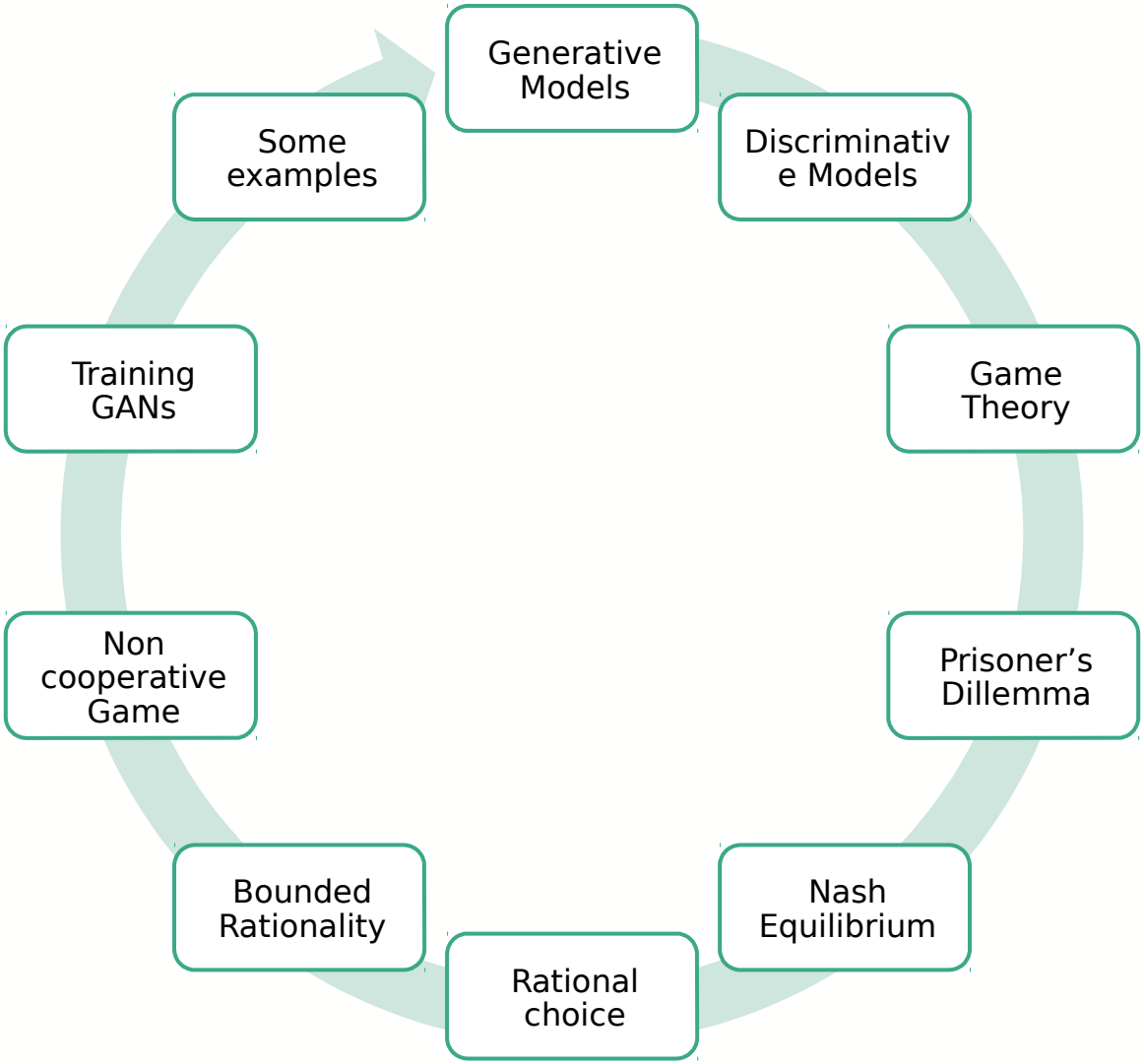
DEEP LEARNING BRASIL
SUMMER SCHOOL

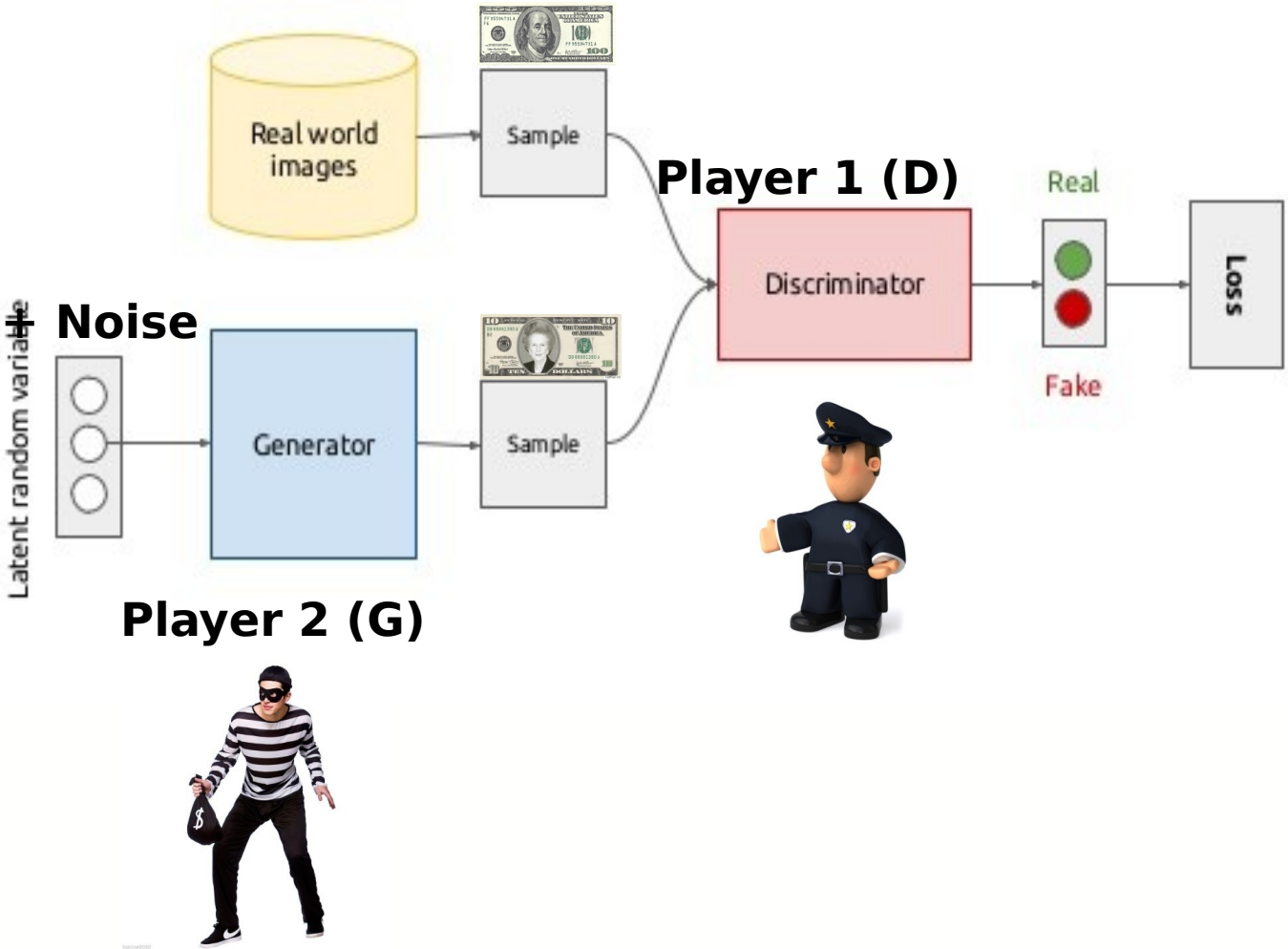
Generative Adversarial Networks

Rubens Zimbres

Data Scientist

Machine Learning for IoT at Vecto Mobile


Generative Models
Discriminative Models
Game Theory
Prisoner's Dillemma
Nash Equilibrium
Rational choice
Bounded Rationality
Non cooperative Game
Training GANs
Some examples

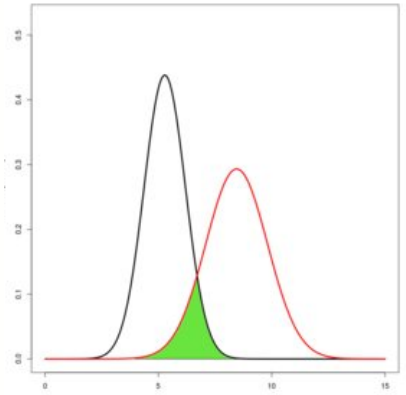

Real world images
Sample
Player 1 (D)
Real
Discriminator
Loss
Fake
Latent random variable
+ Noise
Generator
Sample
Player 2 (G)

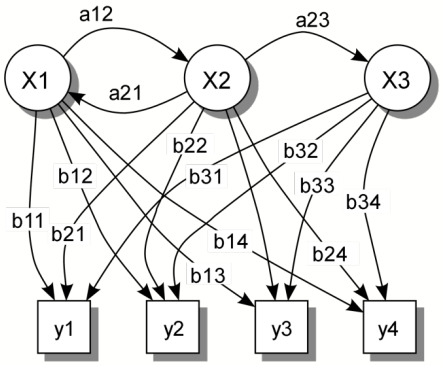
Generative Models

- Probabilistic, samples from joint distribution P(x,y)
- Generate samples (latent variables) following a distribution
- Maximum Likelihood
- Range of values belong to a class

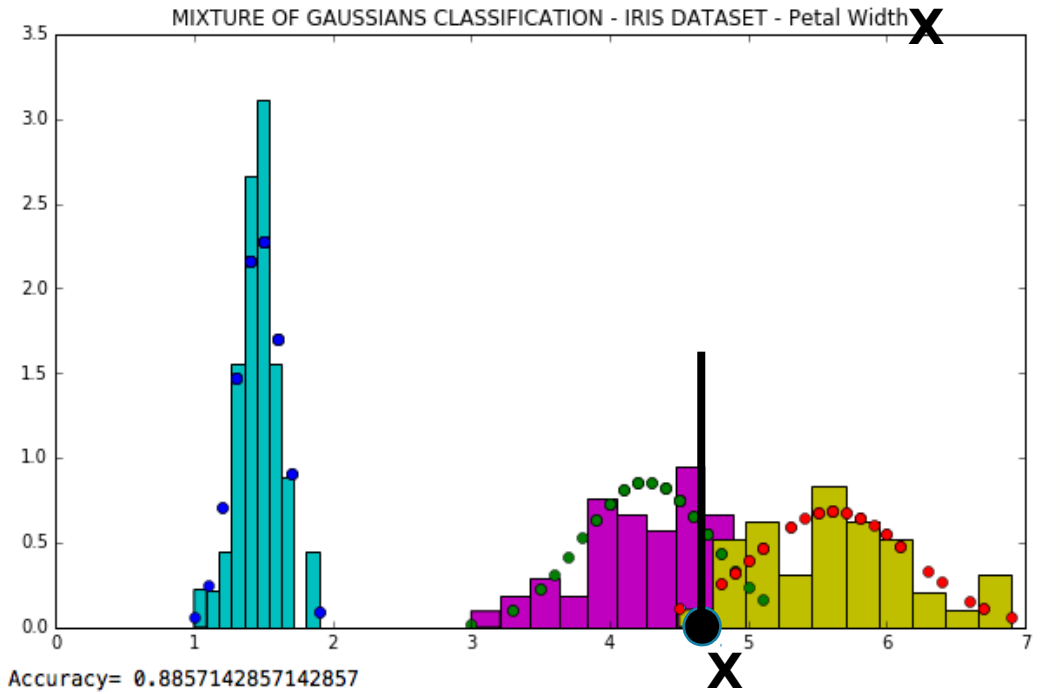
Generative Models

- Latent Dirichlet Allocation (NLP)
- Mixture Gaussians
- Hidden Markov Model
- VAE
- RBM
- GANs

