Generative Model vs **Discriminant** models decision boundary

Generative Model
- ↳ $P(x,y)$ model actual distr
  - ↳ can compute marginal $P(x)$, $P(y)$
- ↳ can generate samples (x_i, y_i)
- ex) Naive Bayes: asymptotic error
  - ↳ assumes conditional indep
  - $P(x_i, x_j | label) = P(x_i | label)$
- ex) generate image from $P(x = white | Y = man)$

Discriminant
- ↳ $P(Y=y | X=x)$
- ↳ less assumption ⇒ can fit complex model
- ↳ learn slow & need big data, but high accuracy
- ex) LS, LR, SVM, decision Tree + Random Forests, Neural Net

Loss Func L: L2 loss if $y \in$ Real val, cross entropy loss if y is discrete label

Linear Regression: $\hat{y} = ax + b \in \mathbb{R}$

True Risk $= \mathbb{E}[(\hat{y}-y)^2]$ = avged loss across all data pts

Squared loss $L = \frac{1}{n}\sum_i^n (\hat{y}_i - y_i)^2$ = **Empirical Risk** = avged loss across finite n data pts
- ↳ fails if (1) data sample is biased
- (2) not enough data to estimate true distr

multivariate lin Reg $[\vec{y}] = [A][x] + [b] = [A \; 1]\begin{bmatrix} x \\ b \end{bmatrix}$

$L = \sum |A\vec{x}_i - \vec{y}_i^T|^2$

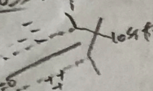
$\nabla_A L = 0 = 2A[\sum x_i x_i^T] - [\sum y_i x_i^T]$

$A = M_{yx} M_{xx}^{-1}$

"sample moments" converge to True moments

SVM: $f(x) = \vec{w}^T\vec{x} - b$, $y_i = \{0, 1\}$ dist of // margin

Hinge loss $= \sum_i^n \max\{0, 1 - y_i f(x_i)\}$



soft margin SVM: $L = \sum \max\{0, 1 - y_i f(x_i)\} + \lambda \|w\|^2$

Logistic Regression can learn Naive Bayes classif

$x \in \mathbb{R}^m$, $y = \begin{cases} 1 & \text{if cat} \\ 0 & \text{ow} \end{cases}$ (Binary label)

$f(x) = \frac{1}{1 + e^{-\vec{w}^T\vec{x}}} = \mathbb{P}(Y = cat | \vec{X} = \vec{x})$

cross entropy loss $= -\log \mathbb{P}(\text{correct})$
$= -y \log \hat{y} + (1-y)\log(1-\hat{y})$
$= -\sum y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)$

multiclass Logistic Reg

$f_i(x) = \mathbb{P}(x \in class\ i)$, $y_i = \begin{cases} 1 & \text{if } i \in class\ i \\ 0 & \text{ow} \end{cases}$

let $S = \underset{k \times m}{W}\vec{x}$, $f_i(x) = \frac{e^{S_i}}{\sum_j^k e^{S_j}}$

$CEL = \sum_i^m y_i^T \log f(x_i)$

Optimization

Gradient Decent (GD): $w^{t+1} = w^t - \alpha \nabla_w L(w)$

ex) $\frac{\partial L}{\partial w} = \sum^N \frac{\partial}{\partial w} L(x_i, y_i, w)$
- ↑ @ each time t, need $O(n)$

SGD: use minibatch of m random samples

$g^t = \frac{1}{m}\sum_{i \in 1..m} \nabla_w L(x_i, y_i, w)$ ← introduce high var

$w^{t+1} = w^t - \alpha g^t$

Newton's Method: jump to local optima & saddle

$x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)}$ converge fast!!

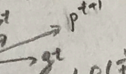
$= x_t - H_f(x_t)^{-1}\nabla f(x_t)$

↳ in practice
- (1) takes $O(n^3)$ ⇒ soln: L-BFGS
- (2) gets stuck in saddle pt
- (3) unstable b/c we need inverse

SGD w/ momentum converge w/ $O(\frac{1}{t})$

$p^{t+1} = \mu p^t - \alpha g^t$ s.t. μ is momentum constant

$w^{t+1} = w^t + p^{t+1}$



Nesterov Moment converge w/ $O(\frac{1}{t^2})$
- ↳ Take moment step before gradient

$p^{t+1} = \mu p^t - \alpha \nabla_w L(w^t + \mu p^t)$

$w^{t+1} = w^t + p^{t+1}$

RMSprop (No momentum)
- ↳ scale grad by inverse of moving avg s^t

$s^t = \beta s^{t-1} + (1-\beta)(g^t)^2$ = moving avg

$w^{t+1} = w^t - \alpha \frac{g^t}{\sqrt{s^t}}$

$\beta \in [0,1]$ decay factor

ADAGRAD (No momentum)
- ↳ cummulative sum of squared grad

$c^t = \sum_i^t (g^i)^2 \propto O(t)$

$w^{t+1} = w^t - \alpha \frac{g^t}{\sqrt{c^t}}$

learning rate decreases w/ $O(\frac{1}{\sqrt{t}})$
- ↳ good for convex, bad for non convex

