RNN | Systems POV: 1) FIR Filters

Convolution-like form; some maximum receptive field (accepts finite inputs)

- Constant Z-transform
- Response characterizes filter
- A la CNNs / GNNs

2) IIR Filters operate on potentially infinite length signals

- State (evolution over time) encodes dynamics; complex Z-tf
- Multiple poles exist due to recurrent dynamics
- View as streaming/online algo

- Ex Kalman Filtering. Given known dynamic of linear sys. driven by Gaus. noise (known covariance)

$$\begin{cases} x_{t+1} = A x_t + B_u u_t + w_t \\ y_{t+1} = C x_{t+1} + v_t \end{cases}$$

- Asm w_t & v_t iid jointly gaus. s.t. x, y, w, v all jointly " . Following there being linear map $y \to (w, v)$
- Goal: min. $\mathbb{E}[\|\hat{x}_t - x_t\|^2]$

$$\hat{x}_{t+1} = \hat{A}\hat{x}_t + \hat{B}_u u_t + \hat{B}_y y_{t+1}$$

- Init: $\hat{x}_0$ should create stable dyn. could be learnable, or 0/random hoping robustness will get us okay

RNN seq2seq challenges:

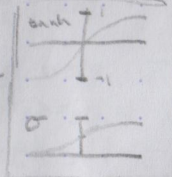
- Requires sequential processing (slow)
- Fixed length context (doesn't work well for long-range dependency)
- Target/source word ordering may differ but can't "align" to fix

↳ Motivates encoder-decoder seq2seq

Non-linearity: Can destroy info, motivates skip

- Non-saturating: f lim $\to \pm\infty$ is $\pm\infty$
  - Ex ReLU (etc.)

- Saturating: Stabalizes dynamics by preventing explosions
  - Ex tanh & sigmoid
  - Apply to recurrent state $h_{t+1} = \sigma(A h_t + \dots)$
    - ↳ Could also layer norm h_t or x_t
    - ↳ Not batch, as this wouldn't effect recurrent aspect
  - Leads to dying grads for extreme values



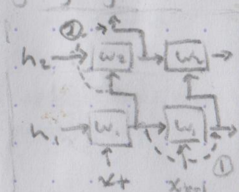
Skips: Tackle dying grads

① leads to too much info/long-range dependence

↳ LSTM solves by gating

↳ Attention solves by using soft hashmap

- Capture hierarchical relations based on similarity measure between states
- Weighted avg. so grads aren't harsh
- Alleviates bottleneck from encoder; enhance feature expressiveness/reduce regularization
- Cross-attn: (k, v) from enc. q from decoder
- Self-attn: (k, q, v) from same source
  - ↳ Decoder often made causal by masking
- Multi-head: Lookup l items over l tables & concatenate
- Multi-query: Lookup l items over 1 table; (k, v) shared across layers/queries
  - ↳ Less params; faster @ eval
  - ↳ Softmax makes 1 primary feature per head
  - ↳ Receptive field whole seq.



Auto Encoders | Linear: $\mathcal{L} = \|X - W_2 W_1 X\|_F^2$

- Projects $W_2 W_1$ onto k largest S.V. of X
  - ↳ $W_2 W_1$ should look like I for $k[\dots]$

  i.e. $W_2 \approx W_1^T \equiv \Sigma_2 \approx \Sigma_1^T$

- Regularizer: $\lambda(\|W_1\|_F^2 + \|W_2\|_F^2)$

induces	$\cong \lambda(\|\Sigma_1\|_F^2 + \|\Sigma_2\|_F^2)$
orthonorm	$= \lambda(\sum_{i=1}^{k} \sigma_i^2 + 1/\sigma_i^2)$
cols W_2	$0 = \nabla[\dots] \equiv 2\sigma_i = 2/\sigma_i^3$
rows W_1	$\Rightarrow \sigma_i = 1$
asm	Unique $\to [\dots] \to \infty$ as $\sigma \to [0, \infty]$

λ small enough to decouple terms

Now let $\mathcal{L} = \|X - WX\|_F^2 + \lambda\|W\|_F^2$

$$= \sum_{i=1}^{m} \|X_i - W_i X\|_2^2 + \lambda\|W_i\|_2^2$$

$\|y - Ab\| + \lambda\|b\|^2 \to \|X_i^T - X^T W_i^T\|_2^2 + \lambda\|W_i^T\|_2^2$