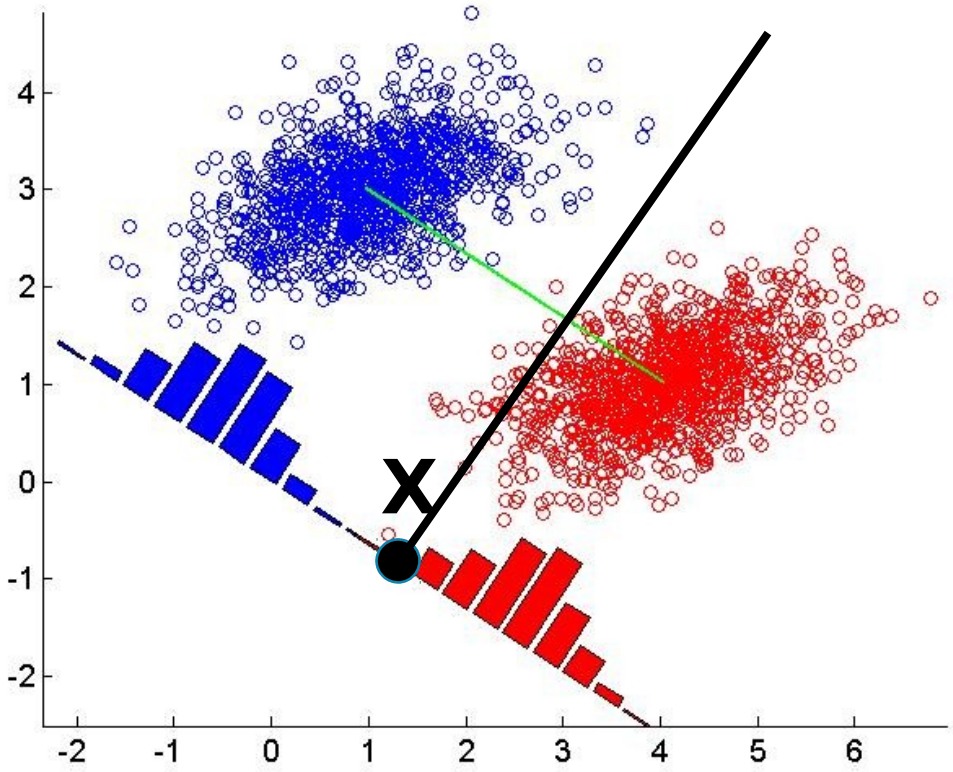




Mixture of Gaussians

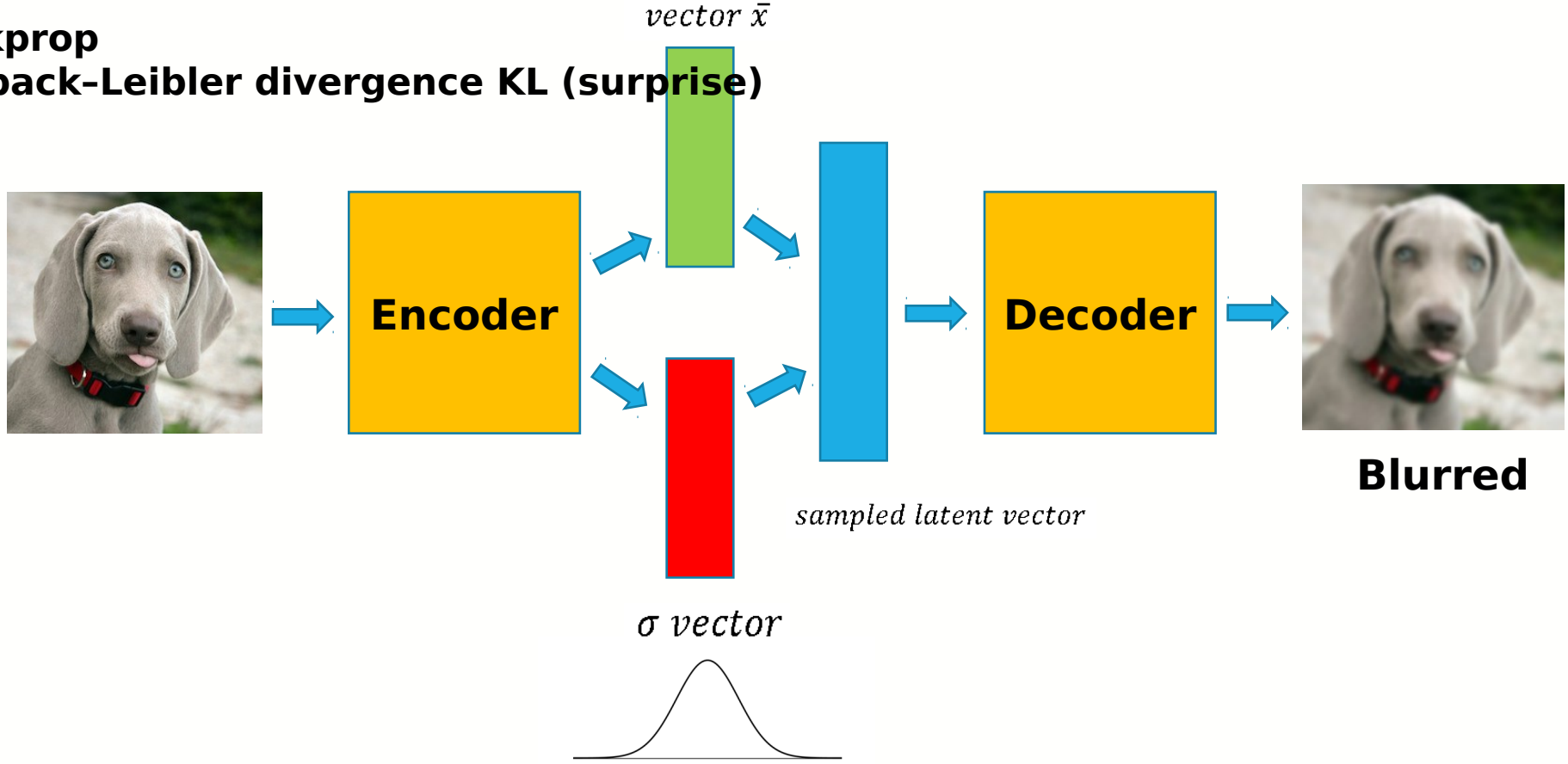


Linear Discriminant Analysis

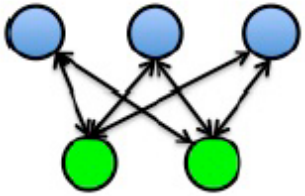


Variational Autoencoders

ckprop
llback–Leibler divergence KL (surprise)



Restricted Boltzmann Machines



$$E(v,h) = -\sum v_i h_j w_{ij}$$

Energy; Weight; Binary state of visible; Binary state of hidden

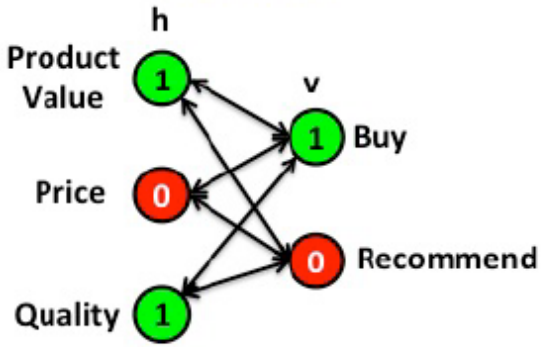
$$p(v,h) = \frac{e^{-E(v,h)}}{\sum e^{-E(u,g)}}$$

Probability of joint configuration; Energy of joint configuration; Energy of the system

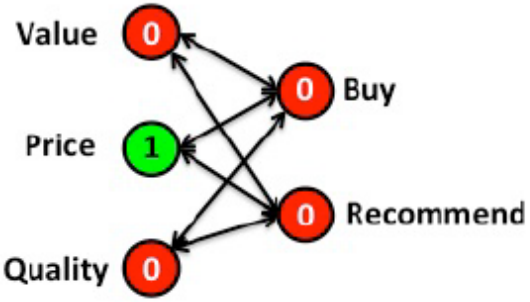
RBM Learning (update in parallel)

$$\Delta w_{ij} = \eta \cdot \frac{\partial}{\partial x} \log\left(p(v)\right) = \eta \cdot (< v_i h_j >^0 - < v_i h_j >^1)$$

EXAMPLE

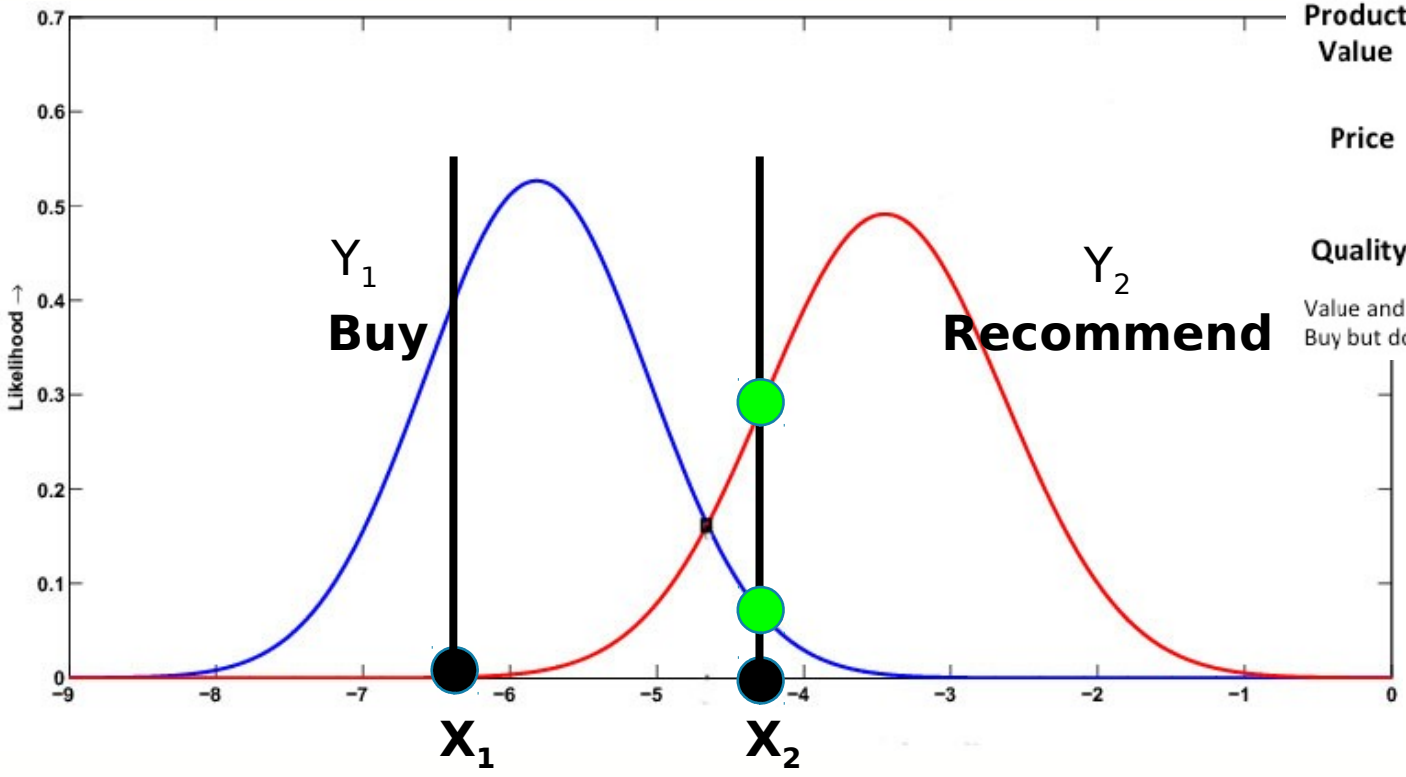


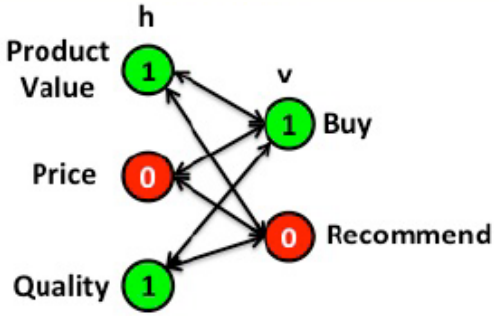
Value and Quality above average, but too expensive
Buy but does not Recommend to friends



Poor Value and Quality and affordable
Does not Buy and does not Recommend to friends

Generative Models





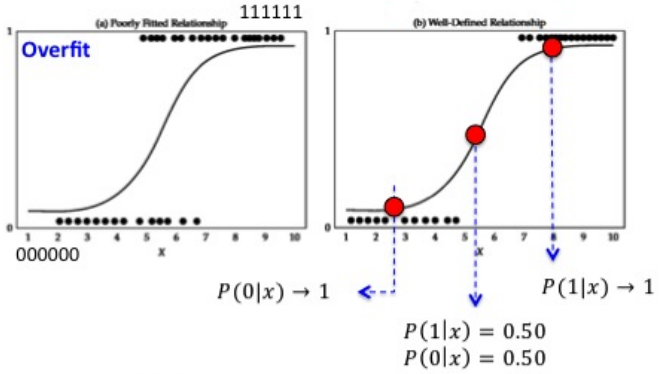
Value and Quality above average, but too expensive
Buy but does not Recommend to friends

Discriminative Models

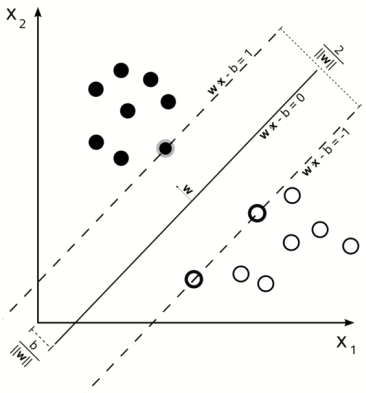
- Observed variables: target, infer outputs, conditional
- Function maps from x to Y
- Conditional density function
- P(Y|x) subsample
- Decision boundary
- Computationally expensive

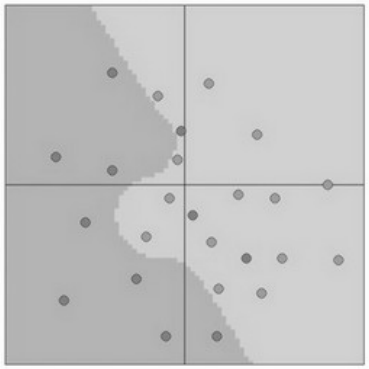
Discriminative Models

- Logistic Regression



- SVMs





- Neural Networks

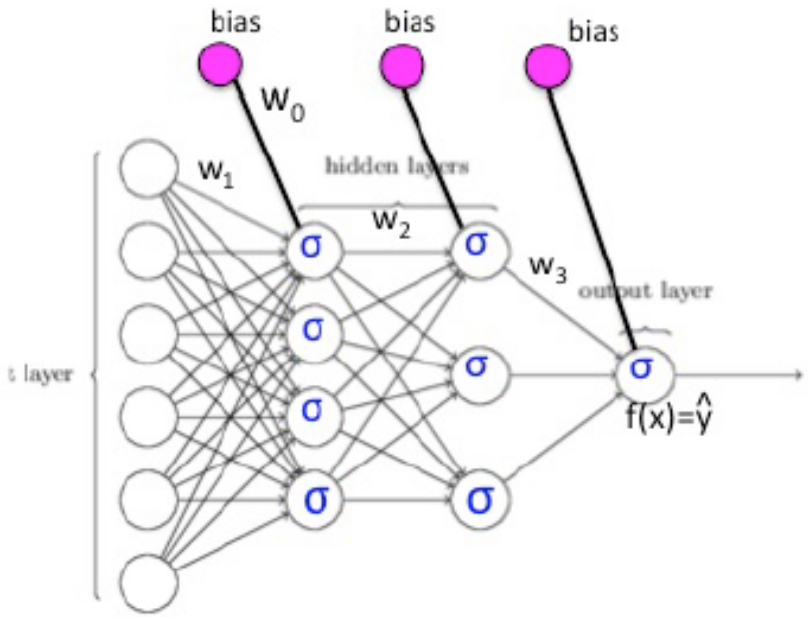
$$Odds\ Ratio = log\left(\frac{P}{1-P}\right) = mx + b$$

$$Odds = \left(\frac{P}{1-P}\right) = e^{mx+b}$$

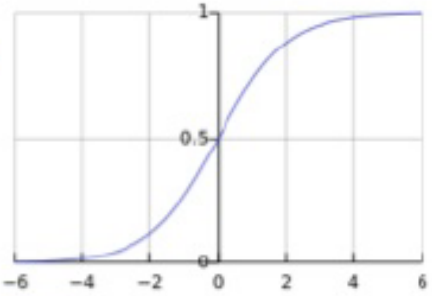
Cost function Logistic Regression

$$J(\theta) = -\frac{\sum y.\log(\hat{y}) + (1-y).\log(1-\hat{y})}{n}$$

$$where\ \hat{y} = \frac{1}{1+e^{mx+b}}$$



Sigmoid Activation Function σ



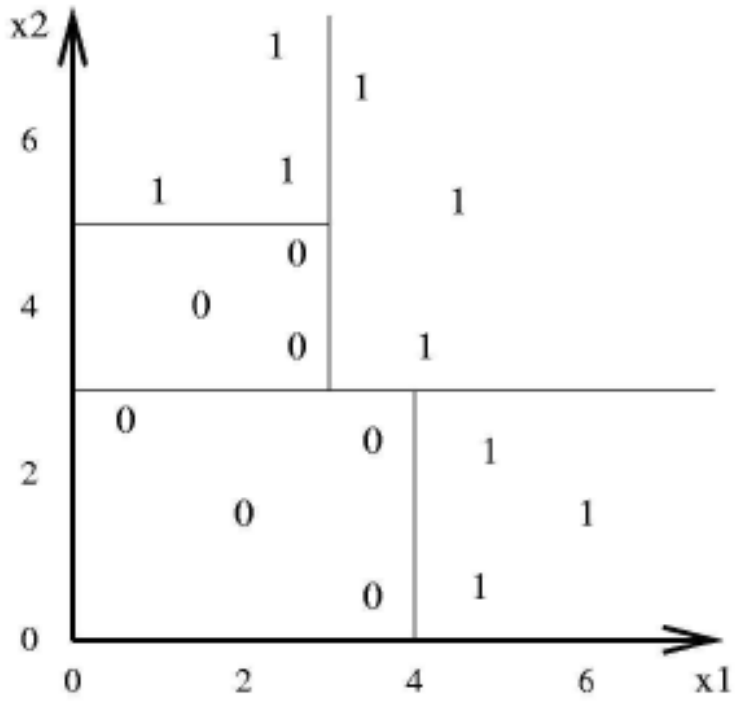
Cost function Neural Networks

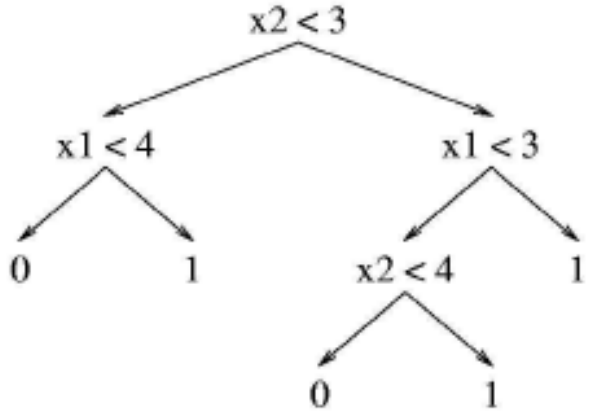
L2 Regularization term

$$J_\theta = \frac{\sum_{i=1}^{n}\sum_{k=1}^{k} t_k.\log(o) + (1-t).\log(1-o)}{N} + \frac{\lambda\sum_{l=1}^{ll}\sum_{i=1}^{jj}\sum_{j=1}^{jj+1}\theta_{ji}^2}{2N}$$

