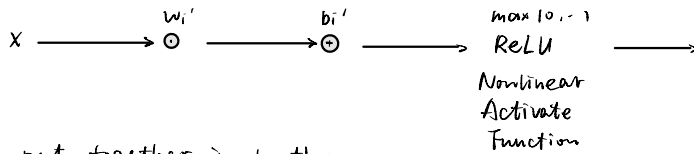


Lecture 1

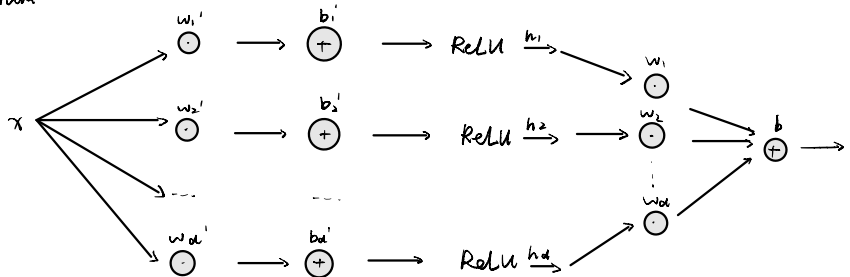
Deep Neural Network



put together in math:

$$b + \sum_{i=1}^d w_i \text{ReLU}(w_i' x + b_i') = b^2 + w^2 \text{ReLU}(w' x + \vec{b}')$$

or diagram



Machine Learning

Supervised

(x, y) data

x input / covariable

y output / label

→ Regression: y real number.

→ Classification: binary
discrete

Unsupervised

x data

no label

→ PCA style

where is the variance?

→ Clustering.

K-means.

...

Deep Learning.

Supervised

Regression.

Classification

Localized Annotation

Unsupervised

Learned Embedding. (like PCA)

Generative models (like density estimation)

1. Typical Supervised Learning Setup

① Training Data: (x_i, y_i)
input, label

② Model: $f_{\theta}(\cdot)$

③ Loss function: $L(y, \hat{y})$

④ Optimizer

Goal: $\hat{y} = f_{\hat{\theta}}(x)$
↑
learned parameter.

2. Empirical Risk Minimization (ERM).

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n L(y_i, f_{\theta}(x_i))$$

↑
Loss function

Key Assumption.

(x, y) are drawn from $P(x, y)$

$$\Rightarrow \theta^* = \underset{\theta}{\operatorname{argmin}} \mathbb{E}(L(y_i, f_{\theta}(x_i)))$$

Challenge 1: Don't know $P(x, y)$

Solution: Hold Out Test Set.

Challenge 2: Loss function doesn't work with optimizers. $\Rightarrow$ Change Loss function

Challenge 3: $\hat{\theta}$ works good at training data.
but poorly at test data

(Overfitting !!)

3. $\rightarrow$ Regularization.

Regularizer

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^n L_{\text{train}}(y_i, f_{\theta}(x_i)) + R(\theta) \right]$$

Ridge Regularizer $\arg\min_w \|Xw - y\|_2^2 + \lambda \|w\|_2^2$

hyperparameter. 训练前预设

→ New Problem: Hyperparameter Search.

- Scale hyperpara. together, couple them.
- Use other people did :)

4. Gradient Descent. 沿梯度反方向寻找最优解

$$L(\theta, f_\theta(x)) \rightarrow L(\theta, \vec{x}_{\text{train}}, \vec{y}_{\text{train}})$$

θ_0 initial condition.

$$\star \vec{\theta}_{t+1} = \vec{\theta}_t - \eta \nabla_{\theta} L(\theta_t, \vec{x}_{\text{train}}, \vec{y}_{\text{train}}).$$

Consider $L(\vec{w}) = \|X\vec{w} - \vec{y}\|_2^2$ 以最小二乘为例

Least Square:

理论最优解

$$\vec{w}^* = (X^T X)^{-1} X^T y.$$

Normal Equation

对 Least Square Loss Function 求导 = 0

$$\nabla_{\vec{w}} L(\vec{w}) = 2X^T (X\vec{w} - \vec{y}).$$

误差. $\vec{w}_{t+1} - \vec{w}_* = (I - 2\eta X^T X) \vec{w}_t + 2\eta X^T \vec{y} - \vec{w}_*.$

$$= (I - 2\eta X^T X) (\vec{w}_t - \vec{w}_*)$$

那么, 误差每次都减小. ⇒ 算法收敛

特征值 < 1

eigenvalue of $(I - 2\eta X^T X) \rightarrow$ symmetric.

$$|1 - 2\eta \lambda_{\max}| < 1 \quad \& \quad |1 - 2\eta \lambda_{\min}| < 1.$$

Fastest convergence is given by: $0 < \eta < \frac{1}{\lambda_{\max}(X^T X)}$

学习率的理论界限.

$$\begin{aligned} J(w) &= \|Xw - y\|^2 \\ &= (Xw - y)^T (Xw - y) \\ &= (Xw)^T - y^T (Xw - y) \\ &= (w^T X^T - y^T) (Xw - y) \\ &= w^T X^T X w - 2w^T X^T y + y^T y \\ \frac{\partial J(w)}{\partial w} &= 2X^T X w - 2X^T y = 0 \\ \Rightarrow X^T X w &= X^T y \\ w &= (X^T X)^{-1} X^T y. \end{aligned}$$

5. Explicit Regularization.

Regularization. $\mathcal{L}(\vec{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$

Ridge Solution $\vec{w}_* = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \vec{y}$
 $= \underbrace{\mathbf{X}^T (\mathbf{X} \mathbf{X}^T + \lambda \mathbf{I})^{-1}}_{\text{Kernel Ridge}} \vec{y}.$

此处运用恒等式:
 $(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T = \mathbf{X}^T (\mathbf{X} \mathbf{X}^T + \lambda \mathbf{I})^{-1}$
 特征. 样本.

6. SVD Simplify Regularization

$$\mathbf{X}\mathbf{w} = \mathbf{y}$$

$$\text{SVD} \Rightarrow \mathbf{U} \Sigma \mathbf{V}^T \mathbf{w} = \mathbf{y} \Rightarrow \Sigma \mathbf{V}^T \mathbf{w} = \underbrace{\mathbf{U}^T \mathbf{y}}_{\vec{y}}$$

we get $\Sigma \tilde{\mathbf{w}} = \tilde{\mathbf{y}}$

$$\sigma_i \tilde{w}_{[i]} = \tilde{y}_{[i]}$$

$$\Rightarrow \tilde{w}_{[i]} = \frac{1}{\sigma_i} \tilde{y}_{[i]}$$

对于 Ridge Solution $\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$

$$\Rightarrow \tilde{w}_{[i]} = \frac{\sigma_i}{\sigma_i^2 + \lambda} \tilde{y}_{[i]}$$

Lecture 2.