ADAM = RMSprop + momentum + Bias correction
- ↳ use mov avg of grad & squared grad

$p^t = \beta_1 p^{t-1} + (1-\beta_1) g^t$

$s^t = \beta_2 s^{t-1} + (1-\beta_2)(g^t)^2$

$p^t_{corr} = \frac{p^t}{1-\beta_1^t}$, $s^t_{corr} = \frac{s^t}{1-\beta_2^t}$

Assumes s^t & p^t are constant. But g^t oscillates, so p^t is not const

$w^{t+1} = w^t - \alpha \frac{p^t_{corr}}{\sqrt{s^t_{corr}} + \epsilon}$

RMSprop vs ADAGrad
- ↳ good for long running training b/c lr is fixed
- good for short running train b/c lr $\propto O(\frac{1}{\sqrt{t}})$ (fast)

Bias - Variance Trade off

Let $\hat{f}_D(x)$ estimate on dataset D

$Bias \equiv E[\hat{f}_D(x) - f(x)] = \bar{f}(x) - f(x)$

$var \equiv E[(\hat{f}_D(x) - \bar{f}(x))^2]$

decomp

MSE: $E_D[(\hat{y} - y)^2] = E_D[((\hat{y}_D - \bar{y}) + (\bar{y} - y_t))^2]$ (var, Bias)

$= E_D[(\hat{y}_D - \bar{y})^2] + E[(\bar{y} - y)^2] - 2E_D[(\hat{y} - \bar{y})Bias]$

$= var(\hat{y}_D) + Bias^2 - 2(E_D[\hat{y}] - \bar{y})Bias$ ($E_D[\hat{y}] = \bar{y}$, → 0)

$= var(\hat{y}_D) + Bias^2$

⇒ overfit if var↑, underfit if Bias↑

Regularization

ex) lin. regression w/ L2. Assume $A_{ij} \sim N(0, \sigma)$

$L(A) = \sum_i (Ax_i - y_i^T)^2 + \lambda \sum A_{ij}^2$

$\nabla L(A) = 2AM_{xx} - 2M_{yx} + 2\lambda A = 0$ low var

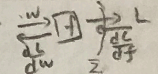
$A = M_{yx}(M_{xx} + \lambda I)^{-1}$ is stable
- ↳ eval of $M_{xx} + \lambda I$ is $\lambda_1 + \lambda, \lambda_2 + \lambda, \ldots$

$(M_{xx} + \lambda I)$ is PSD & $\frac{1}{\lambda_i + \lambda}$ is upper bounded by $\frac{1}{\lambda}$

⇒ $\frac{1}{\lambda}$ can be controlled by λ ⇒ var↓, but bias↑

- λ↑ → var↓; Bias↑
- λ↓ → var↑, Bias↓

Backprop



$\frac{dL}{dw} = \frac{dL}{df}\frac{df}{dw}$

$J_L(w) = J_L(f) J_f(w)$ (vector, vector, matrix)

$J_f(w) = \begin{bmatrix} \frac{\partial f_1}{\partial w_1} & \cdots & \frac{\partial f_1}{\partial w_n} \\ \frac{\partial f_k}{\partial w_1} & \cdots & \frac{\partial f_k}{\partial w_n} \end{bmatrix}$

Gradient $\nabla_A L \equiv J_L(A)^T$

N-step Chain Rule

$J_L(A) = J_L(k) \cdots J_C(B) J_B(A)$ for $L(A) = L(k(\cdots B(A)))$

Backprop → first layer

$\frac{dL}{dx} = \sum_i \frac{dL}{dy_i}\frac{dy_i}{dx}$

BP w/ cross Entropy loss: $L = -\sum y_i \log x_i$

$J_L(x)_i = (\nabla_x L_i)^T = -\frac{y_i}{x_i}$

BP w/ Multiclass logistic Reg: $y_i = \frac{e^{x_i}}{\sum e^{x_j}} = \frac{f}{g}$

$\frac{\partial y_i}{\partial x_j} = \frac{f'g - g'f}{g^2} = \frac{e^{x_i} g \delta_{ij} - e^{x_i}e^{x_j}}{g^2}$ s.t. $\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{ow} \end{cases}$

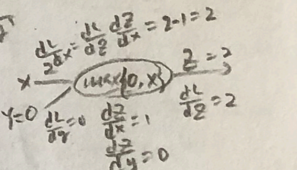
$= y_i \delta_{ij} - y_i y_j$

$J_L(x) = J_L(y) \circ y^T - J_L(y) y y^T$

BP w/ Relu: $y_i = \max\{0, x_i\}$

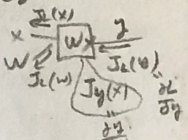
$J_y(x) = \frac{\partial y_i}{\partial x_j} = \begin{cases} 1 & \text{if } i = j \;\&\; x_i > 0 \\ 0 & \text{ow} \end{cases}$