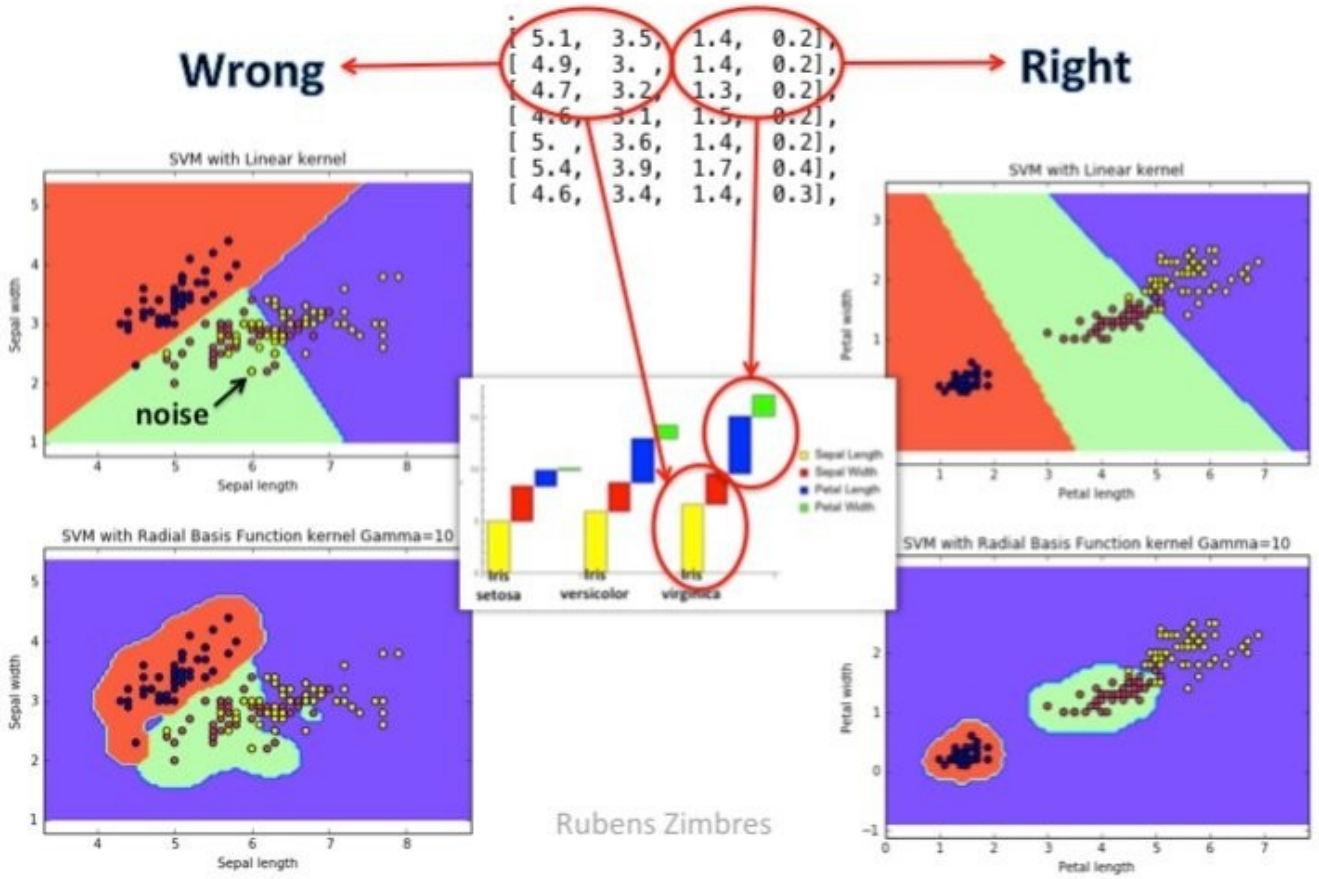
Where y = t = Target and y = o = Output

Decision Trees





Occam's razor Principle



Generative vs Discriminative

- Generative learns joint probability distribution = P(x,y)

$$\frac{Probability}{\sum samples}$$

- Discriminative learns conditional probability distribution = P(y|x)

$$\frac{Probability}{subsamples}$$

GAN Advantages over Generative Models

- D & G = Multi Layer Perceptron
- Regular backpropagation like VAE (G and D differentiable)
- Does not necessarily involve Maximum Likelihood estimation
- No need to use Markov Chains
- Subject but robust to overfitting

Game Theory

- 2 Player Game – Zero Sum
- Minimax solution: Nash Equilibrium
- Unique solution: D=1/2 everywhere

		PLAYER B	
		COOPERATE	DEFECT
PLAYER A	COOPERATE	A: 1 year jail B: 1 year jail	A: 10 years jail B: 0 years jail
	DEFECT	A: 0 years jail B: 10 years jail	A: 5 years jail B: 5 years jail

PAYOFF MATRIX			
		Discriminator	
		Cooperate	Defeat
Generator	Cooperate	(1,1)	(0,1)
	Defeat	(1,0)	(0,0)

Nash Equilibrium

- Sub optimal
- Non cooperative
- Emulate human behavior: bounded rationality (Herbert Simon)
- Simultaneous but not equivalent ** (GANs)

GANs

- Semi Supervised Learning: missing labels

- Inverse Reinforcement Learning

MARKOV DECISION PROCESS

$$U_s = R_s + \delta \max_a \sum_s T(s,a,s').U(s')$$

$$\pi_s = \operatorname{argmax}_a \sum_s T(s,a,s').U(s')$$

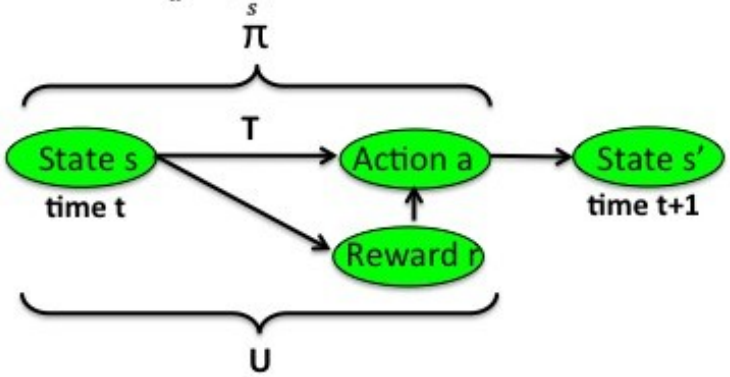
$$Q_{s,a} = R_s + \delta \max_{s'} \sum_s T(s,a,s'). \max_{a'} Q(s',a')$$

$$\hat{Q}_{s,a} \leftarrow_\eta R_s + \delta \max_a Q(s',a')$$

Reinforcement Learning

Markov Decision Process

$$\pi_s = \underset{a}{\operatorname{argmax}} \sum_{s} T(s,a,s').U(s')$$



$$U_s = R_s + \delta \max_{a} \sum_{s} T(s,a,s').U(s')$$

Net Present Value

Prospect Theory
Expected Utility Theory
Rational Choice Theory
Game Theory

Where:
s = state at time t
T = transition function
U = expected utility
a = action A(s)
r = reward R(s), R(s')
s' = state at time t+1
δ = discount rate (NPV)
π = policy

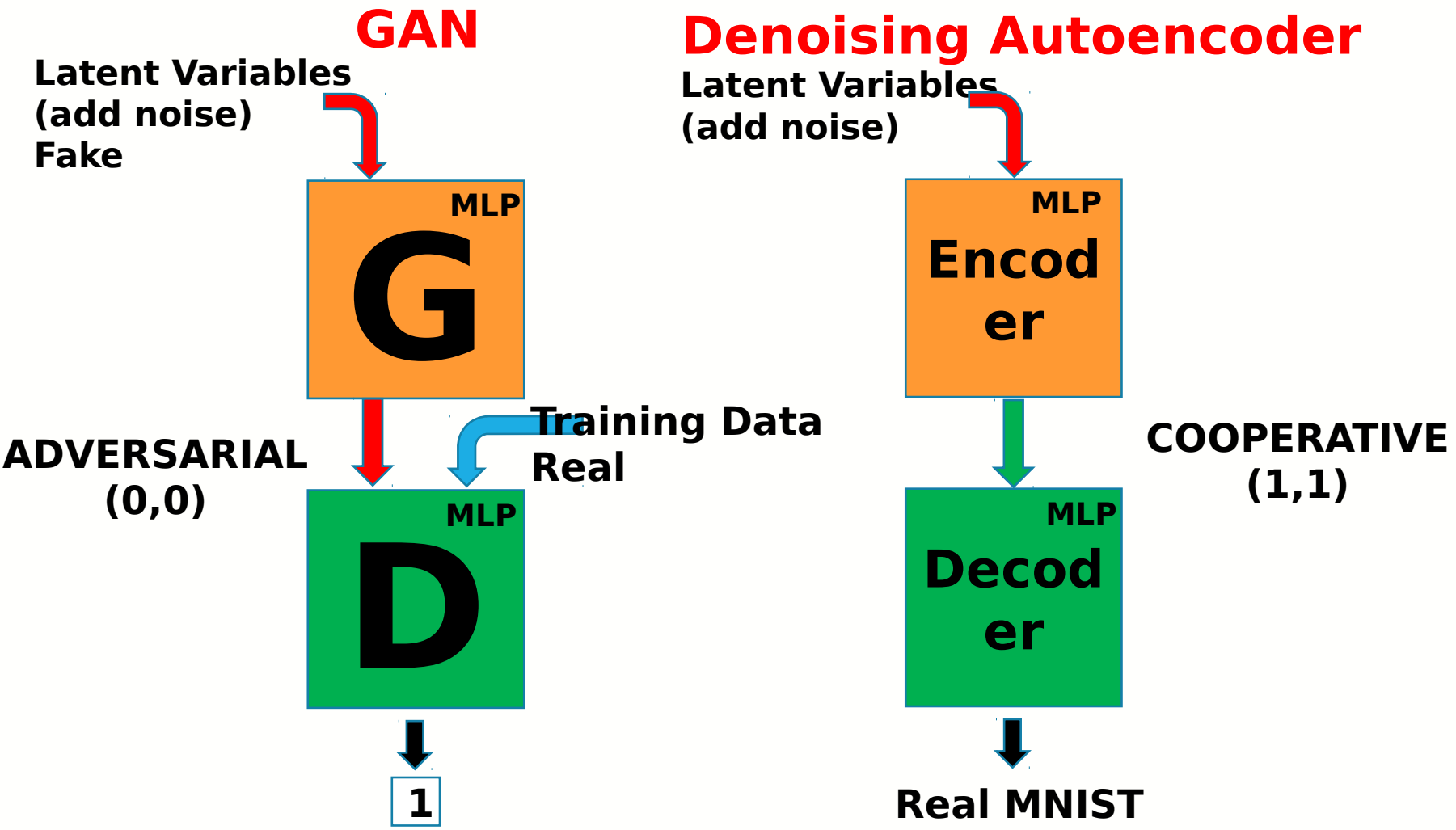
Game Theory

Prisoner's Dilemma

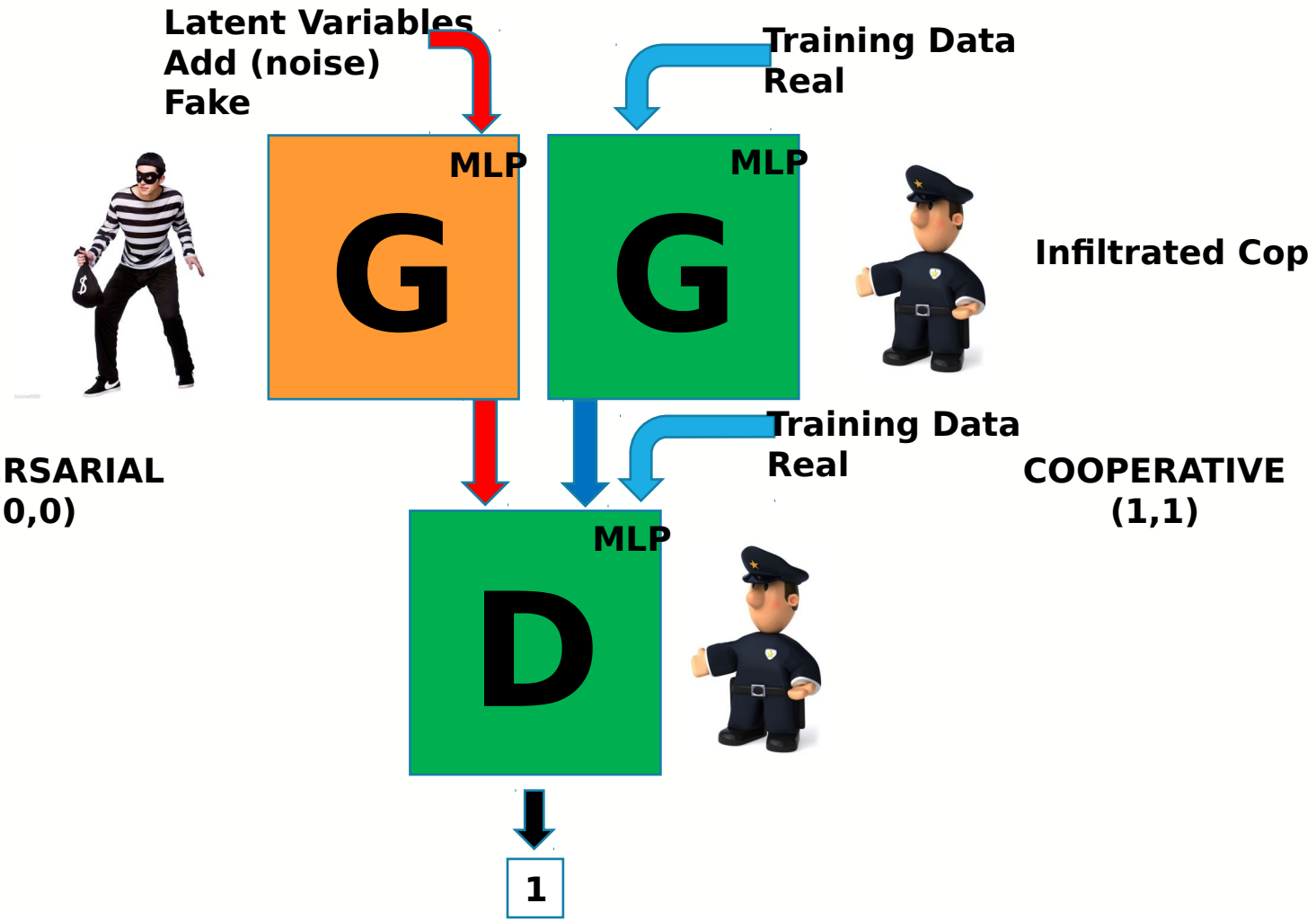
		B: Cooperate	B: Defeat
A	Cooperate	(-1,-1)	(-8,0)
A	Defeat	(0,-8)	(-5,-5)