For $y = X_i^T$, $b = W_i^T$, $A = X^T$:

$$b^* = (A^T A + \lambda I)^{-1} A^T y$$
$$W_i^{*T} = (XX^T + \lambda I)^{-1} X X_i^T$$
$$W^* = XX^T(XX^T + \lambda I)^{-1}$$
$$= (U\Sigma V^T V \Sigma U)(U\Sigma\Sigma^T U + \lambda I)^{-1}$$
$$= U\Sigma\Sigma^T U U^{-1}(\Sigma\Sigma^T + \lambda I)^{-1} U^T$$
$$= U(\Sigma\Sigma^T / \Sigma\Sigma^T + \lambda I) U^T$$

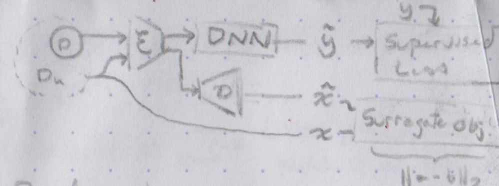
Attn(k, q, v): $e_i = q^T v / \sqrt{d} \sim N(\mu, \sigma^2)$

$\alpha_i = \exp(e_i) / \sum_j \exp(e_j)$

r.t $\sum_i \alpha_i v_i$

- Scaling: $q^T k_i \simeq \frac{1}{d}\sum_{i=1}^{d} x_i \xrightarrow{d} N(\mu, \sqrt{d}\sigma^2)$
  - Ex $x \sim N(\mu, \sqrt{d})$ and sample $x_1 \dots x_n$. $\mathbb{E}[x_i - x_j] = \sqrt{d}/n$ so dist. between pts $\propto \sqrt{d}$, enlarged by softmax (acts like hardmax at init b.c. winner wins by scale/dumb luck)
  - Had we normalized by d, we'd constrain $q, k \in [0, 1]$ so sim. hardly differ (all small)
    - ↳ Prior, aligned case magnitude $\frac{d}{\sqrt{d}}$, here it's $\frac{d}{d} = 1$
  - Improper scaling $\Rightarrow$ large loss causing softmax to give small grads (slow learning)

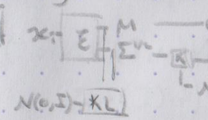
Aside: $T = 0 \equiv$ pick best



Regularization:

- Adding noise, $x_i + \eta$ for $\eta \sim N(0, \sigma^2)$ makes network de-noise, which requires understanding of implicit structure of x_i
- Masking $\equiv$ subtractive noise

VAE



ViT: $O((H/P \times W/P)^2 D)$

MQA only computes XW_k & XW_v once saving $2Md^2(h-1)$ operations

For input seq $= M$, output $= N$, h heads $W_k, W_q, W_v \in \mathbb{R}^{d \times d}$

Encoder SA takes $3Md^2$ for $\{XW_i\}_{i=1}^{3}$ compute attn mat. $M^2 d$ and $O(M^2)$ softmax all over h heads mul w/value $M^2 d$

- Enc. SA/MQA: $O((Md + M^2)dh)$
- Cross Attn: $O((Md + Md + MN)dh)$
- Decoder SA: $O((Nd + N^2)dh)$

Pf Have $k, q \sim N(\mu, \sigma^2 I)$; $\mu = 0, \sigma = 1$

$Var(q^T k) = Var(\sum \dots) = \sum_{i=1}^{d} Var(q_i k_i)$

$\dots = d \times (\mathbb{E}[q_i]^2 Var(k_i) + \mathbb{E}[k_i]^2 Var(q_i) + Var(q_i)Var(k_i))$

$[Alt] = \mathbb{E}[(q^T k)^2] - \mathbb{E}[q^T k]^2$

$= \mu^T \Sigma_q \mu + \mu \Sigma_k \mu + tr(\Sigma_k \Sigma_q)$

$= 2\sigma^2\|\mu\| + d\sigma^2$

So $Var(q^T k / s) = d / s^2 = 1 \iff s = \sqrt{d}$

> Sample $T_i \sim p(T)$
>
> For all T_i: [Calc grad over k samples]
> $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{T_i}(f_\theta)$
>
> $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{T_i} \mathcal{L}_{T_i}(f_{\theta_i'})$