$J_L(x) = (y > 0)^T \circ J_L(y)$



BP w/ FC layer: $y_i = \sum_j W_{ij} x_j$

$J_L(x) = J_L(y) W$
- $J_y(x)_{ij} = \frac{\partial y_i}{\partial x_j} = \frac{\partial}{\partial x_j}(\ ) = W_{ij}$
- $J_y(x) = W$



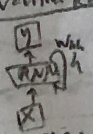
$\frac{\partial L}{\partial W_{ik}} = \frac{\partial L}{\partial y_i}\frac{\partial y_i}{\partial W_{ik}} = J_L(y)_i x_k = \frac{d\,Loss}{d(params)}$ = Model path

$J_L(w) = J_L(y)^T x^T$

$J_L(x) = \frac{d\,Loss}{d(\text{input vector})}$ = Data path

RNN: Notion of time + sequence

vanilla RNN



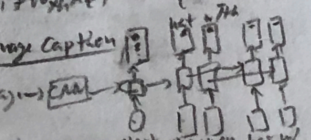
$h_t = f_W(h_{t-1}, x_t)$

$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t) = \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$

$y_t = W_{hy}h_t$

- one-one
- one-many: image caption
- many-one: movie genre
- many-many: translation, stock pred, newsheadline generation

ex) Image Caption



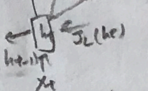
↳ **Beam search**
- ↳ maintain top k captions for each k inputs

Backprop

$h_t = \tanh(\tilde{h})$

$y_t = W_{hy}h_t$

$J_{h_t}(h_{t-1}) = \frac{\partial}{\partial h_{t-1}}\tanh(\ ) = \frac{1}{\cosh^2 \tilde{h}} W_{hh}$

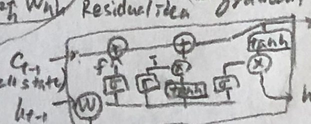


- if $\lambda_{max}(W_{hh}) > 1$, grad → ∞
- if $\lambda_{max}(W_{hh}) < 1$, grad → 0

⇒ soln: (1) Grad clipping (2) LSTM

LSTM: add memory path where no matrix multiply nor non-linearity ⇔ gradient doesn't → 0/∞

↓ Residual idea



- (1) forget
- (2) how to remember / output
- (3) C grows in linear!!
- h_t grows in exp, but ok

$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$

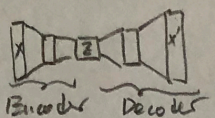
$\frac{\partial}{\partial C_{t-1}} = \frac{\partial}{\partial C_{t-1}} + f_t \circ \frac{\partial L}{\partial C_t}$

Generative Model

classifier: Given (x, y), find the probability $P_{x,y}(x, y)$
Generator: Given y, generate sample from $x \sim P(\cdot, y)$

① Auto Encoder

↳ Given a sample x, transform x into a "code" variable z then reconstruct x' which is close to original x



$P(z)$ training is fast, but
issue: { sampling (test time) is slow
{ inverse may not be unique

soln: VAEs

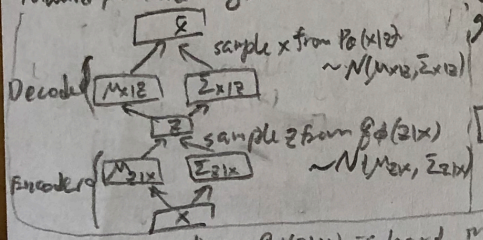
② VAEs

↳ build a decoder as a probabilistic model $P_\theta(x|z)$
↳ invert using variational inference to get approximation $P_\theta(z|x)$
↳ construct an approximate encoder from decoder

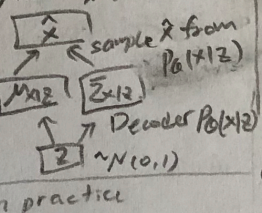
Encoder
↳ approximate inverse density $q_\phi(z|x)$ w/ NN w/ param ϕ

Decoder
↳ Generator(θ) generate N samples from $x \sim P_\theta(x|z)$

Forward path training:



@ test time, generate x from $z \sim N(0, I)$



issue: computing $q_\phi(z|x)$ is hard in practice
soln: Reparameterization Trick
↳ instead, compute $z = g_\phi(x, \varepsilon)$ s.t. $\varepsilon \sim N(0, I)$

- loss func for VAE

↳ max likelihood $P(x) = \max_x \log P(x^i)$

$$\geq \max_x \mathbb{E}_{z \sim q_\phi(z|x^i)}\left[\log\left(\frac{P_\theta(x^i, z)}{q_\phi(z|x^i)}\right)\right]$$

variational lowerbound (convex)

issue: $L(\theta, \phi|x)$ is not accurate w/o sampling estimate
soln: KL-Divergence

$$L(\theta, \phi|x) \approx -D_{KL}(q_\phi(z|x) \| P_\theta(z)) + \frac{1}{L}\sum \log P_\theta(x|z^i)$$

no sampling needed ⇒ reduce variance

log sampling

issue w/ VAE
① Density estimate for $q_\phi(z|x)$
$z = g_\phi(x, \varepsilon) = g_0(x) + g_1(x)\cdot\varepsilon$ assumes ε is indep
⇒ image becomes blurry
↳ soln: Autoregressive Model
② approximation w/ $D_{KL}(q_\phi(z|x) \| P_\theta(z))$
↳ soln: GAN