Static: Nash Equilibrium:
- "Best" strategy chosen by all players at same time
- No reason to change strategy
- Inefficient: (-1,-1) is better
- Herd behavior

Dynamic: Iterative
- Pavlovian conditioning
- Tit-for-tat (Cold War times)
- Mutual Assured Destruction
- Mixed strategy (probability)


GAN
Latent Variables
(add noise)
Fake
MLP
G
Training Data
Real
ADVERSARIAL
(0,0)
MLP
D
1
Denoising Autoencoder
Latent Variables
(add noise)
MLP
Encoder
COOPERATIVE
(1,1)
MLP
Decoder
Real MNIST

Siamese GAN


Latent Variables
Add (noise)
Fake
Training Data
Real
MLP
G
MLP
G
Infiltrated Cop
ADVERSARIAL
(0,0)
Training Data
Real
COOPERATIVE
(1,1)
MLP
D
1

GAN Tuning

- Neural Network architecture
- Hyperparameters
- Game Theory strategies

Neural Networks

24 Adjustements

ARCHITECTURE

- Variables type
- Variable scaling
- Cost function
- Neural Network type:
  - RBM,FFN,CNN,RNN...
- Number of layers
- Number of hidden Layers
- Number of nodes
- Type of layers:
  - LSTM, Dense, Highway
  - Convolutional, Pooling...
- Type of weight initialization
- Type of activation function
  - Linear, sigmoid, relu...
- Dropout rate (or not)
- Threshold

HYPERPARAMETER TUNING

- Type of optimizer
- Learning rate (fixed or not)
- Regularization rate (or not)
- Regularization type: L1, L2, ElasticNet
- Type of search for local minima:
  - Gradient descent, simulated
  - annealing, evolutionary...
- Batch size
- Nesterov momentum (or not)
- Decay rate (or not)
- Momentum (fixed or not)
- Type of fitness measurement:
  - MSE, accuracy, MAE, cross-entropy,
  - precision, recall
- Epochs
- Stop criteria

Rubens Zimbres

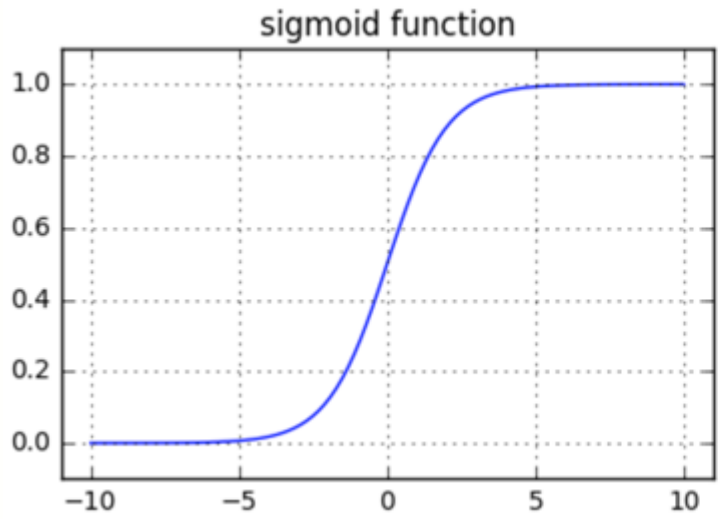
GANs

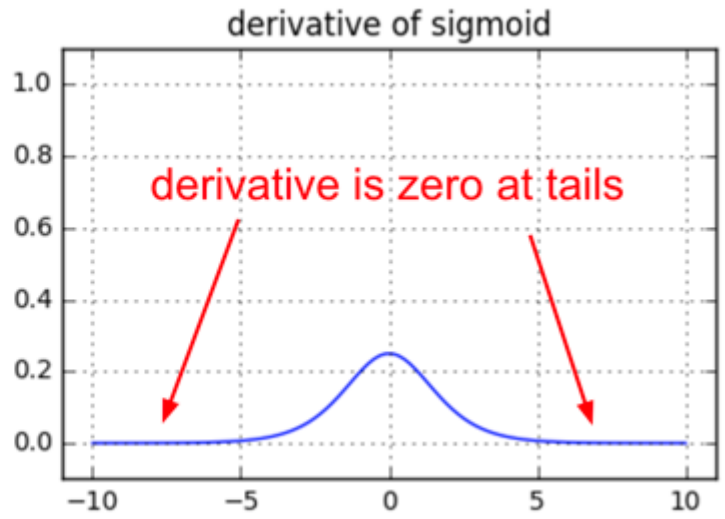
- Train <u>Simultaneously</u> (no freeze)
- Train Discriminator one step
- Train Generator k steps (no Nash)
- Until $p_G = p_{data}$ (global optimu $D(x) = P(x \epsilon data) = 1/2$
- Discriminator cannot distinguish both distributions

GANs

- x comes from p_{data} y=1 (Real)
- x comes from p_G y=0 (Fake)
- Optimize θ rather than p_G
- G: Relu (vanishing gradient)
- Dropout
- Supervised Learning in D:
- Beginning: high confidence
- Usually D is bigger/deeper than G

Vanishing Gradient

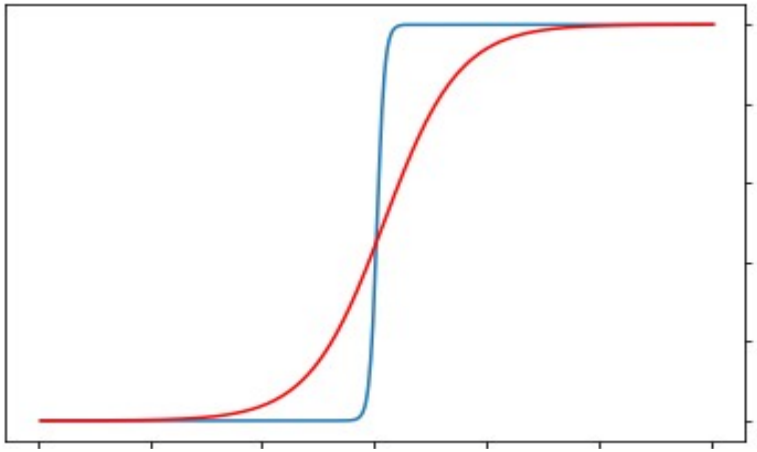


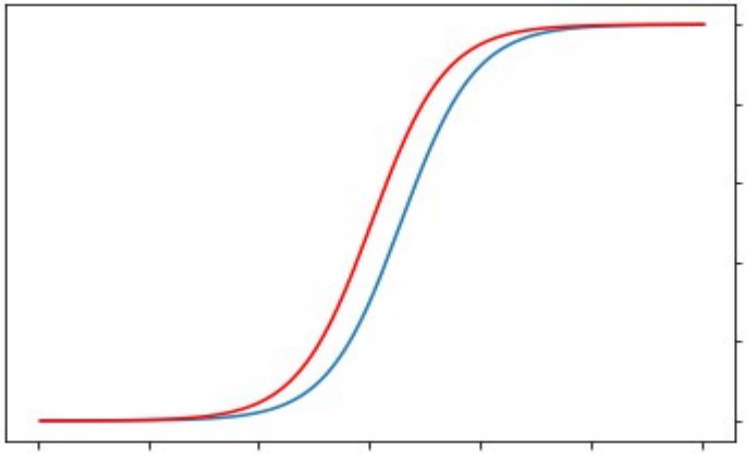


-25% each activation

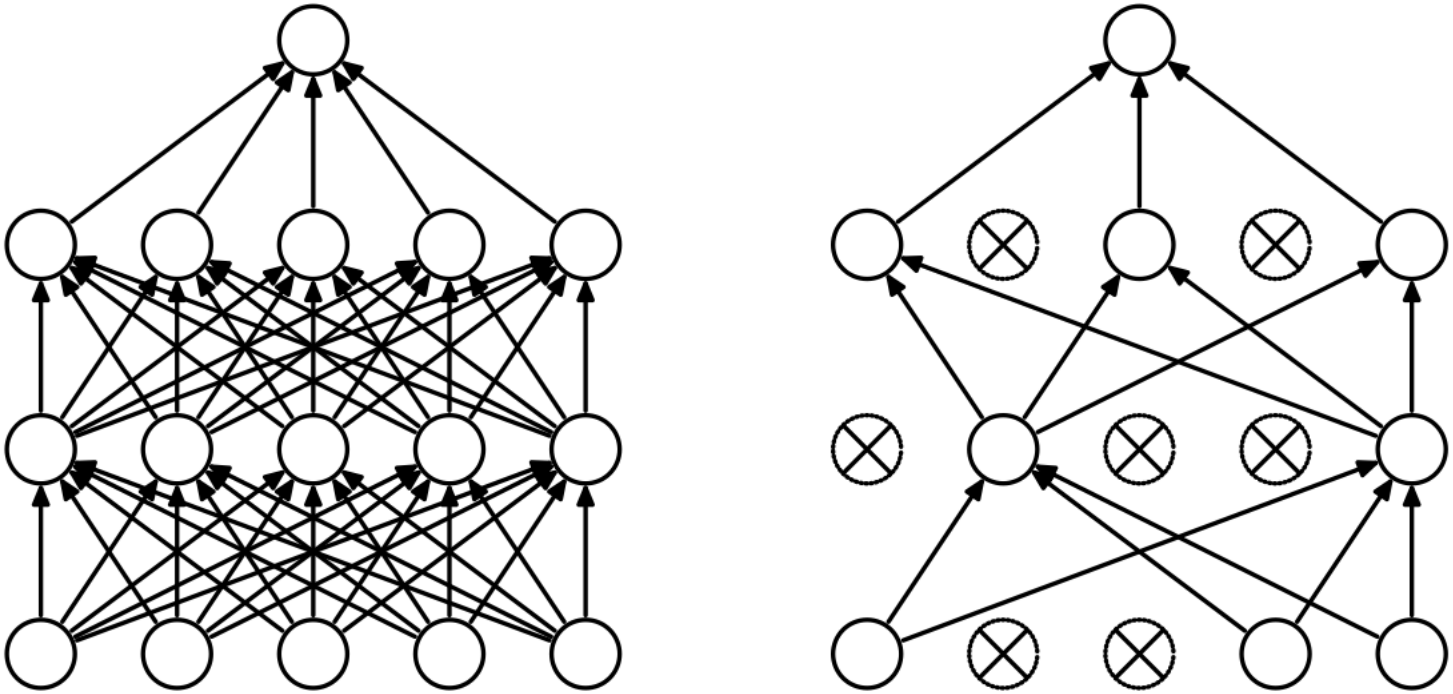
Image source: Andrej Karpathy

Coefficient and Intercept: Regularization





Dropout



(a) Standard Neural Net

(b) After applying dropout.

Training GANs

$$V(G, D) = \mathbb{E}_{p_{\text{data}}} \log D(\boldsymbol{x}) + \mathbb{E}_{p_{\text{generator}}} \left(\log \left(1 - D(\boldsymbol{x})\right)\right)$$

- Generator: Gradient Descent on V
- Discriminator: Gradient Ascent on V
- Minibatch
- Batch Normalization in G ***
- To minimize Cross Entropy
- Minimax Game
- Regular cross entropy

Training GANs

- Discriminator minimize $J^{(D)}(\theta^D, \theta^G)$ controlling θ^D
- Generator minimize $J^{(G)}(\theta^D, \theta^G)$ controlling θ^G
- Cannot control each other

$$D \text{ tries } D\big(G(z)\big) \sim 0$$

$$G \text{ tries } D\big(G(z)\big) \sim 1$$

$$J_D = -\frac{1}{2}\mathbb{E}_{x \sim p_{data}} logD(x) - \frac{1}{2}\mathbb{E}_z log(1 - D(G(z))$$

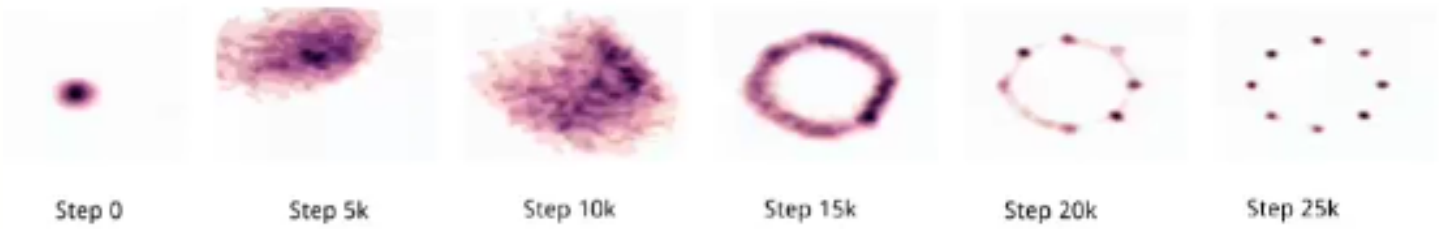
$$J_G = -J_D \qquad J_G = -\frac{1}{2}\mathbb{E}_z logD(G(z)) \qquad \text{(G still learns)}$$

Training Challenges

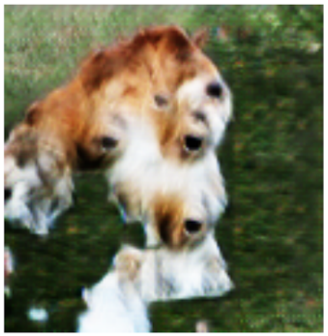
- Non convergence: Nash Equilibrium, harder to optimize than objective function

- Mode collapse: G fails to output diversity (few good samples)
  - D converges to right distribution
  - G generates samples in the most probable point

-



Training Challenges






Tips and Tricks for GANs

- Normalize data (-1,1)
- Activation function Tanh output of Generative
- Sample from gaussian distribution instead of uniform distribution
- Develop diferente mini-batches in Discriminator:
  - All real
  - All fake
- Use Batch Normalization

Tips and Tricks for GANs

- Use DCGANs or VAE+GAN
- Adam in Generator (exp. momentum decay)

(Adaptive Moment Estimation)

- SGD in Discriminator

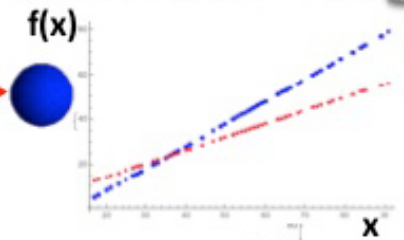
$$J(\theta_j) := \theta_j \pm \eta . (\hat{y} - y)^2 . x$$

- If Generator loss high = garbage to Discriminator
- Dropouts in Generator (overfitting)