此处是复习!!

$$\begin{cases} \vec{w}_* = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \vec{y} = \mathbf{X}^T (\mathbf{X} \mathbf{X}^T + \lambda \mathbf{I})^{-1} \vec{y} \\ \mathbf{X} = \mathbf{U} \Sigma \mathbf{V}^T \end{cases} \rightarrow \text{复习解析解公式.}$$

$$\Rightarrow \vec{w}_* = (\mathbf{V} \Sigma^T \mathbf{U}^T \mathbf{U} \Sigma \mathbf{V}^T + \lambda \mathbf{I})^{-1} \mathbf{V} \Sigma^T \mathbf{U}^T \vec{y}$$

正交矩阵性质

$$= \mathbf{V} (\Sigma^T \Sigma + \lambda \mathbf{I})^{-1} \Sigma^T \mathbf{U}^T \vec{y}$$

$$= \mathbf{V} (\Sigma^T \Sigma + \lambda \mathbf{I})^{-1} \Sigma^T \mathbf{U}^T \vec{y}$$

$$= \mathbf{V} \begin{bmatrix} \sigma_1^2 + \lambda & & \\ & \sigma_2^2 + \lambda & \\ & & \ddots \\ & & & \sigma_n^2 + \lambda \end{bmatrix}^{-1} \Sigma^T \mathbf{U}^T \vec{y}$$

$$\vec{w}_* = \sum_{i=1}^n \tilde{v}_i \boxed{\frac{\sigma_i}{\sigma_i^2 + \lambda}} \tilde{u}_i = \vec{y}$$

if $\lambda \gg \sigma_i$

$$\Rightarrow \frac{\sigma_i}{\sigma_i^2 + \lambda} \rightarrow 0 \text{ as } \lambda \rightarrow \infty.$$

if $\sigma_i \gg \lambda$

$$\frac{\sigma_i}{\sigma_i^2 + \lambda} \rightarrow \frac{1}{\sigma_i}$$

强迫 w 保持较小, 避免 overfitting.

span of data.

$$\text{GD Update: } \vec{w}_{t+1} = \vec{w}_t - 2\eta \underbrace{\lambda^T \lambda \vec{w}_t - \vec{y}}_{\text{residual}}$$

GD Update for ridge:

$$\vec{w}_{t+1} = \underbrace{(1 - 2\eta\lambda)}_{\text{weight decay}} \vec{w}_t - 2\eta \lambda^T (\lambda \vec{w}_t - \vec{y})$$

Implicit Regularization.

$$V^T \vec{w}_{t+1} = V^T \vec{w}_t - 2\eta V^T (V \Sigma^T U^T) (\lambda \vec{w}_t - \vec{y})$$

$$\vec{w}_{t+1} = \vec{w}_t - 2\eta \Sigma^T (U^T U \Sigma V^T \vec{w}_t - U^T \vec{y})$$

$$= \vec{w}_t - 2\eta \Sigma^T (\Sigma \vec{w}_t - U^T \vec{y})$$

$$\vec{w}_{t+1,i} = \vec{w}_{t,i} - 2\eta [\sigma_i^2 \vec{w}_{t,i} - \sigma_i \tilde{y}_i]$$

$$= \vec{w}_{t,i} - 2\eta \sigma_i [\sigma_i \vec{w}_{t,i} - \tilde{y}_i] \rightarrow \text{Early Stopping.}$$

也就是说，在使用 GD 时，会优先更新与数据方差较大（奇异值较大）的方向。

因此，早停法 (Early Stopping) 就成了一种有效的隐式正则化手段。当我们在一个恰当的时机停止训练时，模型已经充分学习了重要的特征 (大奇异值方向)，但还没有来得及学习那些不重要或噪声特征 (小奇异值方向)，从而自然地避免了过拟合，实现了类似岭回归的效果，但没有显式添加正则化项。

Ridge $\rightarrow \lambda$ is large $\rightarrow$ we don't have large components of directions of small singular values in Ridge solution.

GD $\rightarrow$ early stopping $\rightarrow$ also does not move in directions of small singular values.

GD is great for convex function.

SGD - Stochastic Gradient Descent.

适用于 non-convex

→ Add some noise to gradient direction.

In expectation, stml descent.

$$\min_{\theta} f(\theta)$$

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{1}{B} \sum_{i=1}^B \nabla f_i(\theta_t)$$

B: batch size.
"mini-batch"

where $f(\theta) = \frac{1}{n} \sum_{i=1}^n f_i(\theta)$

$$E[\nabla f_i(\theta)] = \frac{1}{n} \cdot \sum_{i=1}^n \nabla f_i(\theta) = \nabla f(\theta)$$

draw i uniformly iid.

每一小步看似随机.
长远来看, 仍然正确

Does SGD Converge?

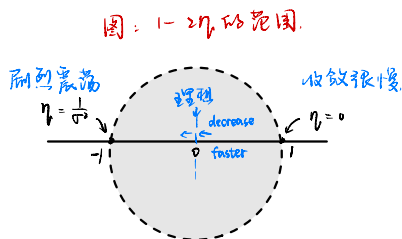
Optimization: Must use a learning rate with decreasing step size.

Real life: Constant stepsize sometimes converge.

Review:

$$\begin{aligned} \vec{w}_{t+1} &= \vec{w}_t - 2\eta \cdot \mathbf{x}^T (\mathbf{x} \vec{w}_t - \vec{y}) \\ &= (\mathbf{I} - 2\eta \mathbf{x}^T \mathbf{x}) \vec{w}_t + 2\eta \mathbf{x}^T \vec{y} \end{aligned}$$

$$\Rightarrow \eta < \frac{1}{\lambda_{\max}}$$



图示: 提高收敛速度 又不至于震荡太大

Averaging / Low Pass Filtering.

e.g. $\frac{1}{t} \sum_{i=1}^t \nabla f(\vec{w}_i) \rightarrow \text{straight average}$

Momentum.

