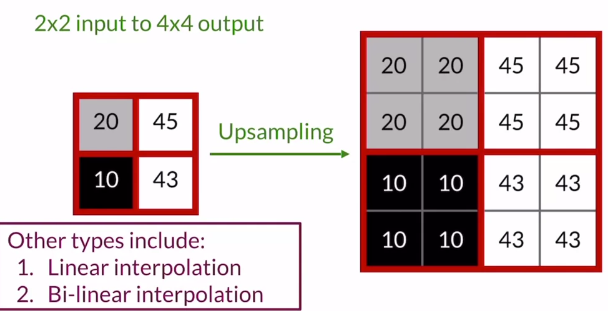
Pass

* + Course5: Padding and Stride

Pass

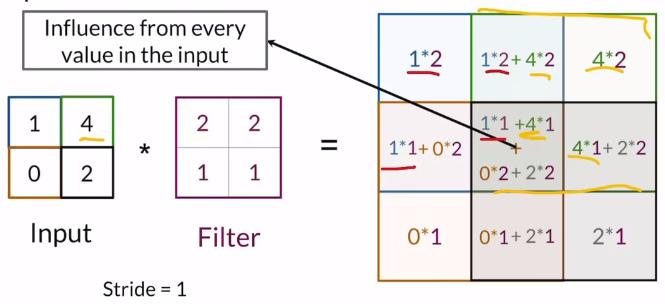
* + Course6: Pooling and Upsampling

Upsampling:



Upsampling increases input size; pooling does the opposite.

* + Course7: Transposed Convolutions(deconvolution)



Learns a filter to upsample

Problem: checkerboard pattern

Has learnable parameters

<https://distill.pub/2016/deconv-checkerboard/>