Gradient Descent for Linear Regression

$$f(x)_{t0} = \sum_{i=1}^{n} m_{i(t0)} x_i + b$$

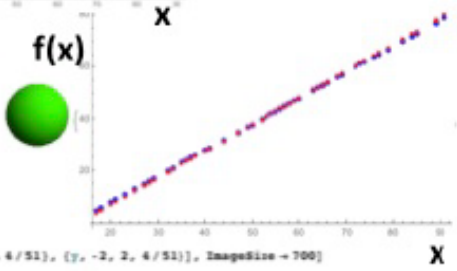
Linear Regression

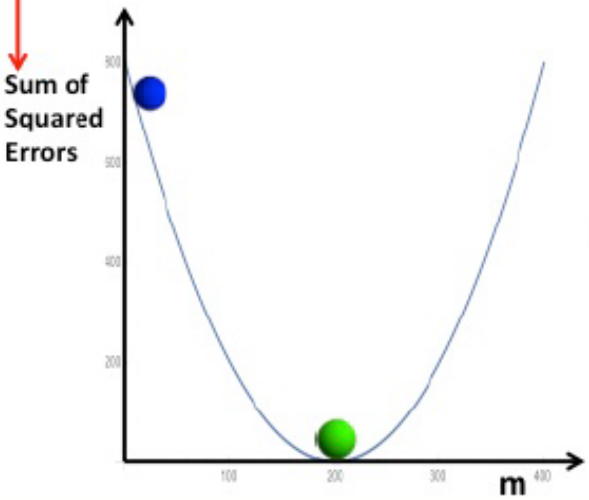


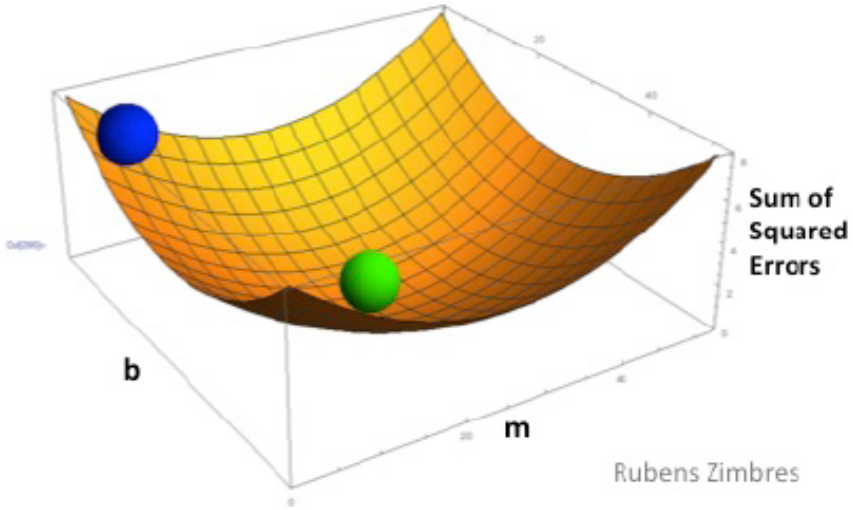
$$m_t := m_{t-1} \pm \eta . \frac{\partial}{\partial x} \frac{\sum_{i=1}^{n} (m_{i(t0)} x_i - m_{i(t-1)} x_i)^2}{2n}$$

$$m_t := m_{t-1} \pm \eta . \frac{\sum_{i=1}^{n} (m_{i(t0)} x_i - m_{i(t-1)} x_i)}{n}$$

$\eta = Learning\ Rate \approx 0.01$

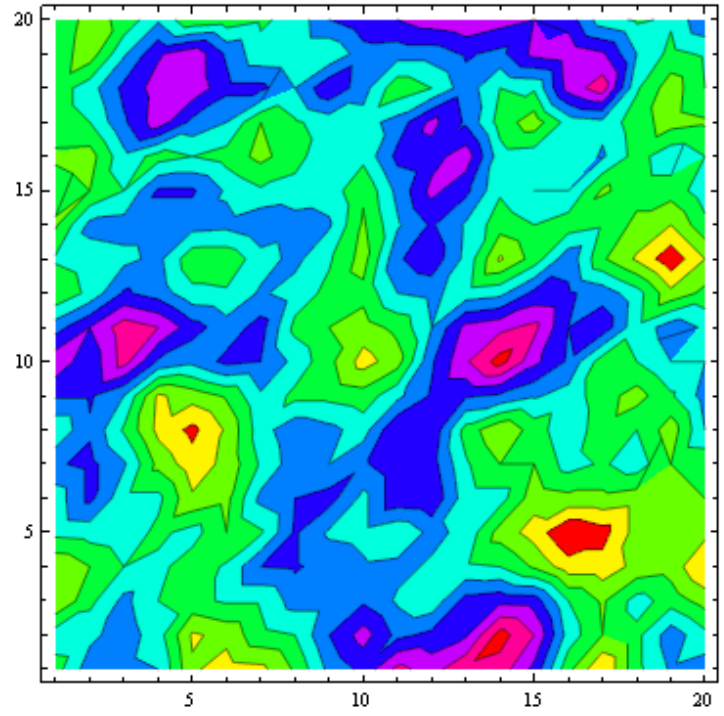




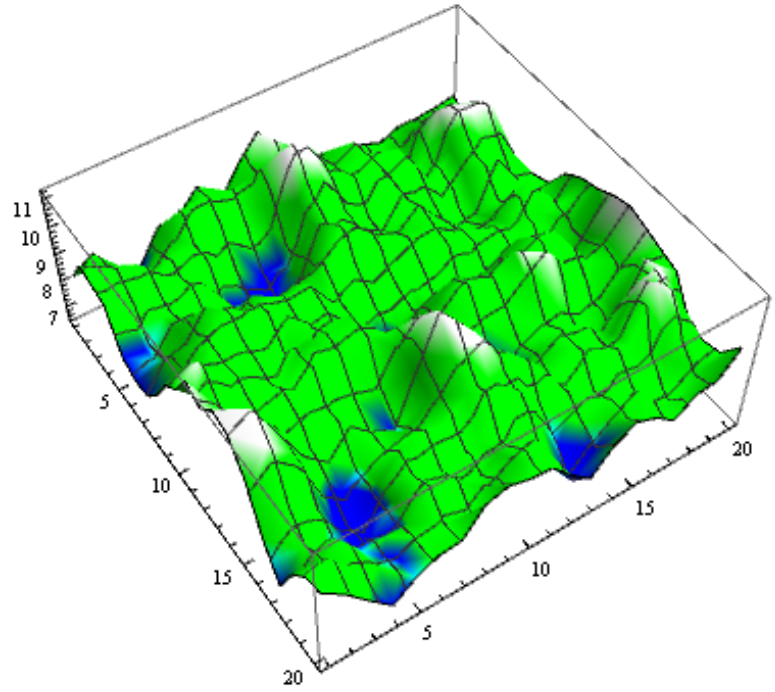


Rubens Zimbres

display Contour Plot 3D Plot Raw Data
smoothness Rough Smoothed Smoothest
rule
initial
steps



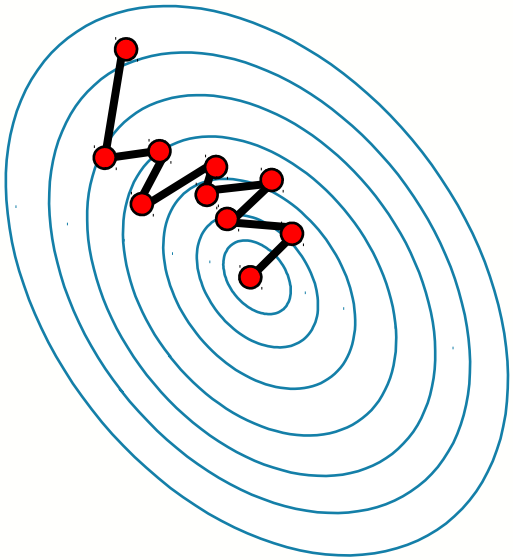
display Contour Plot 3D Plot Raw Data
smoothness Rough Smoothed Smoothest
rule
initial
steps

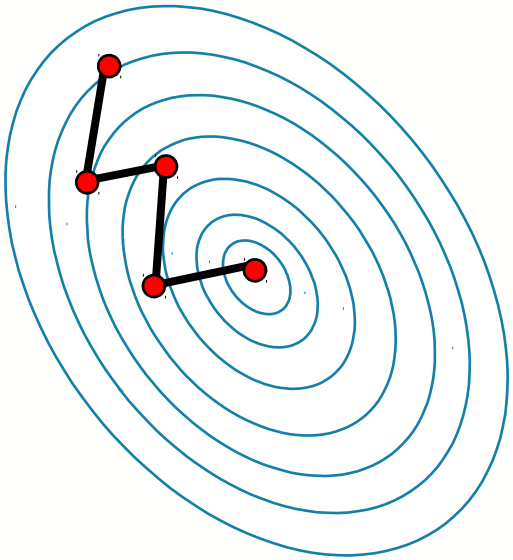


Gradient Descent

Nesterov Momentum

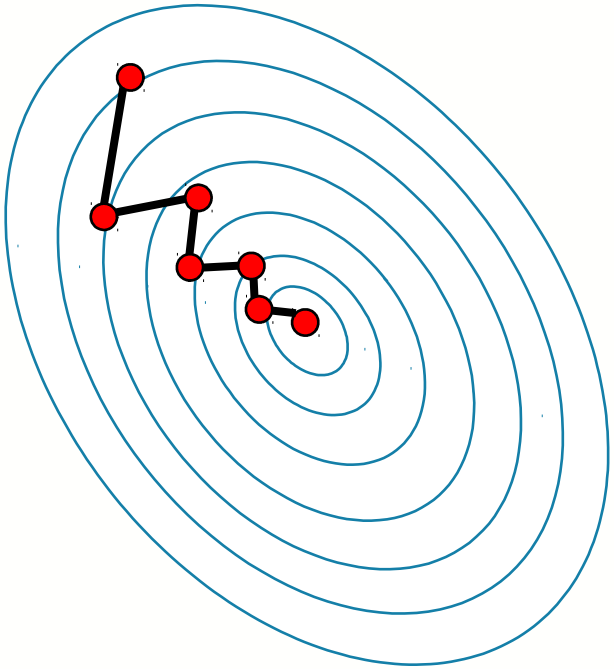
$$\theta = \theta - (\gamma v_{t-1} + \eta . \nabla J(\theta - \gamma v_{t-1}))$$