Add memory. $\vec{w}_{k+1} = \vec{w}_k - \eta \cdot \vec{z}_{k+1}$

$$\vec{z}_{k+1} = \beta \vec{z}_k + (1-\beta) \nabla f(\vec{w}_k). \quad \vec{z}_0 = 0. \quad \beta \in [0, 1]$$

Summary — GD & SGD

1. GD.

对整个数据集, 损失函数是所有样本损失的均值.

$$J(\theta) = \frac{1}{N} \sum_{i=1}^N J_i(\theta; x^{(i)}, y^{(i)})$$

GD更新公式.

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t)$$

2. SGD.

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t; x^{(i)}, y^{(i)})$$

Lecture 3.

$$X\vec{w} = \vec{y}, \quad X \in \mathbb{R}^{n \times d} \quad d > n. \quad \text{full row rank.}$$

Let $\vec{w}_*$ = min-norm solution

$$\vec{w}_* = X^T (X X^T)^{-1} \vec{y}.$$

Do SGD, batch size 1 $\vec{w}_0 = 0$.

Error Space $X(\underbrace{\vec{w} - \vec{w}_*}_{\vec{q}}) = \vec{y} - \vec{y} = 0$

$$X\vec{q} = 0$$

with $\vec{q} = \vec{w}_0 - \vec{w}_* = -\vec{w}_*$

$$= U \Sigma V^T \vec{q}$$

$$= U \begin{bmatrix} \Sigma_1 & 0 \end{bmatrix} V^T \vec{q}.$$

with $V^T \vec{q} = \vec{z}$, $U \Sigma_1 = \tilde{X}$

$$= [\tilde{X}, 0] \vec{z}.$$

$$\Rightarrow \vec{z}_0 = V^T \vec{q}_0 = V^T (-\vec{w}_*)$$

$$= - \begin{bmatrix} \Sigma_1^{-1} U^T \vec{y} \\ 0 \end{bmatrix} \quad \left\{ \begin{array}{l} d-n \text{ zero entries} \end{array} \right.$$



$$\tilde{X} \vec{z} = 0.$$

$$X \vec{z} = 0.$$

Now GD:

$$\vec{z}_{t+1} = \vec{z}_t - 2\eta_t \tilde{X}^T \tilde{X} \vec{z}_t.$$

$$\text{Loss} = \|\tilde{X} \vec{z}\|_2^2.$$

$$\tilde{X}^T \tilde{X} = \Sigma_1^T U^T U \Sigma_1 = \Sigma_1^T \Sigma_1$$

$$\nabla \text{Loss} = 2 \tilde{X}^T \tilde{X} \vec{z}$$

For SGD with batch size 1

$$z_{t+1} = z_t - 2\eta_t \tilde{x}_{I_t}^T \tilde{x}_{I_t} z_t.$$

Can we argue that

随机采样.

$$\text{Loss} = \|\tilde{X} \vec{z}\|_2^2 = \vec{z}^T \tilde{X}^T \tilde{X} \vec{z} \rightarrow 0.$$

Method: Lyapunov function (知道证明).

Connect $\mathcal{L}(\vec{z}_{t+1})$ to $\mathcal{L}(\vec{z}_t)$.

$$\mathcal{L}(\vec{z}_{t+1}) = \mathcal{L}(\vec{z}_t + (\vec{z}_{t+1} - \vec{z}_t))$$

$$= \mathcal{L}(\vec{z}_t) + \boxed{A} + B, \quad \text{损失项.}$$

with $A = 2\vec{z}_t^T \tilde{\mathbf{X}}^T \tilde{\mathbf{X}} (\vec{z}_{t+1} - \vec{z}_t) \rightarrow \text{Linear 发挥作用}$

$B = (\vec{z}_{t+1} - \vec{z}_t)^T \tilde{\mathbf{X}}^T \tilde{\mathbf{X}} (\vec{z}_{t+1} - \vec{z}_t) \rightarrow \text{Quadratic 别影响.}$

W.T. 我们需证明. 损失在期望下是下降的.

$$\Rightarrow \mathbb{E}[A | \vec{z}_t] \leq -c_1 \eta \mathcal{L}(\vec{z}_t)$$

Proof: $= 2\vec{z}_t^T \tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \cdot \mathbb{E}[\vec{z}_{t+1} - \vec{z}_t | \vec{z}_t]$

$$= 2\vec{z}_t^T \tilde{\mathbf{X}}^T \tilde{\mathbf{X}} (-2\eta) \cdot \mathbb{E}[\tilde{\mathbf{X}}_{It} \tilde{\mathbf{X}}_{It}^T] \cdot \vec{z}_t$$

Recall. $\mathbb{E}[\hat{\mathbf{X}}_{It} \hat{\mathbf{X}}_{It}^T] = \frac{1}{n} \sum_{i=1}^n \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T = \frac{1}{n} \tilde{\mathbf{X}}^T \tilde{\mathbf{X}}, \quad \tilde{\mathbf{X}} = \begin{bmatrix} \tilde{\mathbf{x}}_1^T \\ \vdots \\ \tilde{\mathbf{x}}_n^T \end{bmatrix}$

$$= -\frac{4\eta}{n} \cdot \vec{z}_t^T \tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \boxed{\tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \cdot \vec{z}_t} \quad \text{min scaling } \sigma_{\min}^2$$

$$\leq -\frac{4\eta}{n} \cdot \sigma_{\min}^2 \cdot \vec{z}_t^T \tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \cdot \vec{z}_t$$

$$= -\frac{4\eta}{n} \cdot \sigma_{\min}^2 \mathcal{L}(\vec{z}_t)$$

Similarly. $\mathbb{E}[B | \vec{z}_t] \leq c_2 \eta \mathcal{L}(\vec{z}_t)$

Adam

关注两个问题：
{ 更快更稳地前进
如何兼顾非最显著特征。

In Regular GD: $\vec{w}_{t+1} = \vec{w}_t - \eta \nabla_{\vec{w}} L(\vec{w}) \mid \vec{w} = w_t$

Instead. $\vec{m}_t \leftarrow \nabla_{\vec{w}} L(\vec{w})$ gradient
 $\vec{v}_t \leftarrow (\nabla_{\vec{w}} L(\vec{w}))^2$ squared gradient.

$$\vec{w}_{t+1} = \vec{w}_t - \eta \frac{\vec{m}_{t+1}}{\sqrt{\vec{v}_{t+1}} + \epsilon} \rightarrow \text{避免分母为0.}$$

$$\frac{\vec{m}_{t+1}}{\sqrt{\vec{v}_{t+1}}} = \text{sign}(\nabla_{\vec{w}} L(\vec{w})). \quad \text{Sign SGD. 忽略大小, 关注方向}$$

存在问题: 接近 optimum. 振荡. $\rightarrow$ momentum.