③ GAN

want: avoid approximations
⇒ ① compare real & synthetic images in image space (discriminator)
② use $x = g(z)$ that is general
↳ Generator

↳ learns a transformer $P(x|z)$ from $P(z) \sim N(0, I)$
& doesn't estimate density $P(x)$ from image x

Discriminator
↳ use JS divergence b/w real $P_{re}(x)$ & synthetic $P_\theta(x)$

$$JSD(P_{re}(x) \| P_\theta(x)) = \frac{1}{2}\mathbb{E}_{x \sim P_{re}(x)}\left[\log \frac{P_{re}(x)}{m(x)}\right] + \frac{1}{2}\mathbb{E}_{x \sim P_\theta(x)}\left[\log \frac{P_\theta(x)}{m(x)}\right]$$

where $m(x) = \frac{1}{2}(P_{re}(x) + P_\theta(x))$ = avg image

Adversarial Training

$$\min_\theta \max_\phi \mathbb{E}_{x \sim P_{re}(x)}\left[\log \frac{P_{re}(x)}{m(x)}\right] + \mathbb{E}_{x \sim P_\theta(x)}\left[\log \frac{P_\theta(x)}{m(x)}\right]$$

↑ train discriminator ↑, train generator to minimize JSD to fool discriminator

① DC GAN
↳ adopted from CNN

② Conditional GAN (CGAN)
↳ use image + label

③ AC GAN
↳ conditioned on class label
↳ discriminator produces class label as output

issue w/ GAN
① Mode collapse ⇒ soln: unrolled GAN
② Difficulty in training ⇒ soln: Wasserstein GAN (W-GAN)
↳ replace JSD w/ wasserstein distance
③ Difficulty in understanding global structure ⇒ soln: Progressive GAN
↳ incrementally add layers
④ Difficulty in evaluating GAN ⇒ soln: Inception score, Frechet inception score
- Big-GAN
↳ better up/down sampling
⑤ unpaired conditional image generation ⇒ soln: X-GAN; Cycle GAN

- Autoregressive Model

ex) Image Transformers

↳ pixels are highly correlated locally
↳ predict new pixel conditioned on already predicted pixel
↳ Fully generative model (can compute density)

issue: condition on all pixels already predicted is expensive
soln: local Autoregressive model
↳ condition on a few nearby pixels

① PixelCNN ⇒ $O(1)$
② PixelRNN (RowLSTM) ⇒ @train: $O(\#\text{rows in image})$, @test: $O(\#\text{rows}^2)$
③ PixelRNN (diagonal BiLSTM) ⇒ use all prev pixels

CNN

assumes { - locality : important info is contained locally
{ - locality invariance : can reuse same param for whole image

output layer Dim (input size, pad, filtersize)

$D_W = D_H = \lfloor \frac{(W + 2P - F)}{\text{Stride}} \rfloor + 1$

D_{depth} = # filters

ex) { input : 32×32×3
filter : 10 5×5
stride : 1
pad : 2

$D_{out} = 1 + \frac{(32 + 2 \cdot 2 - 5)}{1}$

⇕ 1-1 conv reduces channel/depth

 # parameters

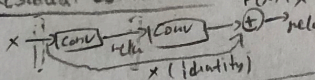
$[(f\_W)*(f\_H)*(f\_depth) + 1] * (\#\text{filters})$
(filter dim) (1 = bias)

ALEX Net
↳ uses Relu, conv → Relu → max pool → Batch Normal
↳ need big data to train ⇒ soln : Transfer learning
↳ if small data, fix all weights & train classif
if medium, use weights for initialization

VGGNet
↳ increase # layers & smaller filters (⇐ small # param)
↳ expensive & use big memory & many params
(memory ↳ CNN; many params ↳ last FC layer)

GoogleNet
↳ Inception module : good for human like behavior
↳ run diff kinds of CNN in parallel
issue : computationally expensive b/c depth grows ⇒ soln : Bottlenecking (1×1 conv layer) ↓ reduce # params

ResNet
↳ add many layers : deep NN is hard to optimize, but maybe they are close to Identity
↳ Residual Connection F(x)+x



Dense Network
↳ connect all layers each other
↳ memory expensive

decay lr : $\alpha = \alpha_0 e^{-x}$

Computer Vision

① classification : classify single obj
② classif + localization : { classify single object
{ bounding box

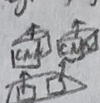
Intersection over Union (IoU) $\equiv \frac{\text{Area of Intersec}}{\text{Area of Union}} \in [0,1]$