$h(x) = \frac{f(x)}{g(x)}$; $h'(x) = \frac{f'(x) g(x) - f(x) g'(x)}{g^2(x)}$

Diffusion

$H(Y) = \mathbb{E}_Y[-\log(p(Y=k))]$

$H(p,q) = -\sum_k p(Y=k)\cdot\log(\,\text{''}\,)$

$= \mathbb{E}_{p(x)}[\log 1/q(x)]$ — Negative log likelihood

$= -\sum_x p(x)\log(q(x))$

$D_{KL}(p\,\|\,q) = \sum_x p(x)\log\left(\frac{p(x)}{q(x)}\right)$

Fixed P, minimizing D_{KL}

$= \mathbb{E}_{p(x)}[\log p(x)] - \mathbb{E}_p[\log q(x)]$

$= -H(p) + H(p,q)$

$D_{KL}(N(\mu,\Sigma)\,\|\,N(0,I))$

$= \frac{1}{2}\left(tr(\Sigma) + \mu^T\mu - k - \log\det\Sigma\right)$

Draw n samples ~ P.

$\mathbb{P}[\text{look like } Q] = \exp(-n\,KL(Q\|P))$

If data is "clean" (ie high SNR) (categories fairly distinct) then classifier has lots of margin; not needed to pick out true characteristics over noise

Adversarial Ex

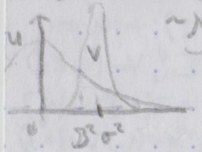
$\underline{x} \in \mathbb{R}^{d+1}$ st. $x_1 \sim N(0,\sigma^2)$, $x_i \sim N(0,1)$ $\forall i = 2..d+1$

w/ class$(\underline{x}) = \text{sgn}(x_1)$

learned cls $= \text{sgn}(\underline{\alpha}^T\underline{x})$

$\alpha_1 = \beta,\ \alpha_i = 1 \rightarrow = \beta x_1 + \sum x_i$

$\sim \underbrace{N(0,\beta^2\sigma^2)}_{U} + \underbrace{N(0,d)}_{V}$



when can we trust, e.g. doesn't pass origin

$\hookrightarrow \beta^2\sigma^2 \gg d \equiv \beta\sigma \gg \sqrt{d}$

Consider $\underline{x}' = \underline{x} + S\underline{\alpha}$ (S is adversary noise mag.)

st. $\text{score}(\underline{x}') = x^T\alpha + S\alpha^T\alpha$

$\approx U + V + S(\beta^2 + d)$

asm. $\sigma \gg \beta$ so nature of x not changed to x'

Can flip if $d \gg \beta\sigma = \text{Var}(U)$

Say $\sigma = d^a,\ \beta = d^b$ giving

$a + b > 1/2$ Eg $a = 1/2$

$a > b$ $b = 1/4$

$1 > a + b$

P.E.] Applied into queries + keys

Relative: Each head has learnable bias term $\langle q_j, k_i\rangle + b_{j-i}$

- Makes nearby elts. bigger
- Doesn't increase dimensionality like RoPE; no slow down

RoPE: $q = W_q\underline{x} \rightarrow q = \Theta_t W_q\underline{x}$

where $\Theta_t = \text{diag}(\exp(jt\theta_1), \ldots, \exp(jt\theta_{d/2}))$

$\theta_i = 10{,}000^{-2(i-1)/d}$

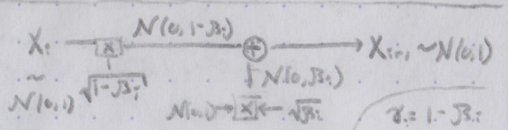
- No learnable params
- Depend just on relative locations

Ex.] Given $\underline{x} \in \mathbb{R}^{2n}$

Represent complex: $\underline{z} = [x_1 + ix_2 \ldots x_{2n-1} + ix_{2n}]$

$\text{RoPE}(\underline{z}; t) = [z_1\exp(it\theta_1) \ldots z_n\exp(it\theta_n)]$

$\hookrightarrow$ n learnable params



$\beta \approx 0$ returns signal $\gamma_i = 1 - \beta_i$, $\varepsilon_i \sim N(0,1)$