$\rightarrow$ 一般接近!

Adam.

$$\begin{cases} \vec{m}_{t+1} = \beta \vec{m}_t + (1-\beta) \nabla_{\vec{w}} L(\vec{w}_t) \\ \vec{v}_{t+1} = \gamma \vec{v}_t + (1-\gamma) (\nabla_{\vec{w}} L(\vec{w}_t))^2 \end{cases}$$

"Bias Correction"

$$\tilde{\vec{m}}_{t+1} = \frac{\vec{m}_{t+1}}{1-\beta^{t+1}}$$

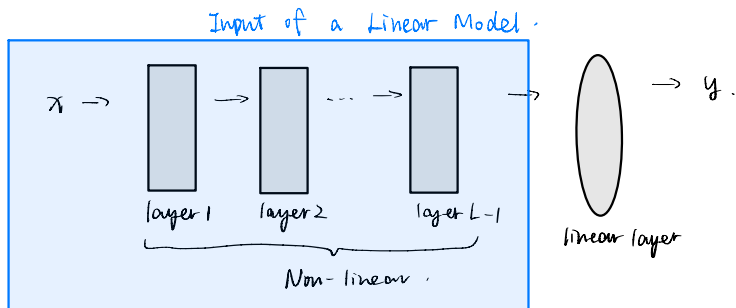
$$\tilde{\vec{v}}_{t+1} = \frac{\vec{v}_{t+1}}{1-\gamma^{t+1}}$$

$$\vec{w}_{t+1} = \vec{w}_t - \eta \frac{\tilde{\vec{m}}_{t+1}}{\sqrt{\tilde{\vec{v}}_{t+1}} + \epsilon}$$

$\rightarrow$ Adam!

Lecture 4

Linearized Model



$$f(x, \theta + \Delta\theta) = f(x, \theta) + \langle \nabla_{\theta} f, \Delta\theta \rangle + \dots \quad \leftarrow \text{Taylor Expansion}$$

Lazy training assumption: Weights are moving by very small amount -

Recall $\nabla_{\theta} f = \left(\frac{\partial f}{\partial \theta} \right)^T$

$\langle \nabla_{\theta} f, \Delta\theta \rangle$

$\frac{\partial f}{\partial \theta}$ (entries are features!) $\Delta\theta$

Neural Tangent Kernel (知道即可)

Revisit Adam (Sign SGD)

Loss Function

$$f(\vec{x}, \vec{\theta}_i + \Delta\vec{\theta}_i) = f(\vec{x}, \vec{\theta}_i) + \left. \frac{\partial f}{\partial \theta} \right|_{\theta=\theta_i} \Delta\vec{\theta}_i$$

矛盾

- Want large steps to converge fast.
- Want small steps to make sure approximation still holds.

Idea.

Bound the size of step:

$$\max_i \|\Delta\theta_{ij}\| \leq \eta$$

无差范数 即所有元素中绝对值最大的绝对值。

Want.

此式不要 $\Delta\theta$ 最长问

$$\argmin_{\|\Delta\theta\| \leq \eta} \frac{1}{n} \sum_{i=1}^n \left[f(\vec{x}_i, \vec{\theta}_i) + \left. \nabla_{\theta} f \right|_{\theta=\theta_i}^{\vec{x}=\vec{x}_i} \Delta\vec{\theta}_i \right]$$

$$\Leftrightarrow \argmin_{\|\Delta\theta\| \leq \eta} \left\langle \frac{1}{n} \sum_{i=1}^n \nabla_{\theta} f \Big|_{\theta=\theta_i}^{\vec{x}=\vec{x}_i}, \Delta\vec{\theta} \right\rangle$$

$$\Leftrightarrow \argmin \sum_j \left[\underbrace{\left(\frac{1}{n} \sum_{i=1}^n \nabla_{\theta} f \Big|_{\theta=\theta_i}^{\vec{x}=\vec{x}_i} \right)_{[j]}}_{\nabla_{\theta} f_{\text{batch}}} \cdot \Delta\vec{\theta}_{[j]} \right] \quad \leftarrow \text{element by element.}$$

So, we can do optimization individually.

$$\Rightarrow \argmin_{\|\Delta\theta_{[j]}\| \leq \eta} \nabla_{\theta} f_{\text{batch}} \cdot \Delta\theta_{[j]}$$

每个方向上吗?

$$\begin{aligned} & \Delta\theta_i = +\eta & \text{if } \nabla_{\theta} f_{\text{batch}} < 0 \\ & \Delta\theta_i = -\eta & \text{if } \nabla_{\theta} f_{\text{batch}} > 0. \end{aligned} \Rightarrow \Delta\vec{\theta}^* = -\eta \cdot \text{sign}(\nabla_{\theta} f_{\text{batch}})$$

Similarly, we can use $\|\Delta \vec{\theta}\|_2 \leq \eta$

$$\vec{v}_h^-: \underset{\|\Delta \vec{\theta}\|_2 \leq \eta}{\operatorname{argmin}} \langle \nabla_{\vec{\theta}} \mathcal{L}(\vec{\theta}), \Delta \vec{\theta} \rangle$$

$$\text{Recall: } \vec{x}^T \vec{y} = \|\vec{x}\|_2 \|\vec{y}\|_2 \cos \theta.$$

To minimize, the angle $\Rightarrow \cos \phi = -1$.

$$\Rightarrow \Delta \vec{\theta} = -\eta \frac{\nabla_{\vec{\theta}} \mathcal{L}(\vec{\theta})}{\|\nabla_{\vec{\theta}} \mathcal{L}(\vec{\theta})\|_2}$$

一个有约束的问题
等价于

一个无约束但有正则项的问题

$\vec{v}_h^-$: Take regularization / Lagrange

$$\underset{\vec{\theta}}{\operatorname{argmin}} \langle \nabla_{\vec{\theta}} \mathcal{L}(\vec{\theta}), \Delta \vec{\theta} \rangle + \lambda \|\Delta \vec{\theta}\|_2^2$$

Alternatively, take the regularization / Lagrange multiplier perspective.