Image → CNN → affine → classif → softmax loss ↘
→ affine → Box coord → L2 loss → ⊕ → weighted sum loss

③ **Object Detection** : for each object, { ① category label
{ ② bounding box

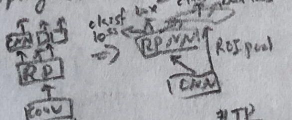
Idea1 : Sliding window regression classif
- need to check many position ⇒ soln : region proposal

Idea2 : R-CNN (region proposal + CNN)
- finding region prop takes long
- classify each region takes long

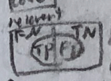


Idea3 : Fast R-CNN (swap conv layer & region proposal)
↳ ROI pooling (efficient cropping of resizing features)

Idea4 : Faster R-CNN
↳ add NN for region proposal
↳ use anchor box



↳ use mAP (mean avg precision)

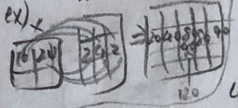


{ recall = # TP / # relevant
{ precision = # TP / # selected

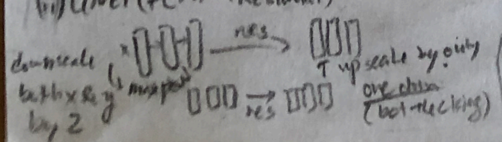
④ **Instance Segmentation** : pixels labels for each thing object ⇒ Mask R-CNN

⑤ **Semantic Segmentation** : label every pixel for both thing & stuff

(i) CNN w/ downsampling & upsampling
downsampling ↳ pooling/strided conv
upsampling ↳ { - replicate nearest neighbor
- max unpooling
- Transpose Convolution →



(ii) UNet (FCNN + Residual)



Training NN

① Sigmoid ; $\sigma(x) = \frac{1}{1+e^{-x}} \in (0,1)$ → good for non linear control

{ - Not zero-centered ⇒ { if input > 0, $\frac{dL}{dw} > 0$
{ if input < 0, $\frac{dL}{dw} < 0$ ⇒ in high dim, grad becomes zig zack
↳ soln : zero-mean data!
- saturation as $x \to \pm\infty$, ($\frac{d\sigma}{dx} = 0$)
- exp is expensive

② tanh $\in (-1, 1)$
↳ good for RNN
↳ zero-centered
But saturation

③ Relu ; $\max\{0, x\} \in [0, \infty)$
↳ good for CNN
↳ converge fast in CNN
{ - not zero-centered
- grad = 0 for x < 0 ⇒ soln : ④ Leaky Relu $\max\{0.1x, x\}$
- Dead Relu

⑤ ELU $f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{o} \end{cases}$
↳ close to zero-mean
↳ avoid discontinuity
↳ saturation add robustness

⑥ Maxout : $\max\{w_1^T x + b_1, w_2^T x + b_2\}$
↳ doesn't saturate
- ×2 # paremeters

rand weight initializ
$w_i \sim N(0, \sigma^2)$
- $\sigma^2 \to 0$ as go deep } want to break symmetry
↳ soln : Batch Normal

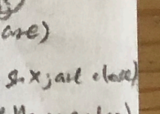
Xavier Initializ allows $\sigma_{in} = \sigma_{out}$ @ every layer
F_{in} ≡ # input neurons that contribute to output
ex) for CNN, $F = F_H \times F_W \times C$
To get $F_{in} = F_{out}$, need $Var(w) = \frac{1}{F_{in}}$

Batch Normalization

$\hat{x}^k = \frac{x^k - \mathbb{E}[x^k]}{\sqrt{Var(x^k)}}$

↳ grad doesn't blow up
↳ can use higher learning rate
↳ Reduce the need of dropout b/c $\frac{1}{\sqrt{\sigma^2}}$ is stochastic, it has regularization effect
@ test, use running avg & stddiv

Random Search



Dropout
↳ randomly set neuron to 0 w/ p
↳ force to learn redundant info
⇒ Inversion dropout
@ train, remain / (1-p)
@ test, do nothing

Gradient Clipping
↳ clip huge grad when misclassify outlier

Gradient Noise
↳ $g_t \leftarrow g_t + N(0, \sigma_t^2)$ for regularization

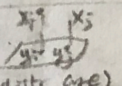
Ensemble
↳ train k indep models & @ test, take avg
① Bagging (parallel) reduce var
↳ train base model on bootstrap samples & take majority vote/avg
↳ good when model doesn't have bias & variance is high
↳ use when models are same
↳ good for deep Net
- take long time to train
② Boosting (sequence) → misclassified sample get higher weight
↳ Each learner tries to reduce error on hard examples
↳ good when we want to reduce bias & possibly variance
∴ not for NN

How to implement?
① True ensemble : train a few models indep @ tail layers &
(i) avg the prediction probability (almost always work) b/c models are trained indep
(ii) avg the parameters (Never works b/c too many possible space)
② Model snapshot : keep training a single base model & take periodic snapshot of parameters
(i) prediction avging : always work
(ii) parameter avging : often works b/c parameter space is narrow
↳ only need to keep a single model for future prediction