$x_t = \sqrt{\gamma_t}\,x_{t-1} + \sqrt{1-\gamma_t}\,\varepsilon_{t-1}$

$= \sqrt{\gamma_t}\left(\sqrt{\gamma_{t-1}}\,x_{t-2} + \sqrt{1-\gamma_{t-1}}\,\varepsilon_{t-2}\right) + \sqrt{1-\gamma_t}\,\varepsilon_{t-1}$

$= \sqrt{\gamma_t\gamma_{t-1}}\,x_{t-2} + \sqrt{\gamma_t(1-\gamma_{t-1})}\,\varepsilon_{t-2} + \,\text{''}$

$= (\,\text{''}\,) + \sqrt{1-\gamma_t\gamma_{t-1}}\,\hat{\varepsilon}_{t-2}$

$N(0, \gamma_t(1-\gamma_{t-1}) + (1-\gamma_t)) = N(0, (1-\gamma_t\gamma_{t-1})I)$

$= \sqrt{\prod_i^t \gamma_i}\,x_0 + \sqrt{1-\prod_i^t \gamma_i}\,\varepsilon_0$ ($\bar{\alpha}_t$)

$\therefore q(x_t|x_0) = N(x_t; \sqrt{\bar{\alpha}_t}\,x_0, (1-\bar{\alpha}_t)I)$

Add

$q_1^T k_2 = (x_1+p_1)^T(x_2+p_2)$

$= x_1x_2 + p_1p_2 + x_1p_2 + x_2p_1$

- Most vectors ⊥ in high-dim, so cross terms dominated

Concat

$q_1^T k_2 = x_1x_2 + p_1p_2$

- only compare words/words, pos/pos
- Position term not dumb — constructed s.t. nearby positions not ⊥

Norm Vectors

Let $\underline{z} = f(\underline{x}) = \underline{x} - \mu\mathbb{1}$

For $\mu = \frac{1}{d}\mathbb{1}^T\underline{x}$ $\rightarrow = (I - \frac{1}{d}\mathbb{1}\mathbb{1}^T)\underline{x}$

so $g(x) = \nabla_x\mathcal{L} = \frac{\partial\mathcal{L}}{\partial z}\cdot\frac{\partial z}{\partial x} = (I - \frac{1}{d}\mathbb{1}\mathbb{1}^T)g(z)$

Let $y = h(z) = \frac{z}{\sigma} = \sqrt{d}\,z/\|z\|_2$

For $\sigma = \sqrt{\|z\|_2^2/d}$

so $g(z) = \frac{\partial\mathcal{L}}{\partial y}\cdot\frac{\partial y}{\partial z} = \sqrt{d}\left(\frac{I\|z\|_2 - z\frac{\partial}{\partial z}[\|z\|_2]}{\|z\|_2^2}\right)$

$= \frac{\sqrt{d}}{\|z\|_2}\left(I - \frac{zz^T}{\|z\|_2^2}\right)g(y)$

For $z' = \alpha\cdot z$

$\|f(z')\|_2 = \alpha\|f(z)\|_2$

$\|g(z')\|_2 = \|g(z)\|_2$

$\|h(z')\|_2 = \|h(z)\|_2$

$\|g(z')\|_2 = \frac{1}{\alpha}\|g(z)\|_2$ — Speedbump preventing exploding grads

Forgetting mitigation

Boils down to regularization (bigger is better here)

1] Freezing. Early layers for if we're in new data domain. Later layers if new task

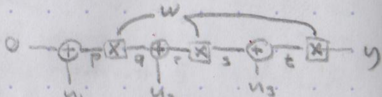
$\rightarrow$ Could also lower LR on per-basis level

2] (Replay) Combine OG data with new data. If new task, graft head and keep OG for supervision

Linear Probing. [illegible]

RNN backprop

$h_t = w(u_t + h_{t-1})$



$\frac{\partial\mathcal{L}}{\partial w} = \sum_{i=1}^T \frac{\partial\mathcal{L}}{\partial h_i}\cdot\frac{\partial h_i}{\partial w}$