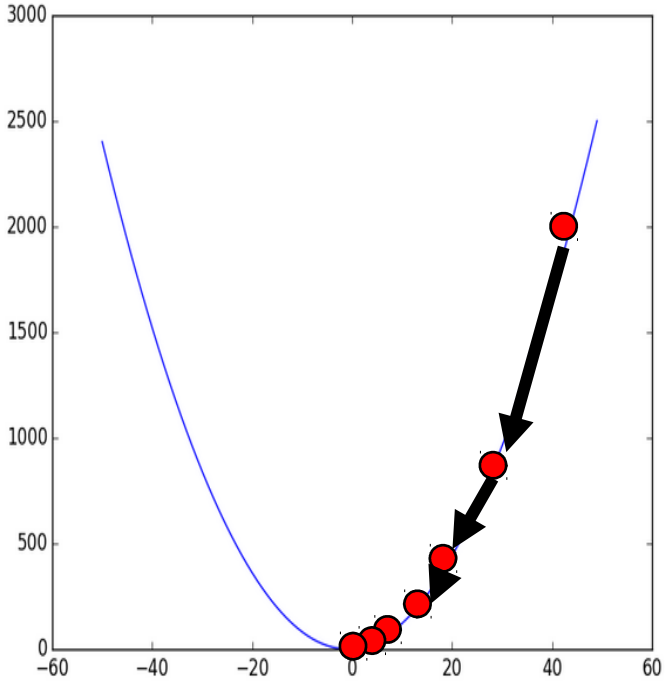




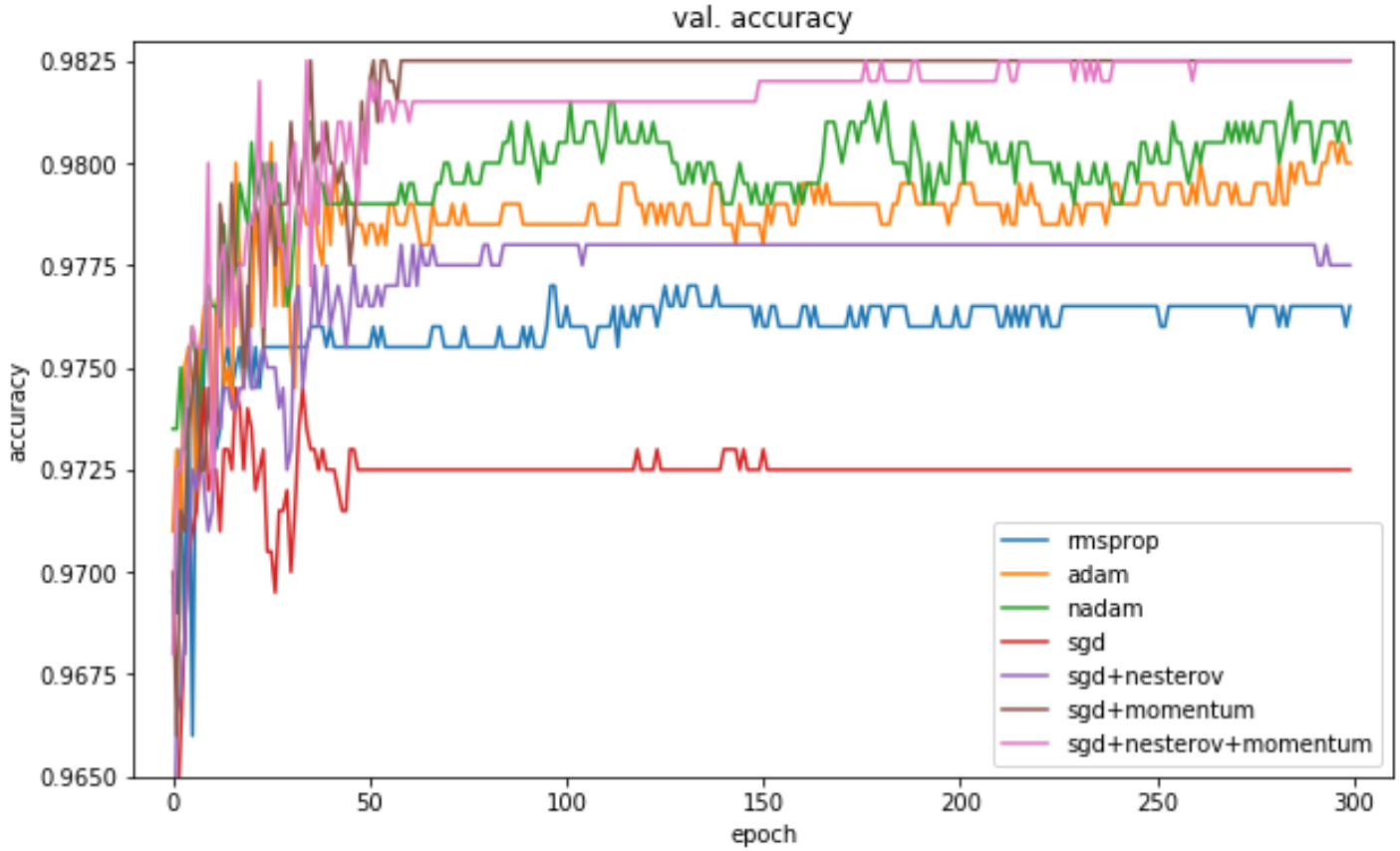
Adam Optimizer

Momentum w/ exponential decay according to derivative of error



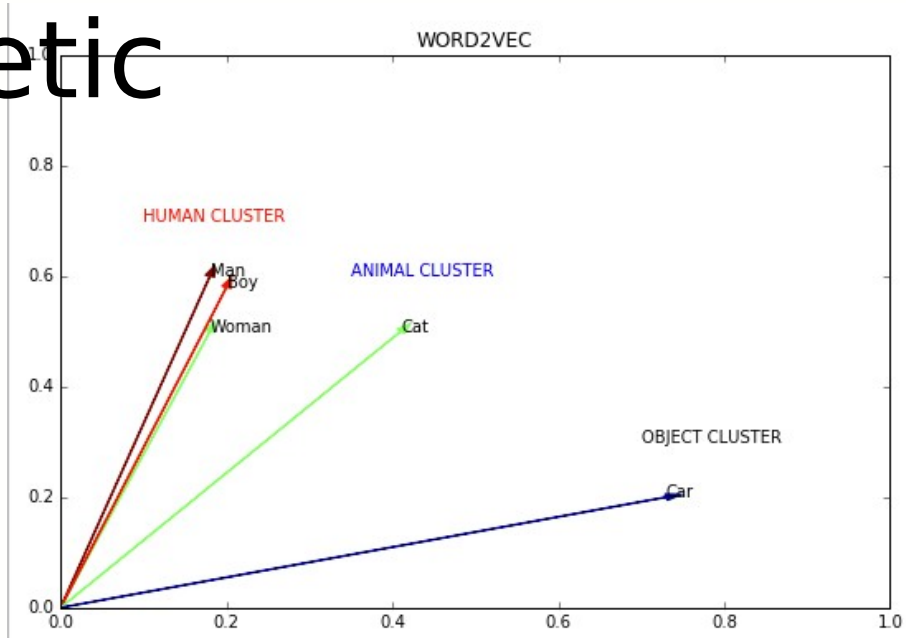


Optimizers: generalization



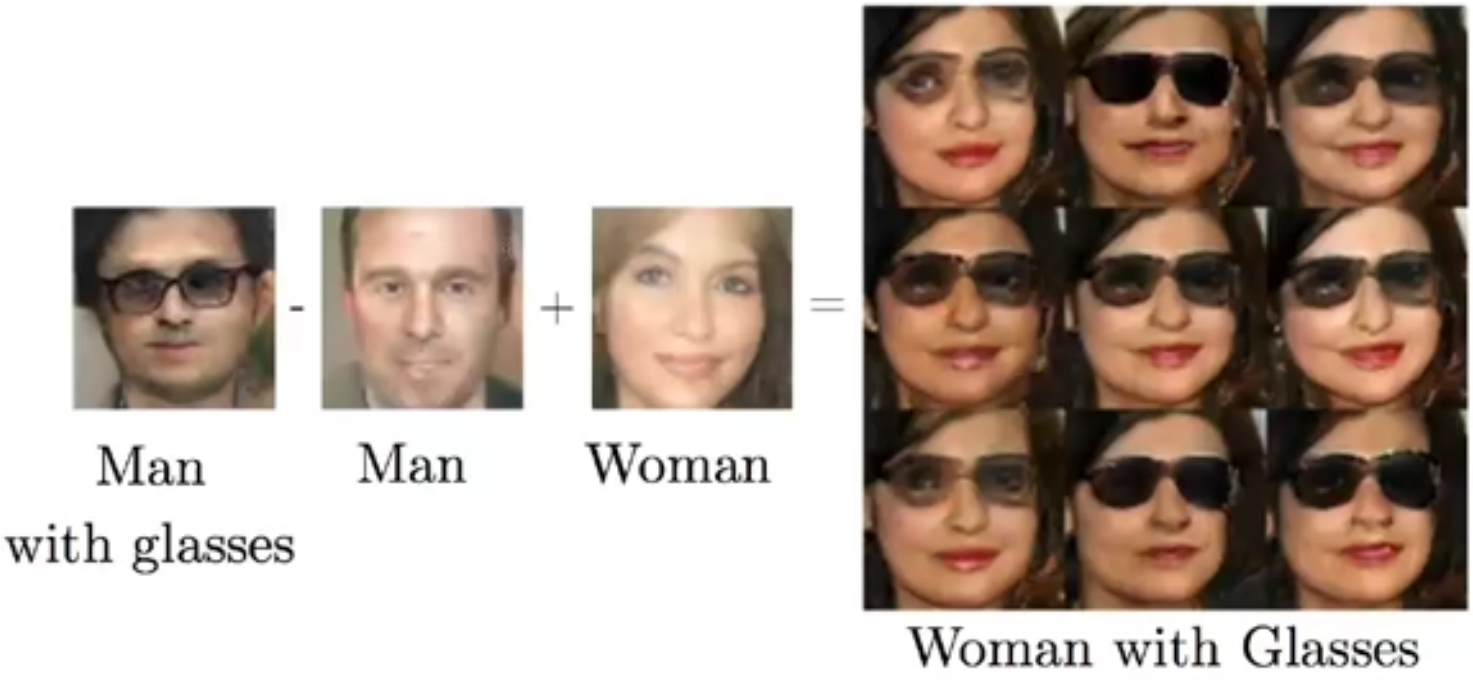
Wilson et al, 2017

GAN Examples: Vector Arithmetic



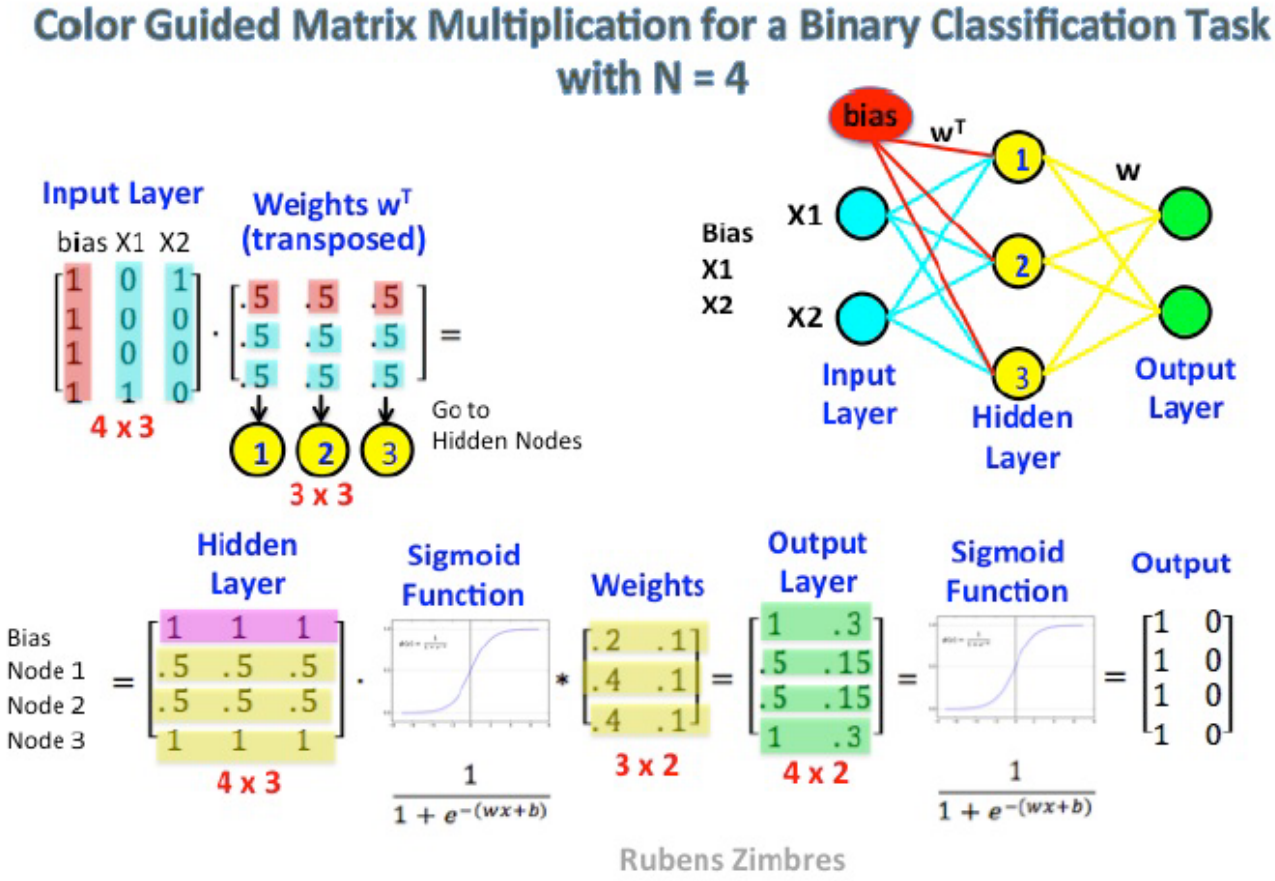

```
KNN Similarity with Man
Woman : 0.9
Man : 1.0
Boy : 0.971715728753
Cat : 0.74920127592
Car : 0.319926474563

Cosine Similarity with Man: Woman 0.882122164357
Cosine Similarity with Man: Man 1.0
Cosine Similarity with Man: Boy 0.971120831449
Cosine Similarity with Man: Cat 0.757031631033
Cosine Similarity with Man: Car 0.437165180814

In [3]:
```

Vector Arithmetic



Radford et al, 2015

Matrix Multiplication



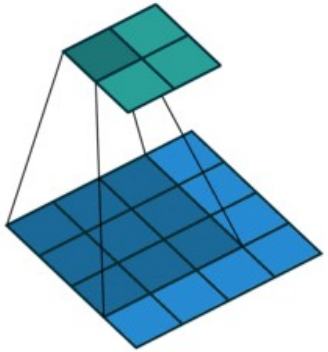
DCGAN

- Deep Convolutional GAN
- Batch Norm not in end of Generator and NOT in Discriminator (x covariate shift)
- To increase dimension, Upsampling: convo2D.T with stride > 1
- Downsampling: average pooling + conv2D + stride

Convolutions

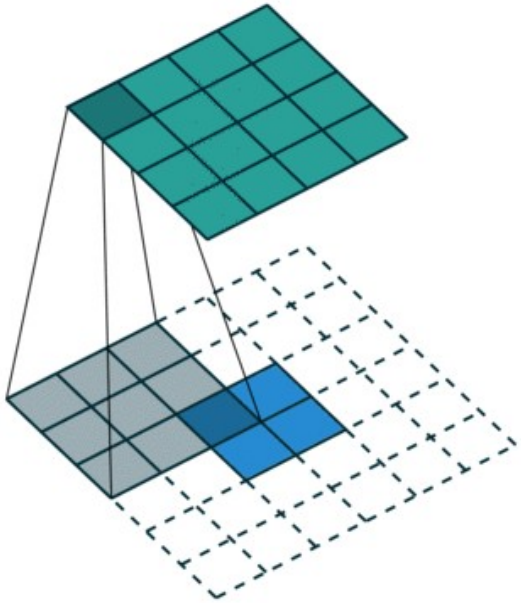
Convolution2D

3x3 kernel 4x4 input stride=1

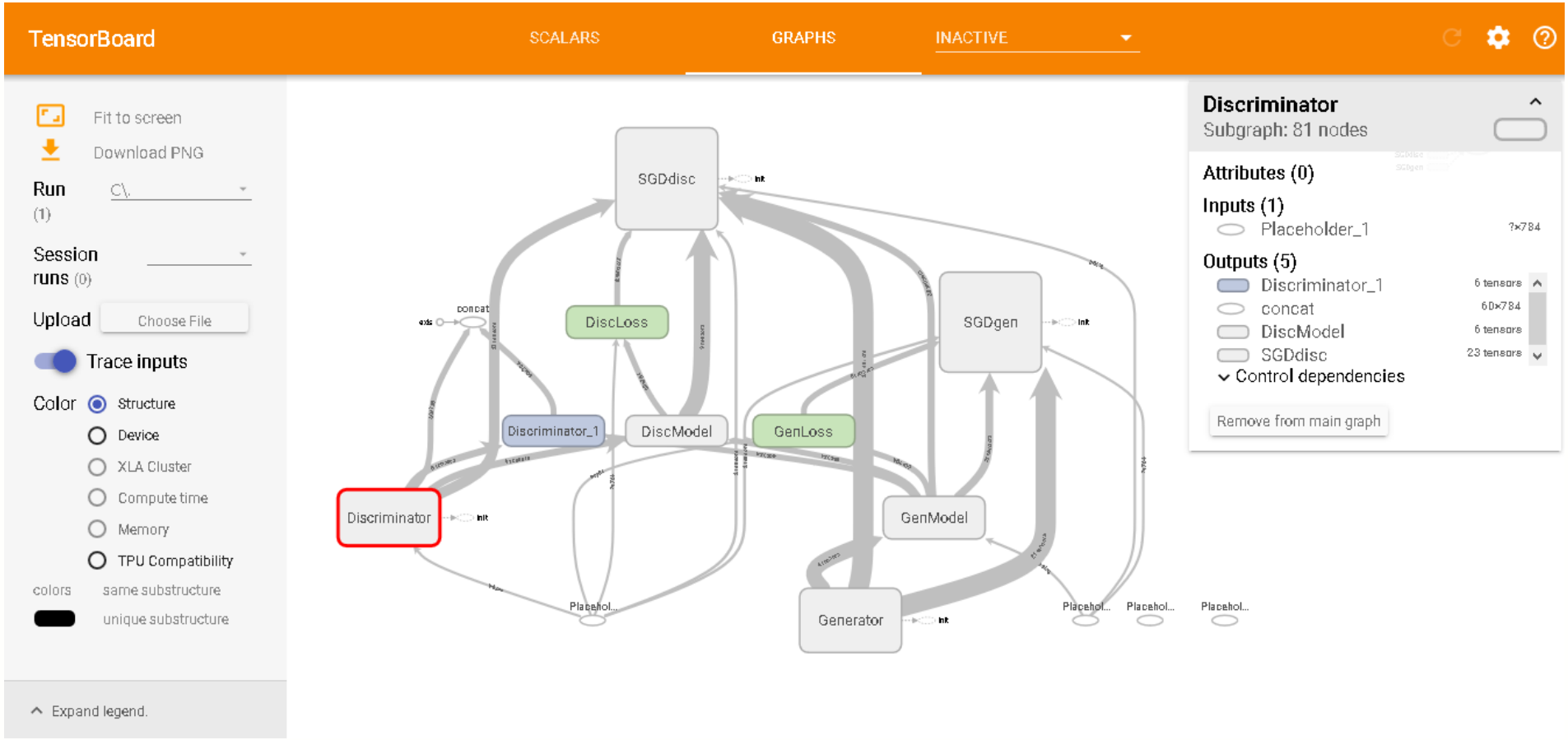


Convo2DTranspose

3x3 kernel 2x2 input stride=1



DCGAN Architecture



DCGAN Training

TensorBoard SCALARS GRAPHS INACTIVE

Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default

Smoothing 0,6

Horizontal Axis
STEP RELATIVE WALL

Runs
Write a regex to filter runs
c\.