Visualization

t-SNE (Stochastic Neighbor Embedding)
↳ preserve locally pairwise distance
↳ KL divergence (measure of how diff 2 distr are)

$D_{KL}(P \| Q) = \sum_{i,j} P_{j|i} \log\left(\frac{P_{j|i}}{q_{j|i}}\right)$ st. $P_{j|i} = P(x_i \,\&\, x_j \text{ are close})$, $q_{j|i} = P(y_i \,\&\, y_j \text{ are close})$



prop { $D_{KL}(P \| q) = 0$ iff p = q almost everywhere
{ $D_{KL}(P \| q) \geq 0$ ∀ p, q .

↳ because cluster is not sphere, k-means doesn't work ⇒ use DBSCAN, HDBSCAN, etc

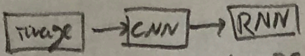
① visualize filters ⇒ only works in low layers
② Saliency Approach ⇒ generate image that maximize class score
③ Gradient ⇒ pick layer & backprop
↳ Deconv : ↳ (dout ≥ 0) × dout
Guided Backprop ↳ remove 0 or negative = (f(x) > 0)(dout > 0)(dout)
④ Neural Style Transfer

Attention
- learns sailiency
- make NN interpretable
- reduce computation

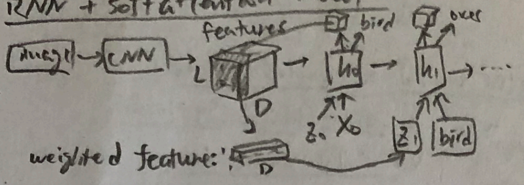
soft attention
↳ weighted combo/attention of some inputs
↳ differentiable ⇒ can back prop

Hard Attention
↳ attend to a single input location
↳ not differentiable ⇒ need RL

RNN - captioning Model

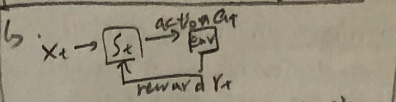


issue: don't remember parts of image
⇓
RNN + soft attention Model



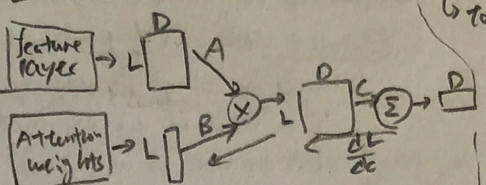
weighted feature:
D = # features
L = width × height

Reinforcement Learning (RL)



↳ maximize reward @ each step

Soft Attention mechanism



$\frac{\partial L}{\partial B} = \sum_{d \in D} A \odot \frac{\partial L}{\partial c}$ = sallience

= { > 0 if increase attention
{ < 0 if decrease "

Statistical Machine Translation (SMT)

$y^* = \arg\max_y P(y|x) = \arg\max_y P(x|y) P(y)$, Language Model

(y: English, x: French; $P(x|y)$: Translation Model)

NLP Challenge
↳ Tokenization challenge
1. <UNK> token
2. Increase vocab size
3. word piece tokenization
   ↳ break down unknown words

↳ Tokenization is not unique & not reversible

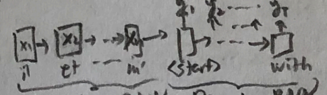
- Unresolved issue w/ translation
  - out-of-vocab words
  - Domain mismatch
  - maintain context over long text
  - what if we have little text available?
  - machine don't have common sense
  - learns bias (race, gender, etc)

Neural Machine Translation (NMT)

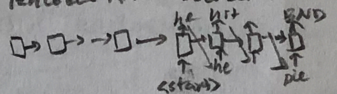
- Seq2Seq RNN
  ↳ conditional lang Model
    ↳ directly calculates $P(y|x) = P(y_1|x) \cdots P(y_t|x, y_1, \dots, y_{t-1})$

Loss: $J_1 \cdots J_T = \frac{1}{T}\sum_t^T J_t$

@ Train: O(n)

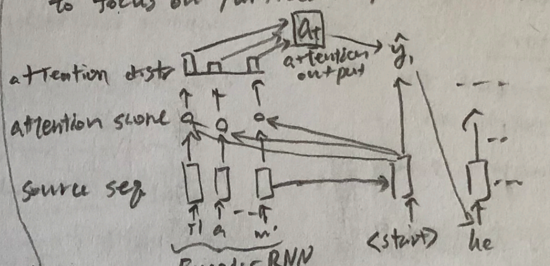


@ test:



issue:
- forgets the beginning of source ⇒ Bi-directional RNN
- bottleneck: single path b/w Encoder & Decoder

- soft Attention for Translation
  ↳ @ each step of the decoder, use direct connection to the encoder to focus on particular part of input



