CS2A.

1. No access to $P(X, Y)$? $\Rightarrow$ Hold back $\rightarrow$ test set.
2. Loss not differentiable? $\Rightarrow$ Use a surrogate (train
3. overfit? $\rightarrow$ regularization / Reduce model order.
4. { Expressivity $\rightarrow$ piecewise linear (ReLU).
   | Learnability (e.g. Ridge reg.)

Ridge Regression:

1) OLS problem $\quad \arg\min \|y - Xw\|^2 + \lambda\|w\|^2$.
2) Shift Singular val $\quad (XX^T + \lambda I)^{-1} X^T y$
3) Maximum A Posteriori (MAP)
$$Y = Xw + \sqrt{\lambda} N, \; N \sim N(0, I).$$
4) Fake data. $\quad \hat{X} = \begin{bmatrix} X \\ \sqrt{\lambda} I_d \end{bmatrix} \quad \hat{y} = \begin{bmatrix} y \\ 0_d \end{bmatrix}$
5) Fake feats. $\quad \check{X} = [X \; \sqrt{\lambda} I_n]$.

lr restrictions: Least-squares linear reg.

1) $\eta < \frac{1}{\sigma^2}$.
2) Takes $\log_{|1 - 2\eta\sigma^2|} \varepsilon$ iters
3) For fastest overall: $\eta = \frac{1}{\sigma_1^2 + \sigma_2^2}$

MGD: adjust the magnitude of the param update in each dim $\rightarrow$ optimizer takes bigger & more confid. updates. less osciliation.

Initialization (keep input $\sim N(0,1)$)

1. weights
   1) Xavier init: $\sigma^2 = \frac{1}{d}$ (d: fan in, non-zero weights).
   2) He init. $\sigma^2 = \frac{2}{d}$ (Hope relu "zero" half).
2. bias { 1) treated as $d+1$ weight.
   2) $b = 0$
   3) small random num ($b = 0.01$)

GD.

1. vanilla gradient (GD)
2. SGD with momentum.
   cope with shake. (other methods: low pass filter & Avg)
3. Adaptive method : Adam.
   { diff lr for diff dim
   | lr decay.

$$\vec{a}_{k+1} = (1-\beta)\vec{a}_k + \beta \nabla_w L(w^k)$$
$$\vec{V}_{k+1} = (1-\beta')\vec{V}_k + \beta' \nabla_w \begin{pmatrix} \cdots \\ \left(\frac{\partial L(w)}{\partial w_i}\right)^2 \\ \cdots \end{pmatrix}$$
$\vec{V}_{k+1}$ $\rightarrow$ track the size of gradient
$$w_{k+1}[i] = w_k[i] - \eta \frac{a_{k+1}[i]}{\sqrt{V_{k+i}[i]} + \epsilon}$$
ϵ $\rightarrow$ for stable.

CNN.

1. key ideas { ① locality (neighbors matter)
   ② invariance (translation $\rightarrow$)
   ③ support hierachical structure & multi-resolution understanding
2. output $w \left\lfloor \frac{W + 2P - K}{S} + 1 \right\rfloor$.
   weights: $K \times K \times C \times F$.
3. pooling: no weights compared with stride.
   { Max-pool: invariant feats ✓. retain strong feats
   | highlight the presence of an object.
   | Avg-pool: capture smooth transitions / subtle chang

Dropout.

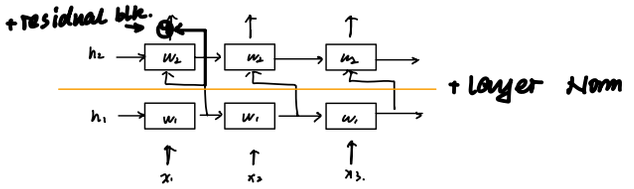
1. improve generalization of network, overfit ↓.
2. dropout rate: cause a discontinuity when 0+, cuz not full rank when dr=0. mult- solu.
3. Dropout & Normalization.
$$L(w) = \|y - pX\check{w}\|^2 + p(1-p)\|\Gamma \check{w}\|^2.$$
4. weights muled by $\frac{1}{1-p}$.

GNN.

1. CNN vs GNN (neighbors distinguishable?)
   1) weight-sharing : GNN neighbors
   2) pooling: GNN cluster and merge.
   3) Normalization: { layer (over channel)
   | batch (sample nodes as a batch)
2. ① A: adjacency mat. D: degree mat.
   $A^{Normalized} = AD^{-1}$.
   $A^{SymNorm} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
   ② $L = D - A$ (Laplacian mat)
$$L_{ij} = \begin{cases} \deg(v_i), & i = j \\ -1, & i \neq j \;\&\; v_i \text{ ad. to } v_j \\ 0, & \text{else.} \end{cases}$$
$$L^{SymNorm} = I - A^{SymNorm} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}.$$
3. Be careful:
   { Which pts are for testing when calc. loss.
   | 100% test acc with misclassified samples?