$$\underset{\substack{\vec{\theta} \\ \text{unconstrained}}}{\operatorname{argmin}} \underbrace{\langle \nabla_{\vec{\theta}} \mathcal{L}(\vec{\theta}), \Delta \vec{\theta} \rangle + \lambda \|\Delta \vec{\theta}\|_2^2}_{g(\Delta \vec{\theta}), \text{convex.}}$$

→ Sweeps out the same family of solutions as constrained version.

$$\nabla g(\Delta \vec{\theta}) = \nabla_{\vec{\theta}} \mathcal{L}(\vec{\theta}) + 2\lambda \Delta \vec{\theta} \rightarrow \text{Set to zero.}$$

$$\text{Minimize @ } \Delta \vec{\theta}^* = \underbrace{-\frac{1}{2\lambda}}_{\text{stepsize}} \underbrace{\nabla_{\vec{\theta}} \mathcal{L}(\vec{\theta})}_{\text{gradient descent}}$$

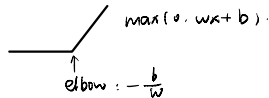
Other Things to Consider

Optimizer isn't all for ML!

① Standardize Data $\rightarrow$ be in a reasonable range !!

Consider ReLU

$$x \rightarrow \odot \rightarrow \oplus \rightarrow \boxed{\text{ReLU}} \rightarrow$$



Say we initialize

$$b \sim N(0, 1) \quad w \sim N(0, 1)$$

then $-\frac{b}{w} \sim \text{Cauchy Distribution}$. $\rightarrow \text{pdf: } \frac{1}{\pi} \left[\frac{1}{x^2 + 1} \right]$

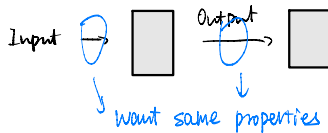
Property: mostly around 0

② Initialization

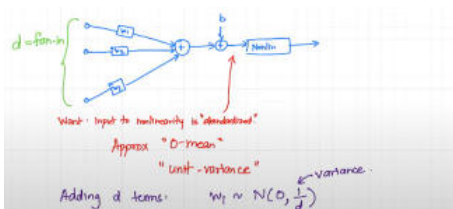
$\rightarrow$ Set everything to 0?

Everything zero. weights never move.

$\rightarrow$ What do we do?



Xavier initialization.



$$\begin{aligned} \text{Var}(\sum w_i h_i) &= E[(\sum w_i h_i)^2] - \underbrace{(E(\sum w_i h_i))^2}_0 \\ &= \sum_i E(w_i^2) E(h_i^2) \end{aligned}$$

He-Initialization. (Xavier works bad for ReLU)

Output of a ReLU is zero half the time.

$$w_i \sim N(0, \frac{1}{d/2}) = N(0, \frac{2}{d})$$

Lecture 5

Recall. Lazy Training Assumption.

GD step is ∇L .

Apply in Matrices.

$$\argmin_{\|\Delta W\| \leq \eta} \langle \nabla_W L(W), \Delta W \rangle$$

matrixmatrix

↓
which norm?

Consider $\|\Delta W\|_2 \leq \eta$.

$$\argmin_{\|\Delta W\|_2 \leq \eta} \langle \nabla_W L(W), \Delta W \rangle.$$

$$\text{Let } \nabla_W L(W) = U \Sigma V^T.$$

$$\Rightarrow \Delta W^* = -\eta U V^T$$

Recall

$$\langle A, B \rangle_F = \text{trace}(A^T B)$$

Recall

$\|A\|_2 = \sigma_{\max} \rightarrow$ largest singular value.

$\max_{\|x\|_2=1} \|Ax\|_2 \rightarrow$ what's the max scaling for a unit vector.

This is Shampoo Optimizer

如何证明?

$$\argmin \langle \nabla_W L(W), \Delta W \rangle$$

$$= \argmin \text{trace}(\nabla_W L(W)^T \Delta W)$$

$$= - \argmax \text{trace}(\nabla_W L(W)^T \Delta W)$$

简化为.

$$\max_{\|B\|_2 \leq 1} \text{trace}(A^T B)$$

$$\begin{aligned}
 &= \max_{\|B\|_2 \leq 1} \text{trace}(V \Sigma^T U^T B) \\
 &= \max_{\|B\|_2 \leq 1} \text{trace}\left(\sum_i \sigma_i \vec{u}_i \vec{u}_i^T B\right) \\
 &= \max_{\|B\|_2 \leq 1} \sum_i \text{trace}(\sigma_i \vec{u}_i \vec{u}_i^T B) \\
 &= \max_{\|B\|_2 \leq 1} \sum_i \text{trace}(\sigma_i \underbrace{\vec{u}_i^T B \vec{u}_i}_{\text{Scalar}}) \\
 &= \max_{\|B\|_2 \leq 1} \sum_i \sigma_i \underbrace{\vec{u}_i^T B \vec{u}_i}_{\text{Scalar}} \\
 &\leq \sum_i \sigma_i
 \end{aligned}$$

$\|B \vec{u}_i\|_2 \leq \|\vec{u}_i\|_2 \cdot \|B\|_2 \leq 1$
 $\|\vec{u}_i^T B \vec{u}_i\|_2 \leq 1$. Cauchy-Sch.

→ To attain bound $B = UV^T$

$$\text{trace}(ABC) = \text{Trace}(BCA)$$

How to choose a good norm?

RMS

$$\|\vec{x}\|_{\text{RMS}} = \frac{1}{\sqrt{\dim}} \|\vec{x}\|_2 = \frac{1}{\sqrt{\dim}} \sqrt{\sum_i x_i^2}$$

RMS → RMS Norm

Recall: induced matrix norm 诱导矩阵范数

$$\|A\|_{\alpha \rightarrow \beta} = \max_{\|x\|_\alpha = 1} \|Ax\|_\beta$$

Motivated by Xavier Initialization.