Adv:
- solves bottleneck
- helps vanishing gradient
- finds alignment automatic

issue: hard to generalize to NN
↓
soln: Global Attention

issue w/ RNN Model:
- lack long range dependency
- @ training time, O(n) time

↓ soln

- Blue Score for Translation
  ↳ compare machine translation & human translation

unigram precision + clipping = $\frac{\text{correct unigram in ref sentence}}{\text{unique unigram in test sentence}}$

Blue = $BP \cdot e^{\sum_{n}^{N} w_n \log P_n}$ (mean of N-gram precision; P_n: N-gram precision)

where $BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{ow} \end{cases}$

- English to English (Summarization)

in: { length N passage
    { length M summary

: $O(N^2) + O(M^2) + O(NM)$
Encoder self-att, Decoder self-attention, cross-attention

↳ soln: remove input/encoder self attention
$O(M^2) + O(NM)$

- The Transformer
  ↳ QKV attention
  ↳ O(1) training time

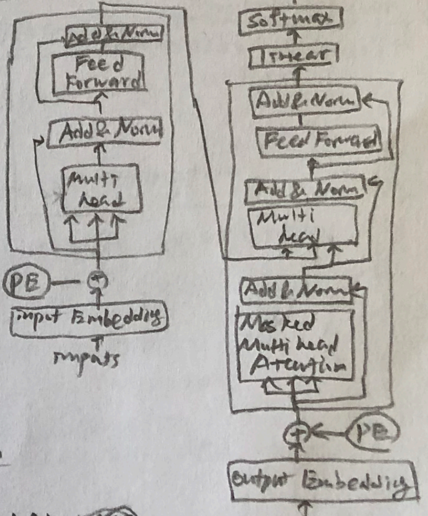
- Self attention
  ↳ Q, K, V come from Encoder/Decoder

- Cross Attention
  ↳ Q come from Decoder
  K, V come from Encoder

- Bidirectional attention
  ↳ each word attend to its left & right
  ↳ used for transformer Encoder
  ↳ can't be used for generation

- Masked Attention
  ↳ each word attend to only to its left
  ↳ used for transformer Decoder
  ↳ can be used for generating next word

- Position Encoding
  ↳ learns relative displacement y-x b/w 2 language



{ @ train, O(1) time!!
{ @ test, O(length of target)

Text Semantics

- Propositional / Formal Semantic
  - ↳ convert text into a formula
  - ↳ good for inference in well-defined domain
  - ex) dog bites man ⇒ bites(dog, man)
- vector representation
  - ↳ embed text into high dim space

① Bag of Words (BoW)

ex) $\vec{vec}("dog") = (0.2, -0.3, 1.5, \dots) \in \mathbb{R}^n$
$\vec{vec}("bites") = (\quad) \in \mathbb{R}^n$
$\vec{vec}("man") = (\quad) \in \mathbb{R}^n$
$\vec{vec}("dog\ bites\ man") = \vec{vec}("dog") + \vec{vec}("bites") + \vec{vec}("man")$

issue: order of words not considered

- Lexical semantics → sum of words
  - ↳ focus on the meaning of individual words

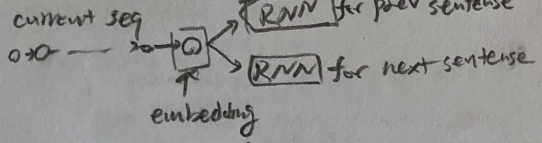
① t-SNE
- ↳ preserve the distance for close pts

② word2vec
- ↳ minimize global distance

③ Co-occurence Matrix

- Compositional Semantics
  - ↳ meaning depends on how words are combined

① skip-thought vectors
- ↳ seq-to-seq. RNNs to predict the next & previous sentences
- ↳ semantic Related Evaluation scores human like score



② Siamese Network
- ↳ semantic relatedness

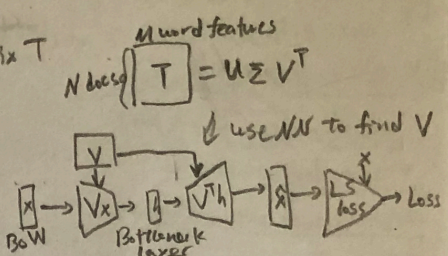
vector Embeddings

↳ paradigmatic word similarity
- ↳ similes words occur in similar context ⇒ can exchange words.

ex) My dog barked, ...
Her dog barked, ...