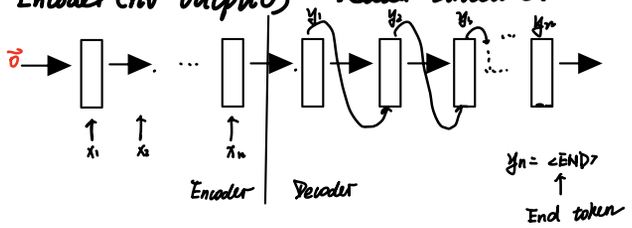
RNN.

1. CNN: Finite Impulse Response Filters. (FIR)
   What if infinite? How to represent? $\Rightarrow$ hidden stat.
   RNN: IIR. $\quad y[t] = a \cdot y[t-1] + b \cdot x[t]$.
2. E.g. Kalman Filter
   { $\vec{x}_t = C\vec{h}_t + \vec{v}_t$ (know $\vec{v}_t$, C).
   | $\vec{h}_{t+1} = A\vec{h}_t + B\vec{x}_t$ (To learn A, B)
   case 1: data $\{\langle \vec{h}_t, \vec{x}_t \rangle\}$ $\Rightarrow$ comp $\vec{h}_t$ and $\hat{h}_0$.
   case 2: data $\{\vec{x}_t\}$ $\Rightarrow$ Comp $\vec{x}_t$ and $\hat{\vec{x}}_t$.
3. Challenges in RNN: gradient explode / vanish.
   1) Saturating activation fns (e.g. sigmoid, tanh)
   2) Gradient clipping.
   3) Normalizations



Seq2Seq model

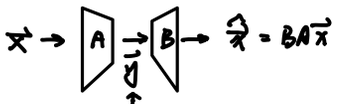
1. Encoder (no output) - Decoder structure.



Note: ① during training, may feed in GT y.

② If len(output$_{true}$) != len(output$_{GT}$): stop early.

2. Autoencoder (expect output → input).



$\vec{x} \to A \to \vec{y} \to B \to \hat{\vec{x}} = BA\vec{x}$

length k representation (bottle neck).

① Vanilla AE.

② Denoise AE. { inputs: $\vec{x_i} + \vec{n}$; expected outputs: $\vec{x_i}$.

Do not add noise to test set.

③ Masked AE. { inputs: $\vec{x_i}$ * mask. ; ... : $\vec{x_i}$

① After pretrain, AE's decoder is useless.

② Pre-trained sometimes worse perf.

③ Autoencoder representation more useful than raw inputs when few labels for downstream.

④ For language problem, loss: cross-entropy.

⑤ With imgs, more effective to mask patch instead of pixels. (Models'll get decent loss by copying neighbor's pixels.

⑥ Can think of denoising & masked AE as vanilla with data augmentation.

Attention

1. queryable softmax pooling: differentiable.

{ Compute "distance": $e_i = \frac{\langle \vec{k_i}, \vec{q} \rangle}{\sqrt{d}}$
Compute "weight": $\alpha_i = \frac{e^{e_i}}{\sum e^{e_j}}$
Compute "output": $\sum \alpha_i \vec{v_i}$.

2. { self attention: key-val pairs from decoder
cross attention: .. encoder.

Self-supervision. (unsupervised learning).

1. PCA: de-mean data.

① data mat → svd. $n\{\overbrace{X}^{d} = U\Sigma V^T$.

② Compute $v_i^T \vec{x}$ to get ith principle feature of a new pt $\vec{x}$.

② Interpreted as self-supervision as AE.

$X_{d\times n} = U\Sigma V^T$.

↳ Project into $\vec{u_1} \cdots \vec{u_k}$ (reduced subspace)

$\min_{W_1, W_2} \| y - W_2 W_1 x \|^2$.

↳: $W_2: d\times k$, $W_1: k\times d$, ensure $\dim \le k$.

Math.

1. $\nabla_x f(x) = \begin{bmatrix} \frac{df(x)}{dx_1} \\ \vdots \\ \frac{df(x)}{dx_n} \end{bmatrix}$ $\nabla_{x^2} f(x) = \begin{bmatrix} \frac{df}{dx_1^2} & \cdots & \frac{df}{dx_1 dx_n} \\ \frac{df}{dx_n dx_1} & \cdots & \frac{df}{dx_n^2} \end{bmatrix}$

2. $\frac{d}{dx} x^T A x = (A + A^T)x$.

3. $\frac{d}{dx} w^T x = w$.

4. $\frac{d}{dA}(x^T A y) = xy^T$.

$\frac{d}{dA}(x^T A^T y) = yx^T$.

Note we're writing derivative in column! (Transpose if in row)