Let $A \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ 将一个 d_{in} 维度的向量映射到 d_{out} 维。

$$\|A\|_{\text{RMS} \rightarrow \text{RMS}} = \max_{\|x\|_{\text{RMS}} = 1} \|Ax\|_{\text{RMS}}$$

Since $\|v\|_{\text{RMS}} = \frac{\|v\|_2}{\sqrt{\dim(v)}}$

$$= \max_{\|x\|_{\text{RMS}} = 1} \frac{1}{\sqrt{d_{\text{out}}}} \|Ax\|_2$$

$$\geq \sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}} \max_{\|x\|_2 = 1} \|Ax\|_2$$

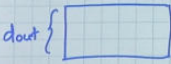
$$= \sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}} \|A\|_2$$

Lecture 6

Recall.

$$\|A\|_{\alpha \rightarrow \beta} = \max_{\|\vec{x}\|_{\alpha}=1} \|A\vec{x}\|_{\beta}.$$

RMS $\rightarrow$ RMS norm $\rightarrow$ Motivated by Xavier initialization.

$$\begin{aligned} \|A\|_{\text{RMS} \rightarrow \text{RMS}} &= \max_{\|\vec{z}\|_{\text{RMS}}=1} \|A\vec{z}\|_{\text{RMS}} \\ A \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}} & \\ &= \max_{\|\vec{z}\|_2 = \sqrt{d_{\text{in}}}} \frac{1}{\sqrt{d_{\text{out}}}} \|A\vec{z}\|_2. \\ &= \sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}} \left(\max_{\|\vec{z}\|_2=1} \|A\vec{z}\|_2 \right) \quad \text{Spectral norm } \|A\|_2! \\ &= \sqrt{\frac{d_{\text{in}}}{d_{\text{out}}}} \|A\|_2 \end{aligned}$$


Can we use in optimizer?

$$\|\Delta W\|_{\text{RMS}^*} \leq \delta$$

$$\Rightarrow \sqrt{\frac{d_1}{d_2}} \|\Delta W\|_2 \leq \delta$$

$$\Rightarrow \|\Delta W\|_2 \leq \delta \sqrt{\frac{d_2}{d_1}}$$

Leads to Mnou !

Different Approach.

We have : $\left\{ \begin{array}{l} \text{Adam} \quad w_{t+1} = w_t - \eta \cdot \text{sign}(\nabla w L(w)) \\ \text{RMS} \quad \text{--- be intelligent in choosing stepsize.} \end{array} \right.$

$$\text{Want : } \|\Delta w\|_{\text{RMS} \rightarrow \text{RMS}} \leq \gamma$$

Consider mini-batch size 1.

$\Rightarrow \nabla w L$ gradient is low-rank. specifically rank 1

证明: why rank 1?

$\Rightarrow \nabla w L = \sigma \vec{u} \vec{v}^T \rightarrow \text{Shampoo?}$

Its spectral norm : $\|\nabla w L\|_2 = \sigma$.

Frobenius norm : $\|\nabla w L\|_F = \sigma$

Spectral norm: 最大奇异值.
 $\sigma_{\max}$

Frobenius norm: $\sqrt{\sum \sigma_i^2}$

$\Rightarrow \|\dots\|_2 = \|\dots\|_F$
for rank 1 !!

Consider $\text{sgn}(\nabla w L) = S$ (elementwise)

What is rank S ?

$$\begin{aligned} S &= \text{sgn}(\sigma \vec{u} \vec{v}^T) = \text{sgn}(\vec{u} \vec{v}^T) \\ &= \text{sgn}(\vec{u}) \cdot \text{sgn}(\vec{v}^T) \end{aligned}$$

$$\Rightarrow \text{Rank } S = 1$$

$$w_{t+1} = w_t - \underbrace{\eta \cdot S}_{\Delta w}$$

Recall. we want Δw to be small.

$$\|\Delta w\|_{\text{RMS} \rightarrow \text{RMS}} \leq \gamma$$

$$\eta \|S\|_{\text{RMS} \rightarrow \text{RMS}} \leq \gamma \quad \Rightarrow \quad \eta \sqrt{\frac{dw}{d\eta}} \cdot \|S\|_2 \leq \gamma$$

What is $\|S\|_2$?

$S = \text{rank } 1$, so spectral norm = frobenius norm 均为1!!!
假设所有元素都是 $\pm 1 \Rightarrow \text{Frobenius norm}^2 = \|S\|_F^2$ 是所有元素的平方和 $= d_{in} \times d_{out} \times 1$
 $\|S\|_F^2 = d_{in} \times d_{out} \Rightarrow \|S\|_2 = \sqrt{d_{in} \times d_{out}} = d_{in} \times d_{out}$

$$\Rightarrow \eta \sqrt{\frac{d_{in}}{d_{out}}} \cdot \sqrt{d_{in} \times d_{out}} \leq \gamma$$

$$\Rightarrow \eta \leq \frac{1}{d_{in}} \gamma \Rightarrow \text{学习率可以随层扩展.}$$

核心理论:

By Xavier and He Initialization.

W (权重矩阵) 的方差应与 d_{in} 成反比.
 $\Rightarrow$ 保证前向传播方差稳定.

Then, we confirm

为保证 γ 约束, η 与 d_{in} 成反比.

In reality, as batch size increases, Frobenius norm
will only be the upper bound of spectral norm
所以我们的约束是相对保守的

Parameterization

$$\text{Consider } F_n(c) \doteq \mathbb{E}_{x_1, \dots, x_n} f(c(x_1 + x_2 + \dots + x_n))$$

Reparameterize $c = \alpha / \sqrt{n}$.

$$G_n(\alpha) \doteq F_n\left(\frac{\alpha}{\sqrt{n}}\right) \rightarrow \mathbb{E}[f(N(0, \alpha^2))]$$

as $n \rightarrow \infty$ $\leftarrow$ by CLT.

$\alpha_n^* = \underset{\alpha}{\operatorname{argmin}} G_n(\alpha) \Rightarrow$ Then $\alpha_n^* \approx \alpha^*$ for large enough n, N
but instead α_n^* can not transfer

Review For L1 - L6

- ZRM.
- Regularization.
- Algorithm.
- Practice.

目标: 找到 θ^* , 最小化损失.

ZRM: 在训练集上学习 $\hat{\theta}$. $\rightarrow$ 平均损失最小.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum L(y_i, f_{\theta}(x_i))$$

$\Rightarrow$ 可能 Overfitting.

解决方法: Regularization.

Regularizer.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[\frac{1}{n} \sum L_{\text{train}}(y_i, f_{\theta}(x_i)) + \underbrace{R(\theta)} \right].$$

① Explicit Regularization.

e.g. Ridge Regularization.