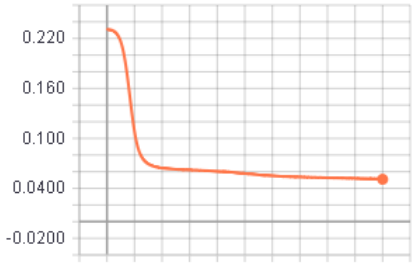
TOGGLE ALL RUNS

C:/Users/razim/Anaconda3/envs/tensorflow/Scripts/

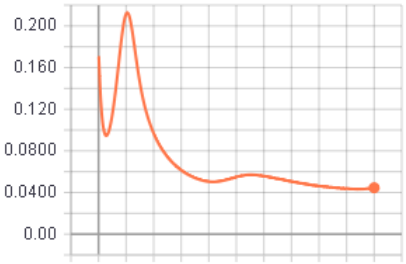
.*

Tags matching /.*/ (all tags) 2

Discriminator_Loss



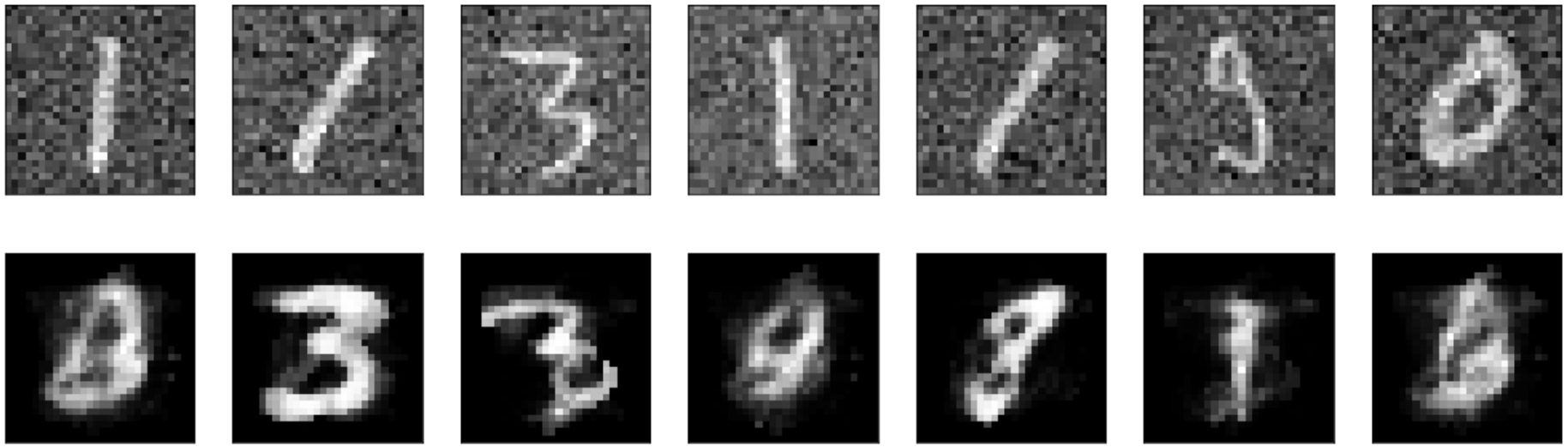
Generator_Loss



Discriminator_Loss 1
Generator_Loss 1

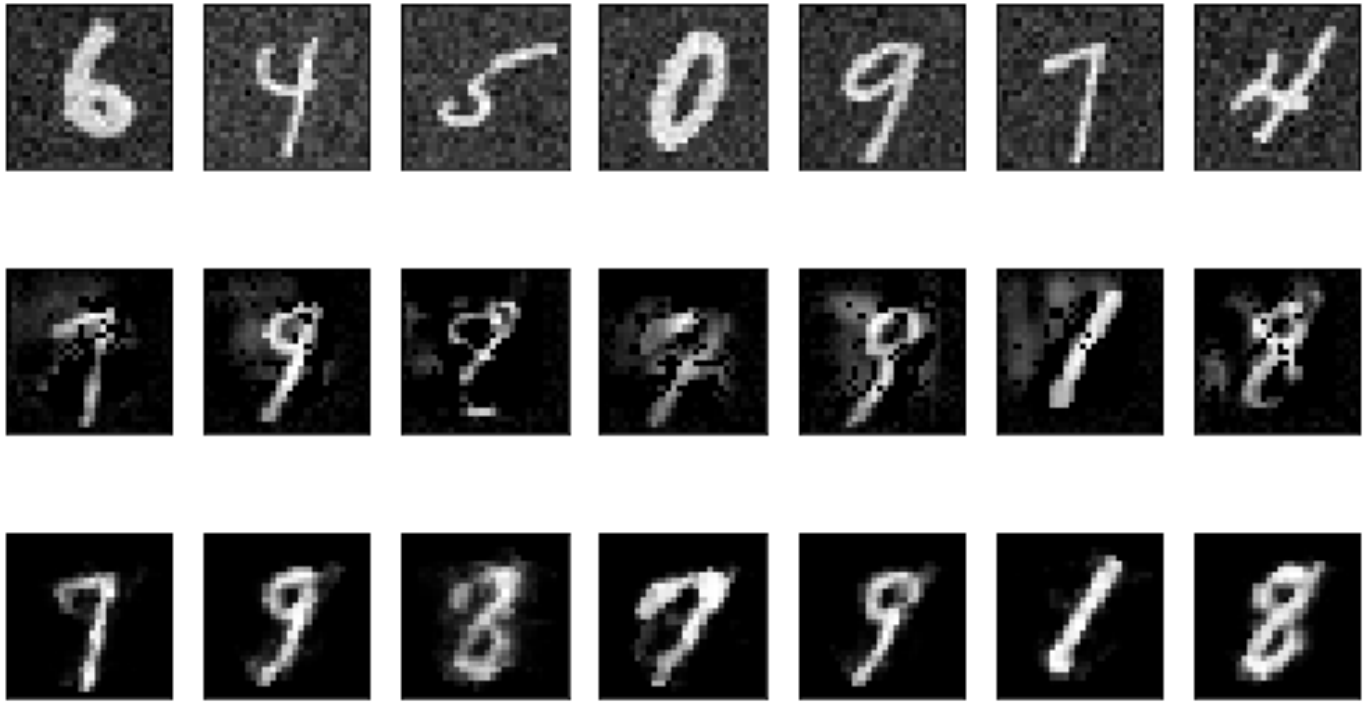
DCGAN Output

```
Epoch 1750: Generator Loss: 0.042980, Discriminator Loss: 0.030590
Epoch 1800: Generator Loss: 0.042635, Discriminator Loss: 0.029989
Epoch 1850: Generator Loss: 0.039975, Discriminator Loss: 0.029853
Epoch 1900: Generator Loss: 0.038871, Discriminator Loss: 0.029369
Epoch 1950: Generator Loss: 0.037111, Discriminator Loss: 0.029259
Epoch 2000: Generator Loss: 0.035706, Discriminator Loss: 0.029093
```

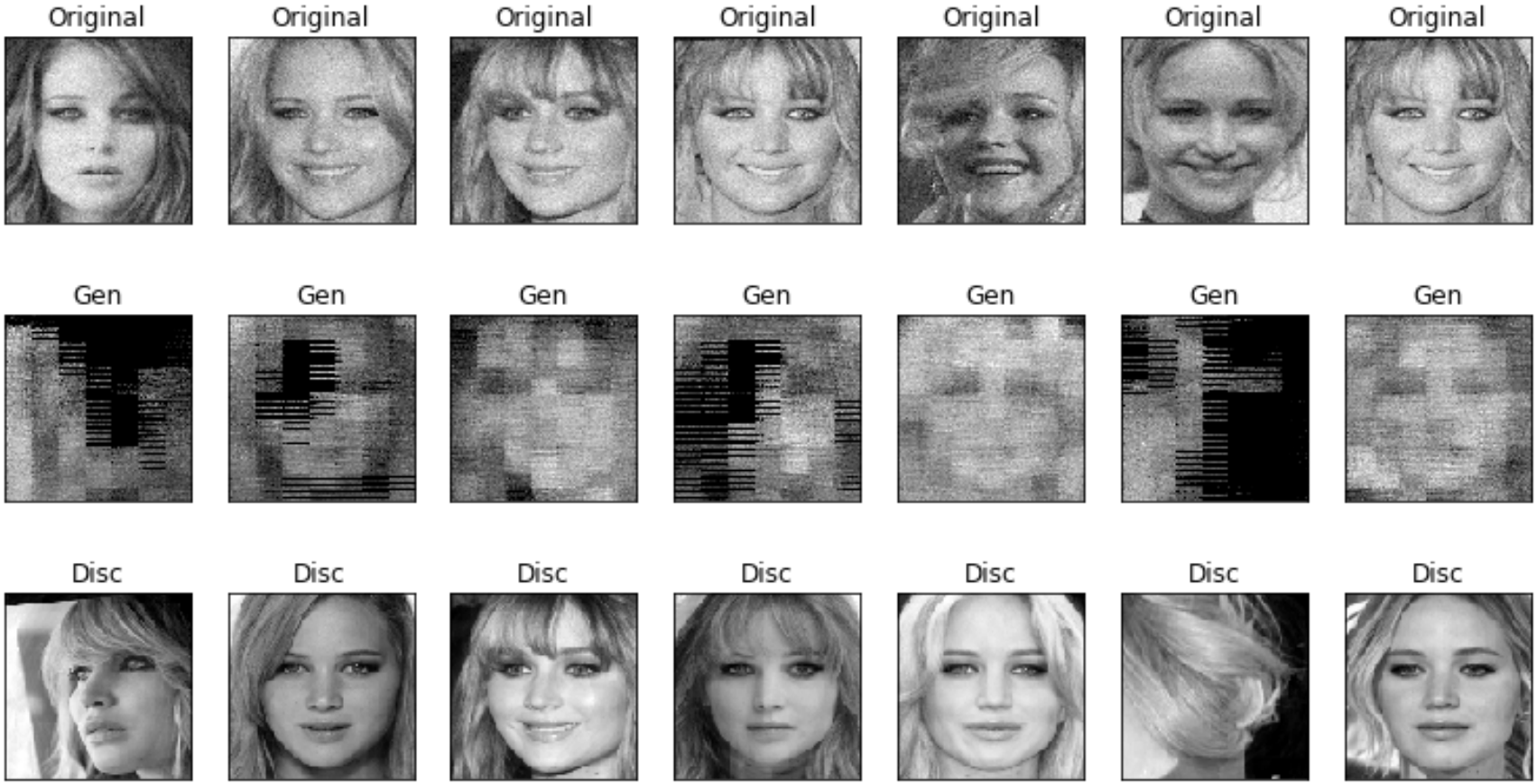


DCGAN Output

```
Epoch 2850: Generator Loss: 0.051090, Discriminator Loss: 0.014070
Epoch 2900: Generator Loss: 0.050746, Discriminator Loss: 0.013686
Epoch 2950: Generator Loss: 0.050411, Discriminator Loss: 0.013317
Epoch 3000: Generator Loss: 0.050048, Discriminator Loss: 0.012959
```



DCGAN – Face Recognition



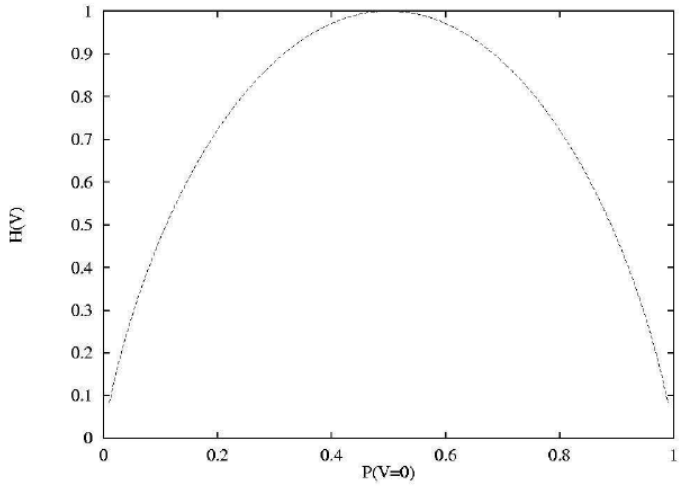
Info GAN

- Based in Mutual Information and Entropy (uncertainty)

$$H(V) = \sum_{v=0}^{1} -P(H = v) \lg P(H = v).$$

$$I(A; B) = H(B) - \sum_{b} P(B = b) \cdot H(A|B = b)$$

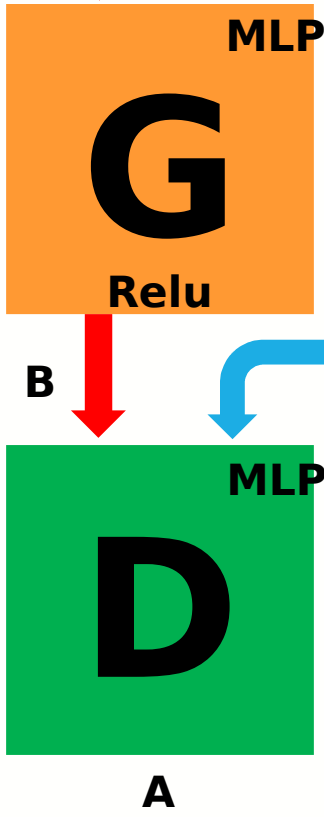
Coin toss



Latent Variables (add noise) Fake

b

b=[1,1,1,1,1,1,1,1,1,1] N=10



H(B)=-SumP(B=b).logP(B=b)

B=[0,0,0,0,0,1,1,1,1,1]
P(B=b) = 0.5

H(B)=-0.5.log(0.5)
H(B)=-0.15

Training Data Real

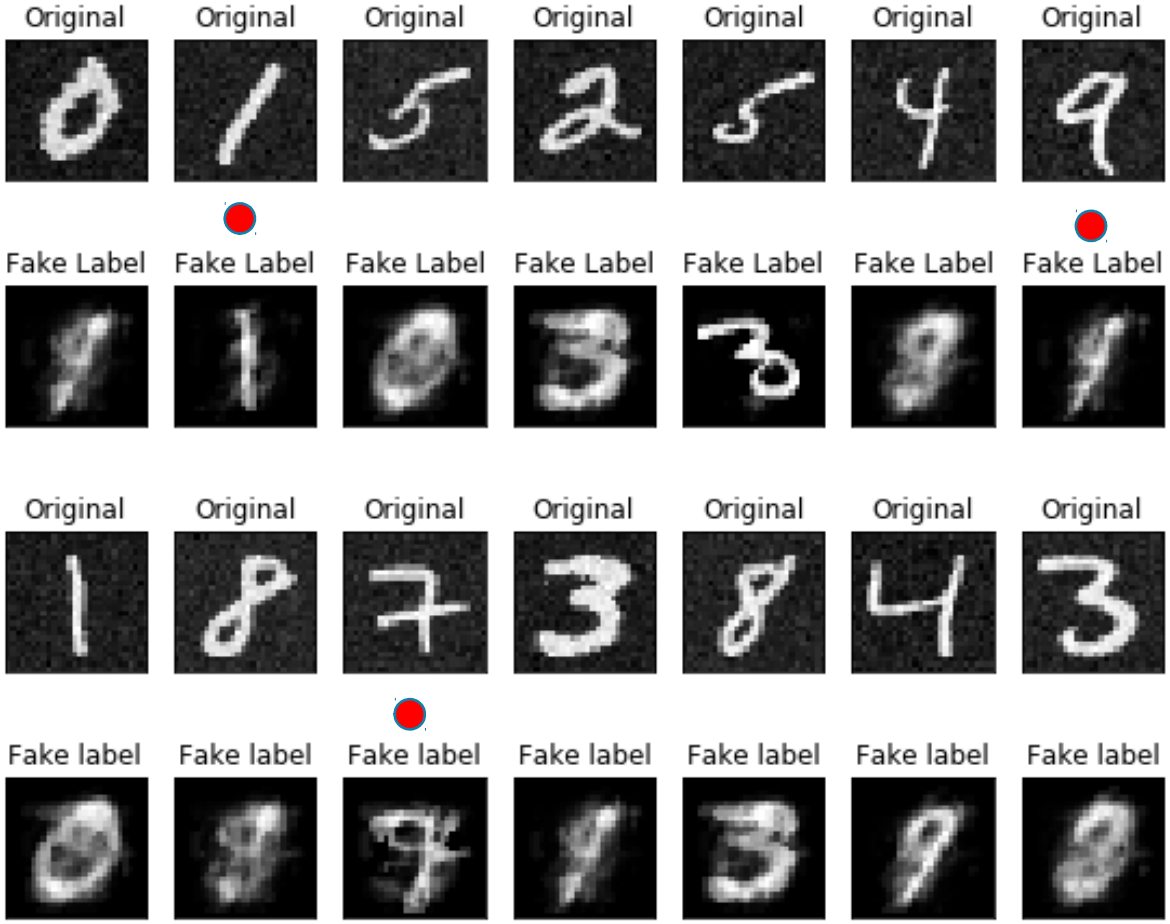
B=[0,0,0,0,0,1,1,1,1,1]