↳ Dimentionality Reduction (PCA)
- issue w/ vector encoding: len of word vector is vocab size

① Latent semantic Analysis (LSA)
- ↳ encode a set of documents in a matrix T
  - T_{ij} = count of word j in doc i
- ↳ T is sparse ⇒ use SVD
- ↳ auto-encoding method ⇒ use NN:



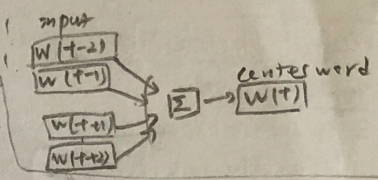
② word2vec
- ↳ focus on local context

skip-gram ≡ a pair of center word & context word
ex) It was a great day yesterday

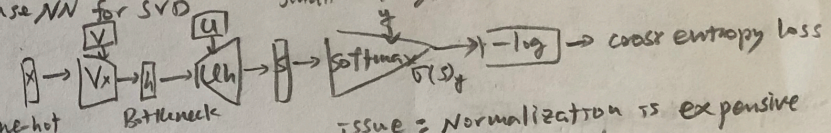
- skip-gram model
  - ↳ predict context word from center word

$$P(j|i) = \frac{e^{u_j^T v_i}}{\sum e^{u_k^T v_i}} \quad \text{s.t.} \begin{cases} u_j = \text{emb of context word } j \\ v_i = \text{emb of center word } i \end{cases}$$

vs CBOW Model
- ↳ predict center word from context word



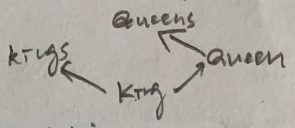
- ↳ SVD minimize dist b/w original & embedding pts
- ↳ capture relationship b/w similes words
  - ex) Japan — Tokyo
  - small — large
- ↳ use NN for SVD



→ coosr entropy loss

issue = Normalization is expensive

- ↳ composition



③ Co-occurence Matrix

- Window-Based Co-occur Matrix
  - ↳ count each pair of words
  - ↳ similes to word2vec

issue:
- increase in size w/ vocab
- classification model is sparse
  - ↳ soln: use SVD

vs Glove Model

$$\min \sum_{i,j}^{V} f(C_{ij})(u_i^T v_j + b_i + \tilde{b}_j - \log C_{ij})^2$$

s.t. $f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha} \text{ if } x < x_{max} \\ 1 \quad \text{ow} \end{cases}$

- ↳ scalable w/ huge corpus
- ↳ efficient

natural lang understanding (NLU) vs Natural Lang Generation (NLG)

↳ Given a text, build a representation of the inputs that can be used for many tasks

ex) Extractive Summarization
↳ extract k sentences that summarizes texts

- Bert (Bidirectional Encoder Representation from Transformer)
↳ masked lang model + Is Next pretrained
↳ select 15% random words & replace w/ <Mask> token
↳ predict what the original words are
↳ predict next sentence

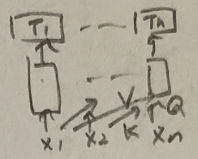
- Fine tune BERT
① Sentence pair classification task.
↳ Given 2 sentences, are they saying same thing? T/F
② Question Answering Task. e.g. SQuAD dataset
③ natural lang inference (NLI)
↳ Given a premise P, classify whether hypothesis H is entailment/contradiction/neutral
④ Corpus of Language Acceptability (CoLA)
↳ Given a sentence, classify if gramatically correct

↳ Given texts, produce a sequence of words

ex) Abstractive summarization
↳ produce a new summary

Perplexity = evaluation score for comparing 2 lang models

- GPT
↳ generative model is computationally efficient & scalable
↳ transformer based generator, but w/o Encoder



NLG tasks
① Conversational Q&A (CoQA)
↳ Given a dialogue (conversation) b/w teacher & student, teacher must generate Answer
↳ issue: coreference : need to be asked previously
② Dialog System (DSTC)

Security to Adversarial Attack

↳ Since Traditional ML assume: Training data = Test data, add adversarial noise to fool model

White Box Attack
↳ optimization based attack

$$\min_{\delta} \|\delta\|_p + J(f_\theta(x+\delta), y^*)$$

(δ: adv. perturbation; y^*: adv. target label)

① Fast Gradient method

$$x^* = x + B\left(\frac{\nabla_x L(f_\theta(x), y)}{\|\nabla_x L(f_\theta(x), y)\|}\right)$$

② Fast Gradient sign method

$$x^* = x + B\,\text{sign}(\nabla_x L(f_\theta(x), y))$$

issue: need to know lot of info (weight, learning rate, etc)

- GAN w/ Adv Network
- Defense to Adv image
↳ ensembling, but GAN can still defeat
- Attack on RL
↳ policy Network Attack.

Black Box Attack
① Zero-query Attack
↳ randomly estimate gradient & perturbation
↳ transferability-Based attack → Targeted attack (mislead to classify specific class) / Non-targeted attack
↳ Train a model on same chosen model (SVM, kNN, etc) & produce adv. example
↳ Ensemble Targeted Black Box attacks

issue: estimating gradient, weights, etc is hard

② Query-Based Attack
↳ Finite differentiating gradient Estimation

Given $x \in \mathbb{R}^d$, we can estimate gradient w/ 2d queries

$$FD_x(g(x), \delta) = \begin{bmatrix} \frac{1}{2}(g(x+\delta e_1) - g(x-\delta e_1)) \\ \frac{1}{2}(g(x+\delta e_d) - g(x-\delta e_d)) \end{bmatrix}$$

issue: need to estimate in all d dimension
soln: Query Reduced gradient Estimation
↳ random feature grouping, PCA, etc