权重 $\left\{ \begin{array}{l} \text{拟合 train data} \\ \text{保持模型简单} \end{array} \right.$

$$\underset{w}{\operatorname{argmin}} \|Xw - y\|_2^2 + \underbrace{\lambda \|w\|_2^2}$$

解析解 (求导 = 0)

$$w_{\lambda} = (X^T X + \lambda I)^{-1} X^T y.$$

Why it work?

$$\text{SVD} \rightarrow \tilde{w}[\lambda] = \frac{\sigma_i}{\sigma_i^2 + \lambda} \tilde{y}[\lambda]$$

$\left\{ \begin{array}{l} \sigma_i \gg \lambda, \frac{\sigma_i}{\sigma_i^2 + \lambda} \rightarrow \frac{1}{\sigma_i} \\ \text{方差大处解不受影响} \\ \sigma_i \ll \lambda, \frac{\sigma_i}{\sigma_i^2 + \lambda} \rightarrow 0 \\ \text{方差小 (可能为噪声)} \rightarrow 0 \end{array} \right.$

② Implicit Regularization.

算法本身也有正则化效果.

GD update for ridge.

$$W_{t+1} = (1 - 2\eta\lambda) W_t - 2\eta (X^T (X W_t - y))$$

→ Early Stopping.

Algorithm.

① GD.

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} J(\theta)$$

② SGD.

$$\theta_{t+1} = \theta_t - \eta \frac{1}{B} \sum_{k=1}^{i+B-1} \nabla J_k(\theta_k)$$

Lecture 7

Recall : Want

$$\begin{aligned} (1) \quad & \|\vec{h}_t\|_{\text{RMS}} = \Theta(1) \\ (2) \quad & \|\Delta \vec{h}_t\|_{\text{RMS}} = \Theta(1) \end{aligned}$$

How can we achieve these ?

$$\vec{h}_t = W_t \vec{h}_{t-1}$$

$\Rightarrow$ if $\vec{h}_{t-1}$ aligns with top singular value of W_t .

$$\|\vec{h}_t\|_2 = \|W_t\|_2 \cdot \|\vec{h}_{t-1}\|_2$$

$$\Rightarrow \Theta(\sqrt{d_{\text{out}}}) = \|W_t\|_2 \cdot \Theta(\sqrt{d_{\text{in}}})$$

$$\Rightarrow \|W_t\|_2 = \Theta(\sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}}) \quad , \quad \text{so } \|W_t\|_{\text{RMS} \rightarrow \text{RMS}} = \Theta(1)$$

!!

if $\|W_t\|_{\text{RMS} \rightarrow \text{RMS}} = \Theta(1)$ $\|\vec{h}_{t-1}\|_{\text{RMS}} = \Theta(1)$

then $\|\vec{h}_t\|_{\text{RMS}} = \Theta(1)$

$$\vec{h}_t = W_t \cdot \vec{h}_{t-1}$$

$$\Delta \vec{h}_t = \Delta W_t \cdot \vec{h}_{t-1} + W_t \cdot \Delta \vec{h}_{t-1}$$

$$\|\Delta \vec{h}_t\|_2 = \|\Delta W_t \vec{h}_{t-1} + W_t \cdot \Delta \vec{h}_{t-1}\|_2$$

$$\leq \|\Delta W_t\|_2 \|\vec{h}_{t-1}\|_2 + \|W_t\|_2 \cdot \|\Delta \vec{h}_{t-1}\|_2$$

$$= \boxed{?} \cdot \Theta(\sqrt{d_{\text{in}}}) + \Theta(\sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}}) \cdot \Theta(\sqrt{d_{\text{in}}})$$

so here
should be

$\Theta(\sqrt{\frac{d_{\text{out}}}{d_{\text{in}}}}) ? \Rightarrow \|\Delta W_t\|_{\text{RMS}} = 1$

Conditions : (1) $\|W_t\|_{\text{RMS} \rightarrow \text{RMS}} = \Theta(1)$

(2) $\|\Delta W_t\|_{\text{RMS} \rightarrow \text{RMS}} = \Theta(1)$

Muon Optimizer

补笔记

Shampoo:

$$L_t = L_{t-1} + G_t G_t^T$$

$$R_t = R_{t-1} + G_t^T G_t$$

$$\nabla_w \mathcal{L}(w) = G_t$$

$$W_{t+1} = W_t - \eta L_t^{-1/4} G_t R_t^{-1/4}$$

Shampoo without accumulation:

$$W_{t+1} = W_t - \eta (G_t G_t^T)^{-1/4} G_t (G_t^T G_t)^{-1/4}$$

$$\begin{cases} G_t G_t^T = U \Sigma V^T V \Sigma^T U^T = U \Sigma \Sigma^T U^T \\ G_t^T G_t = V \Sigma^T U^T U \Sigma V^T = V \Sigma^T \Sigma V^T \end{cases}$$

$$G_t = \text{dmat} \begin{matrix} \text{dim} \\ \boxed{} \end{matrix}$$

$$\rightarrow \Sigma = \begin{bmatrix} \Sigma_r & 0 \end{bmatrix}$$

Assume full row rank

$$\Rightarrow \Sigma \Sigma^T = \text{dmat} \begin{bmatrix} \Sigma_r & 0 \end{bmatrix} \begin{matrix} \text{dmat} \\ \Sigma_r \\ 0 \end{matrix} = \Sigma_r \Sigma_r^T = \begin{bmatrix} \sigma_1^2 & & \\ & \sigma_2^2 & \\ & & \ddots \end{bmatrix}$$

$$(G_t G_t^T)^{-1/4} = U \begin{bmatrix} \sigma_1^{-1/2} & & \\ & \ddots & \\ & & \ddots \end{bmatrix} U^T$$

$$(G_t^T G_t)^{-1/4} = V \begin{bmatrix} \sigma_1^{-1/2} & & \\ & \ddots & \\ & & \ddots \end{bmatrix} V^T$$

$$(G_t G_t^T)^{-1/4} G_t (G_t^T G_t)^{-1/4} = U \boxed{} U^T U \Sigma V^T V \boxed{} V^T$$

$$= U_r V_r^T \rightarrow \text{semi-orthogonal}$$

$$(A A^T = I \text{ or } A^T A = I)$$

What if we had the RMS-RMS norm instead?

$$W \in \mathbb{R}^{\text{dout} \times \text{din}}$$

$$\underset{\|\Delta W\|_{\text{RMS-RMS}} \leq \eta}{\text{argmin}} \quad \langle \nabla_W \mathcal{L}(W), \Delta W \rangle$$

$$= \underset{\|\Delta W\|_2 \leq \sqrt{\frac{\text{dout}}{\text{din}}} \cdot \eta}{\text{argmin}} \quad \langle \nabla_W \mathcal{L}(W), \Delta W \rangle$$

$$\Rightarrow \Delta W^* = -\eta \cdot \sqrt{\frac{\text{dout}}{\text{din}}} U_V V_r^T$$

Why might semi-orthogonal matrices be a good idea?