$y = wt = wu_3 + w^2u_2 + w^3u_1$

$\frac{\partial y}{\partial w} = u_3 + 2wu_2 + 3w^2u_1$

$= t + r\frac{\partial y}{\partial s} + p\frac{\partial y}{\partial q}$

$= u_3 + wu_2 + w^2u_1 + (u_2 + wu_1)(w) + (u_1)(w^2)$

$\frac{\partial y}{\partial t} = w = \frac{\partial y}{\partial s}$

$\frac{\partial y}{\partial r} = \frac{\partial y}{\partial s}\cdot w$; $\frac{\partial y}{\partial q} = \frac{\partial y}{\partial r}\cdot 1$

Finetuning

1] Prompt Tuning. Essentially make first layer's keys learnable

- Vectors more free than tokens (english)
- Works best on large models

$p(x) = \int p(x,z)\,dz$ intractable

$= p(x,z)/p(z|x)$ — unknown, what we want to learn!

surrogate motive: $p_\theta(x|z) \approx q_\phi(x|z)$

ELBO maximize..

$\log p(x) = \log p(x)\int q(z|x)$

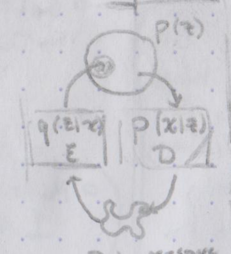
$= \mathbb{E}_{q(z|x)}[\log p(x)]$

$= \mathbb{E}\left[\log\frac{p(x,z)}{p(z|x)}\cdot\frac{q(z|x)}{q(z|x)}\right]$

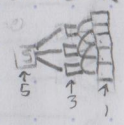
$= \mathbb{E}_q\left[\log\frac{p(x,z)}{q(z|x)}\right] + \underbrace{D_{KL}(q(z|x)\,\|\,p(z|x))}_{\geq 0}$

$\geq \mathbb{E}_q\left[\log\frac{p(x|z)p(z)}{q(z|x)}\right]$

$= \underbrace{\mathbb{E}[\log p(x|z)]}_{\text{reconstruction quality}} - \underbrace{D_{KL}(q(z|x)\,\|\,p(z))}_{\text{learned dist. diff}}$



If blurry @ eval, $\underbrace{q(z|x)}_{\text{learned map}}$ is far from $\underbrace{p(z)}_{\text{eval sampler}}$

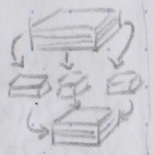
Conv vs | Input: $C \times H \times W$ | $C \times K \times K$ filter
Size (K), padding (P), stride (S), number filters (F)
Weights: $K^2 \times F \times C$ } Learnable parameters
biases: F }
$W' = \lfloor (W - K + 2P)/S \rfloor + 1$, $H' = \text{etc}$
$C' = F$
Receptive Field: $L(K-1)+1 = \mathcal{O}(LK)$ (linear)
↳ L successive conv layers of size K
↳ Formula for stride = 1, stride makes increase faster than linear

↳ Three 3×3, same receptive F. as 7×7
- But former has non-lin.; more expressive
- $3\times(C\times(3\times3\times C)) = 27C^2$ v.s. $C\times(7\times7\times C) = 49C^2$ parameters

- General: $r_{i+1} = s_i \times r_2 + k_{i+1} - s_{i+1}$

Weight Sharing: Rather than per-pixel weights, we have per-patch weights that we apply over whole img.
- Less parameters / slow blowup
- Introduces Translation Invariance

Data Augmentation: Introduce insignificant (wrt classific. problem) perturbations to data s.t. model learns these invariances
- SGD moves along weighed avg. of img patches, so augmentation can be thought of as regularizing noise

Depthwise: Filters unique to each channel of input
- Less expensive, now quite common



Pooling | $W' = (W-K)/S + 1$
Receptive Field $\sim K$ (assm $s=1$)
Spatial resolution $\sim 1/K^2$ (in 2D)
↳ Why we double output channels

Global Avg: Over all channels
- Done prior to SoftMax, rather than flatten: reduces from $H\times W\times C$ to C
- Looses spatial info; okay for classification (present or not; binary) but not segmentation
- Enables us to use sized input: output solely depends on C.