A=[0,0,0,0,0,0,1,1,1,1]
P(A|B=b) = 4/5= 0.8 (subsample)

H(A|B=b)=-0.8.log(0.8)
H(A|B=b)=-0.04

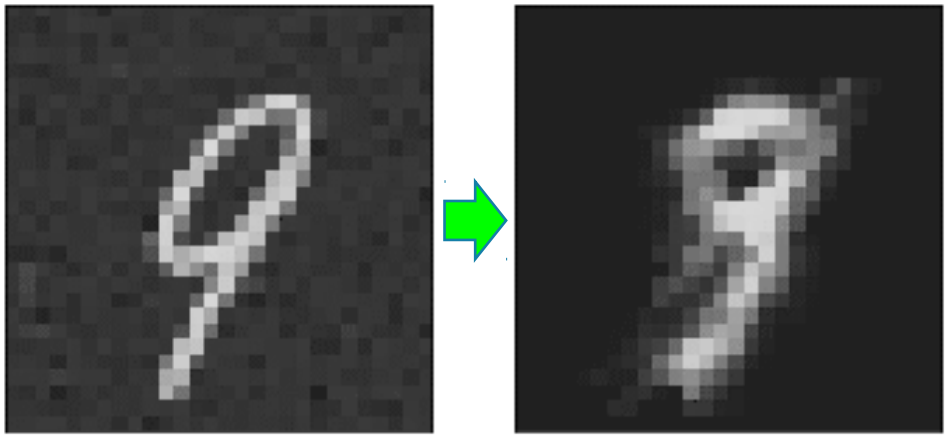
$$I(A;B) = H(B) - \sum_b P(B=b) \cdot H(A|B=b)$$

Info GAN Output



Info GAN Output

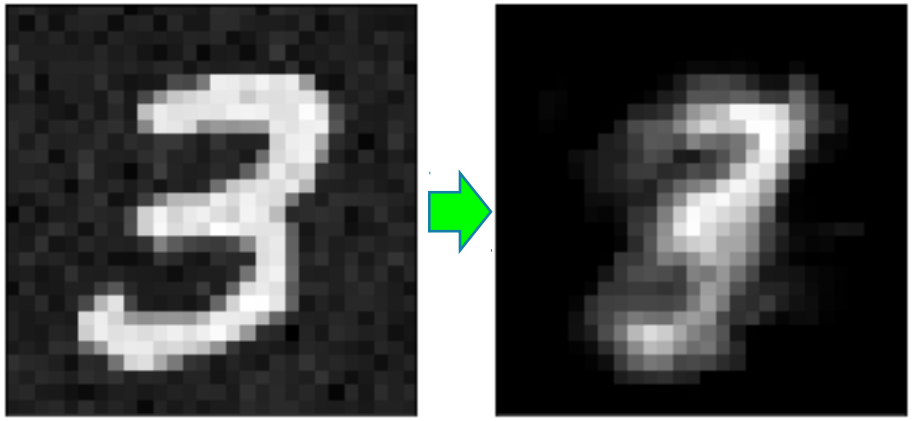
```
In [135]: plt.figure(figsize=(6, 6))
     ...: ax = plt.subplot(1, 2, 1)
     ...: plt.imshow(x_train_noisy[0].reshape(28, 28))
     ...: ax.get_xaxis().set_visible(False)
     ...: ax.get_yaxis().set_visible(False)
     ...: ax = plt.subplot(1, 2, 2)
     ...: plt.imshow(np.array(g).reshape(60,28, 28)[0])
     ...: ax.get_xaxis().set_visible(False)
     ...: ax.get_yaxis().set_visible(False)
     ...: plt.show()
```



TensorFlow

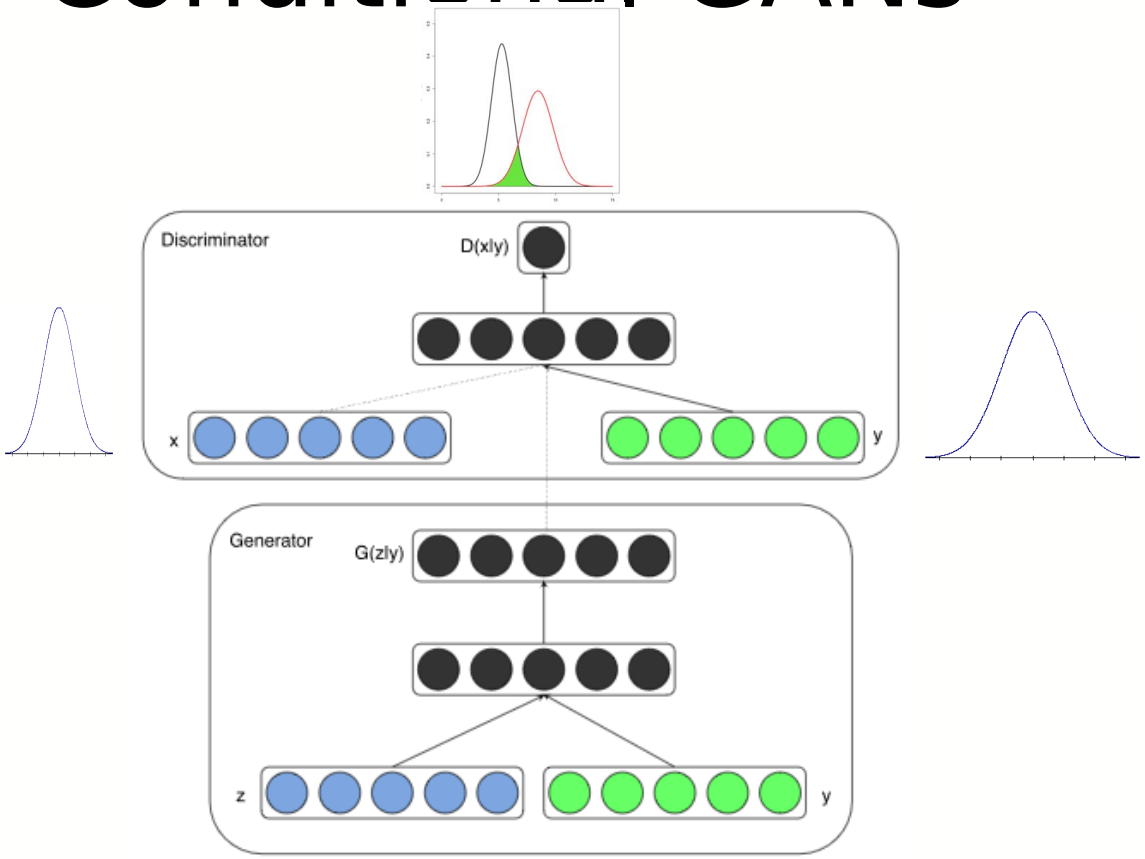
Info GAN Output

```
In [186]: h=9
     ...: plt.figure(figsize=(6, 6))
     ...: ax = plt.subplot(1, 2, 1)
     ...: plt.imshow(x_train_noisy[h].reshape(28, 28))
     ...: ax.get_xaxis().set_visible(False)
     ...: ax.get_yaxis().set_visible(False)
     ...: ax = plt.subplot(1, 2, 2)
     ...: plt.imshow(np.array(g).reshape(60,28, 28)[h])
     ...: ax.get_xaxis().set_visible(False)
     ...: ax.get_yaxis().set_visible(False)
     ...: plt.show()
```



TensorFlow

cGAN: Conditional GANs

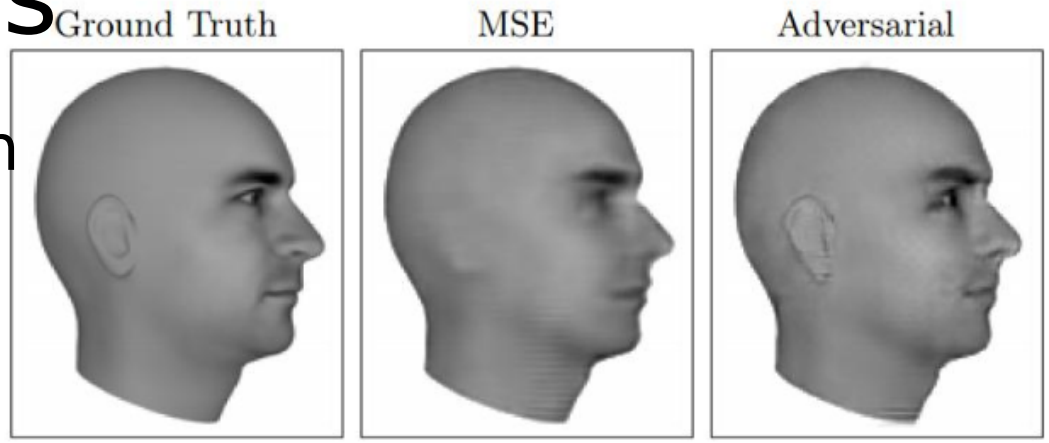


$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]$$

GAN Examples

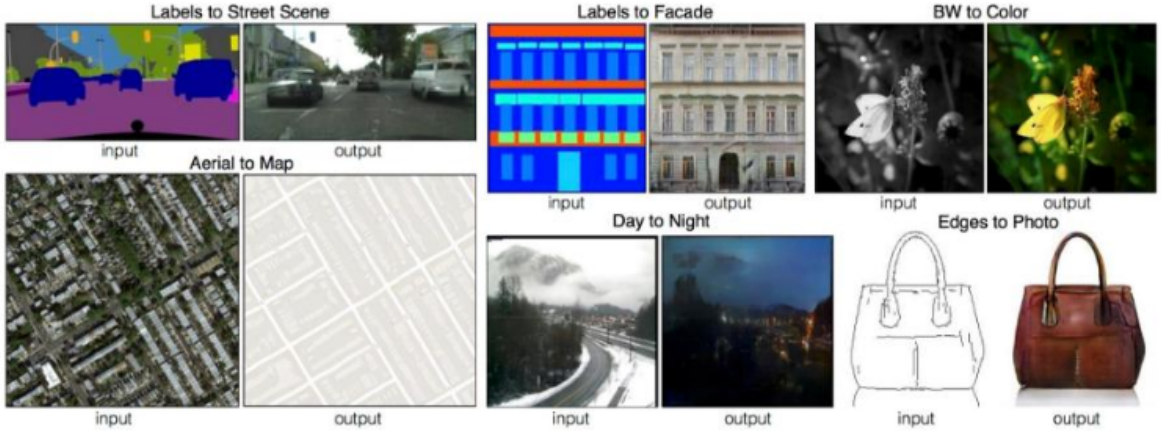
- Next frame prediction

  Lotter et al, 2016



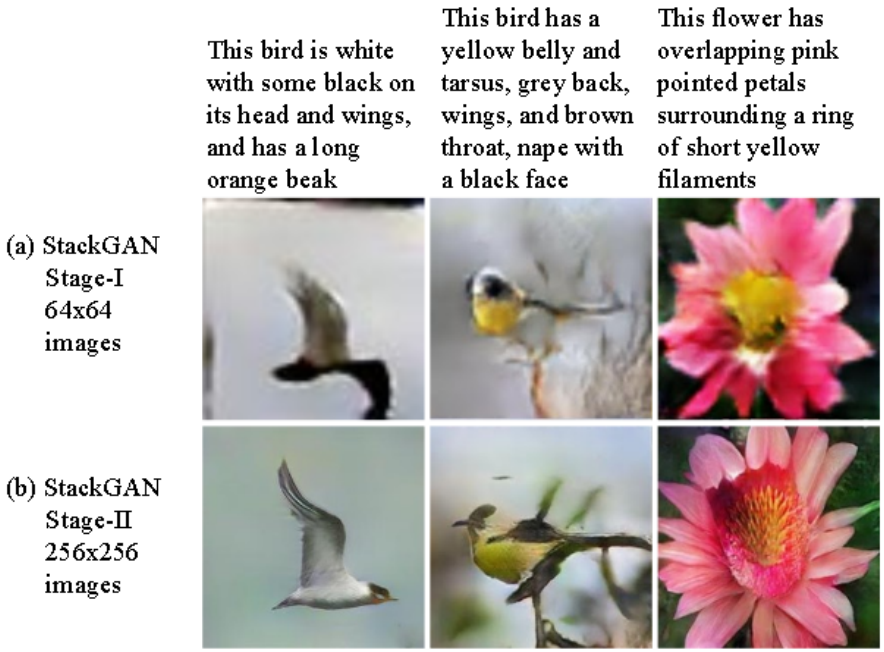
- Image-to-image translation

  Isola et al, 2016



GAN Examples

- Text-to-Image generation
- Image Inpainting

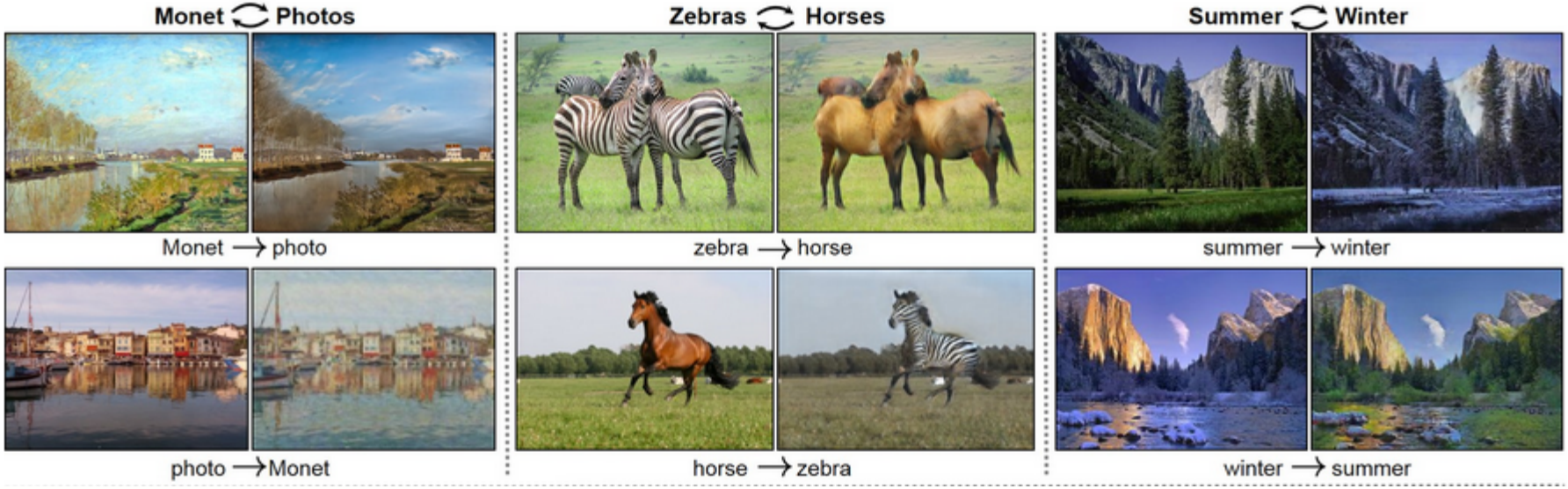


Reed et al, 2016

Pathak et al, 2016

Cycle GAN: Style Transfer

junyanz / CycleGAN



https://github.com/RubensZimbres

Workshop

Try different strategies in GANs