Max: Take highest value of $K\times K$ pool
- Induces learning "invariant" features
- High-pass: retains "strongest" signal

$$\frac{\partial \chi}{\partial x_{ij}} = \sum_{y,a,b} dy_{ab} \cdot \frac{\partial y_{ab}}{\partial x_{ij}}$$

$$\overset{\substack{k=i/r \\ l=j/r}}{=} dy_{kl} \times \mathbb{1}\{x_{ij} = \max\{x_{i,j},x_{i,j+1},x_{i+1,j}\}\}$$

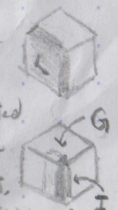
Avg: Reduces spatial domain by downsampling; retains some info about distribution of patch's values
- Better at capturing smooth transitions / subtle changes than Max pool

$$\frac{\partial \chi}{\partial x_{ij}} = \sum_{y,a,b} dy_{ab} \cdot \frac{\partial y_{ab}}{\partial x_{ij}} = dy_{nl} \cdot \frac{1}{4}$$

UNET | For pixel-level classification
↳ Double channels when downsampling s.t. "total info" is const.
- Low-res, high-d (middle) integrates contextual info from whole img
- Up-sampling turns low-res feature vectors into high-res per-pixel preds
↳ Combine with details (raw info) from earlier

Normalizing | If we move somewhere, we want a reason - not b.c. it's big.
- Note: Adam already normalizes grad. over every param. by the avg. across many batches
- Go to $N(0,1)$: $(x - \mathbb{E}[x])/\sigma[x]$
↳ operations can be undone by weights + biases; no reduction in expressive power
↳ Large SV $\Leftrightarrow$ input correlated

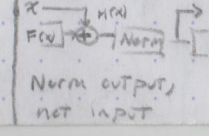
1.) Batch: Norm each channel in a batch by μ/σ of each Feature (all channel pixels)
↳ Treats channels as unrelated
↳ Backprop grad over whole batch; non-trivial implement.
↳ Test time: Use stored $\mu^{(t)}/\sigma^{(t)}$



↳ Suitable for convnets, not FC layers (no syms / spatial awareness)
↳ Tackles internal covariate shift
2.) Layer: Norm. over all pixels in each channel
- E.g. later layers in network when Features should be "learned" and thus sizes should be alike
- Can group-norm (only some of channels) too
- Faster than Batch

3.) Instance: Norm across each channel in each img/F
- Agnostic to contrast of input

- Practically: Inits + Norm + Conv. structure isn't enough for training
- Grads become too small
- Backprop signal noised
↳ Solved w/ skip (or residual) connections
- Decouple behavior: enables finding paths
↳ Tackles internal covariate shift



Inits | Motive: Don't want dead ReLU or wasted computation
- Usually unable to recover

Weights:
- 0: Can't backprop more than 1 layer; always zero gradient
- Ideally, inputs follow $N(0,1)$ dist
1) Xavier: $\sigma^2 = 1/d$, d = Fan-in
- Appropriate for sigmoid / tanh; not ReLU
↳ Saturating fcns; go to some $\neq 0$ const
2) MC: $\sigma^2 = 2/d$
- Before, half of ReLUs begin off so Fan-in is halved
↳ Solves Xavier's vanishing grad problem

Biases:
- Xavier with $d+1$; treat as weight
- 0 (optimizer to set within first cycle)
- 0.01 Empirically better than 0
- Any small random #. (doesn't matter as we expect optimizer to set)

Convnext | Expand to higher dim, then compress/shrink down
- Each layer modifies input, rather than completely transforming it.
- Classification/obj Features inherently low-dim. (e.g. [Feat. Class, x, y])
So it can 'Fit' in few # channels
↳ Don't ReLU output: destroys too much info over low-dim spaces

- Weight Decay: Account for batch selection variance from dominating
↳ Effectively smoothing over batches
- Const is good, cosine better
- Label smooth: Cross-Entropy of SoftMax will rarely have '1' for label
- LS relaxes constraint; for w/ 0.9 s.t fail cases are focuses

$\frac{\partial H}{\partial x} = \frac{\partial F}{\partial x} + I$ } Ensures never too small

Optimization Algos | GD

$$x_{t+1} = x_t - \eta \nabla_x \left[\underbrace{\tfrac{1}{2}\|Ax_t - y\|_2^2}_{A^T(Ax_t - y)}\right]$$

$$= (I - \eta A^T A)x_t + \eta A^T y$$

$$x_{t+1} - x^* \quad || \quad -((A^TA)^{-1}A^Ty)$$
$$= (I - \eta A^TA)^t (x_0 - x^*)$$
$$\|x_{t+1} - x^*\| \le \underbrace{\|I - \eta A^TA\|_2^t}_{\sigma_{max}\{I - \eta A^TA\}^t} \|x_0 - x^*\|_2$$

$$w_t = (I - \eta A^TA)^t w_0 + \sum_{i=0}^{t-1} (I - \eta X^TX)^i X^T y$$

- Learning rate:
  - σ_{max} limits how fast we can converge without oscillations/instability
  - σ_{min} requires larger learning rate
- No osc. if $(1 - 2\eta\sigma_{max}^2) > -1$
  $\Rightarrow \eta < 1/\sigma_{max}^2$
- Fastest rate: $(1 - 2\eta\sigma_{min}^2) = (1 - 2\eta\sigma_{max}^2)$
  $\gamma = \sigma_{max}/\sigma_{min} \Rightarrow \eta^* = 1/\sigma_{min}^2 + \sigma_{max}^2$
  $\therefore$ Rate $= (\gamma^2 - 1)/(\gamma^2 + 1)$

Momentum: We want our direction to not solely depend on singular values, but also on past steps.
- By dampening oscillations, we effectively downscale LR for largest sing. values, enabling higher η

$$\begin{cases} w_{t+1} = w_t - \eta \alpha_{t+1} & \text{Nesterov: } -\eta(1-\beta)\alpha_t \\ \alpha_{t+1} = (1-\beta)\alpha_t + \beta(\nabla \mathcal{L}(w_t)) \end{cases}$$

  ↳ Exponential moving avg. of grad
    ↳ Still represents just $\sigma_{max}/\sigma_{min}$

ADAM: Pretend coords are indep directions by using different η_i each direction.

$$a_{t+1} = (1-\beta)a_t + \beta(\nabla_w \mathcal{L}(w_t))$$
$$V_{t+1} = (1-\beta')V_t + \beta' \times \left[\cdots \tfrac{\partial}{\partial w_i}\mathcal{L}(w_t) \cdots\right]^T$$
$$w_{t+1} = w_t - \eta a_{t+1}/\sqrt{V_{t+1}} + \epsilon$$

- $\beta' \ll \beta$ s.t. variation in normalization doesn't drive dynamics; V decay slowly →
- Size of grads = V; ensures big σ take small steps
- Ideally we'd find $\underbrace{2^{nd}\text{-order deriv.}}_{O(n^2)}$; practically infeasible

Training | Debug

- Train err. of bigger network is worse than smaller / slower
  ↳ Underfitting: grads too small
  - Inits to He/Xavier; non-zero
  - Skip connections
  - Normalizing layers

```
For _ in range(epochs):
    batch_loss = []
    For batch in dataset:
        [...]
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        batch_loss.append(loss.item())
    training_loss.append(np.mean(batch_loss))
```

- Adam introduces bias – not good w/many possible solns
  AdamW – Normalize s.t. start-up is smoother, goes away after many iters

$$\hat{a}_t = a_t / 1 - \beta^t$$
$$\hat{V}_t = V_t / 1 - (\beta')^t$$

GNNs | Generalization of CNNs

- Graph-level: Look at all graph attrs and make statement/pred. about whole
- Node-level: " statement about nodes
  ↳ Duality: Can be defined over edges

Weight Sharing: Must not have dependency on neighbor cnt or order
- Aggregate operations: tend to blur, rather than sharpen
  ↳ Σ / max / min / Π / LS / Variance

$$f_{w_1}\Big(\text{me}, \sum_N \underbrace{s_{w_2}(\text{me}, \text{them})}_{\text{Attention}} \underbrace{g_{w_3}(\text{them})}_{\text{Importance}}\Big)$$

Dropout | Input standardization is to batch+layer norm as data aug. is to dropout

- Fully-connected layers have no spatial awareness (unlike conv. nets) so normalization less effective
- Implementation: Randomly "kill" a unit (set to 0, don't upd weights)
  ↳ Eval: scale weights by $(1-p)$ [if not done when training]
- Ensures wholistic network: no one unit should be solely responsible
  ↳ Encourage redundant activations, i.e. lower-dimension; discourage sparsity (e.g. one large singular value)
- Stochastic Depth Regularization: Drop residual/skip blocks
  ↳ Used in CNNs
- Drop channels: Image already captures some redundancy of pixels
  ↳ Causes jagged grads
- Dropout params: don't depend on fine-grain details of pixels
  ↳ Only for fully-connected layers

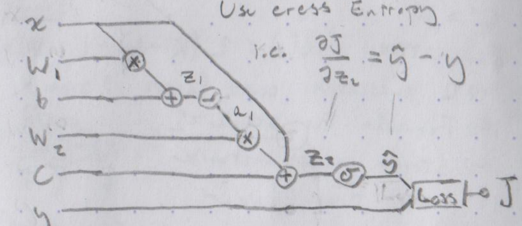
- Global: Inter-layer communication
  ↳ Notion of time/cycle invariance
- Messages either go from node to neighbors, or to all [Larger Receptive Field]
  ↳ Distinguish w/global weights
  - Many global nodes ≡ single " node, multiple channels

Pooling: Task specific
- Content-specific: Alter topology, i.e. add edges till FC
- Cluster: Eg random walk, finds "central" nodes

Vector Calc

$$y = W^{(2)}\text{ReLU}(W^{(1)}x + b) + c$$
$$\nabla_{W_i^{(2)}} y = \text{ReLU}(0, W_i^{(1)}x + b_i)$$
$$\nabla_{b_i} y = \mathbb{1}\{W^{(1)}x + b > 0\} \times W_i^{(2)}$$
$$\nabla_{W_i^{(1)}} y = \mathbb{1}\{W^{(1)}x + b > 0\} \times W_i^{(2)} x$$

Use cross Entropy i.e. $\frac{\partial J}{\partial z_2} = \hat{y} - y$



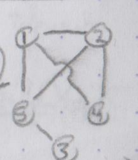
$$\hat{y} = \sigma(W_2 \text{ReLU}(W_1 x + b) + b_2 + x)$$
$$\frac{\partial J}{\partial W_2} = \frac{\partial J}{\partial z_2}\cdot\frac{\partial z_2}{\partial W_2} = (\hat{y} - y)\cdot a_1$$
$$\frac{\partial J}{\partial c} = \frac{\partial J}{\partial z_2}\cdot\frac{\partial z_2}{\partial c} = \hat{y} - y$$
$$\frac{\partial J}{\partial W_1} = \frac{\partial J}{\partial z_2}\cdot\frac{\partial z_2}{\partial a}\cdot\frac{\partial a}{\partial z_1}\cdot\frac{\partial z_1}{\partial W_1} = \left[(\hat{y} - y)W_2 \cdot \frac{\partial a}{\partial z_1}\right] x$$
$$\frac{\partial a_1}{\partial z_1} = \mathbb{1}\{z_1 > 0\}$$
$$\frac{\partial J}{\partial x} = [\,N\,]W_1 + (\hat{y} - y)$$

$$\sigma(x) = 1/1+e^{-x} \quad \Big| \quad \sigma'(x) = -1/(1+e^{-x})^2 \cdot (-e^{-x}) = \frac{1}{1+e^{-x}}\cdot\frac{e^{-x}}{1+e^{-x}} = \sigma(x)\cdot(1 - \sigma(x))$$

$$s_i^{\ell} = s_i^{\ell-1} + W_1\left(\frac{\sum_{j=1}^{n_i} \sigma(W_2 m_{i,j}^{\ell-1})}{n_i}\right)$$

Let $m_{i,j} = s_j$

$$s_2^{\ell} = s_2^{\ell-1} + \frac{W_1}{3}\left(\sigma(W_2 s_1^{\ell-1}) + \sigma(W_2 s_4^{\ell-1}) + \sigma(W_2 s_5^{\ell-1})\right)$$



$$\mathbb{P}(X|Y) = \mathbb{P}(Y|X)\mathbb{P}(X)/\mathbb{P}(Y)$$

$$N_i \sim \mathcal{N}(0, \sigma^2 I) : \sigma^2 I = \mathbb{E}[N_i N_i^T]$$
$$\text{Var}(y) = \mathbb{E}[y^2] - \mathbb{E}[y]^2$$
$$\text{Var}(x) = \underset{\text{STD}}{\sigma(x)}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$