→ Condition number $U_V V_r^T = 1$. Uniform step in all directions.

→ Not dominated by largest singular values.

BUT! Computing UV^T is expensive! → Needs SVD!

Muon solves this issue through two observations

(1) Getting the direction approximately correct is good enough.

(2) Newton-Schulz iterations.

$$U \Sigma V^T \rightarrow U V^T.$$

i.e. replace all the singular values by 1.

How?

Want $f(U \Sigma V^T) \approx UV^T$ (so maybe sth. converge to UV^T),

Newton-Schulz.

① Odd polynomials commute with the SVD.

$$p(x) = a_0 x + a_1 x x^T x + a_2 (x x^T)^2 x + \dots + a_n (x x^T)^n x.$$

$$\text{e.g. } p(x) = \frac{3}{2} x - \frac{1}{2} x x^T x.$$

$$p(U \Sigma V^T) = \frac{3}{2} U \Sigma V^T - \frac{1}{2} (U \Sigma V^T)(V \Sigma^T U^T) \cdot U \Sigma V^T$$

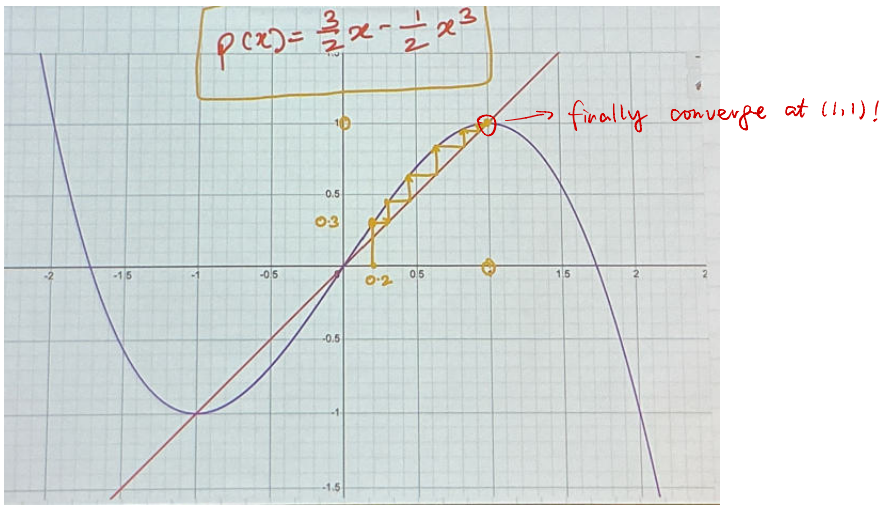
$$= \frac{3}{2} U \Sigma V^T - \frac{1}{2} U \Sigma \Sigma^T \Sigma V^T$$

$$= U \left[\frac{3}{2} \Sigma - \frac{1}{2} \Sigma \Sigma^T \Sigma \right] V^T$$

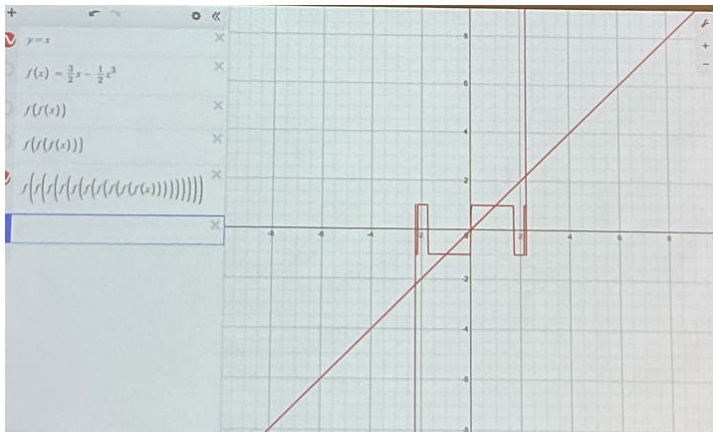
$$= U p(\Sigma) V^T$$

So $p(x) = U p(\Sigma) V^T \Rightarrow$ can apply to a matrix without changing singular value

$\Rightarrow$ Can we find $p(x)$ such that $p(x) \rightarrow 1$



Observe : $p(p(p(\dots p(x))\dots)) \rightarrow 1$



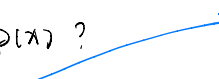
$\rightarrow$ Need to have singular values $\in (0,1]$ before applying!
 $\rightarrow$ How? Normalize by the Frobenius norm : $\frac{G}{\|G\|_F}$

Muon (Momentum Orthogonalized by Newton Schulz)

$$B_t = \mu \cdot B_{t-1} + \nabla_w \mathcal{L}(w)$$

$$O_t = \text{NewtonSchulz}(B_t)$$

$$W_t = W_{t-1} - \eta \cdot W_t$$

How to choose $p(x)$?  决定起始的斜率、截距等项。

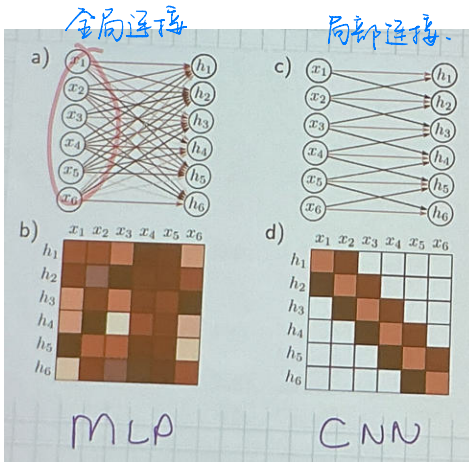
$$p(x) = ax + bx^2 + \dots$$

Higher Order = faster but expensive.

Lecture 8.

order .

MLP $\rightarrow$ CNN $\rightarrow$ Graph NN $\rightarrow$ RNN $\rightarrow$ Transformer



CNN is inspired by CV problem - classification

Key Idea (Expressivity).

- 1) Respect locality. $\rightarrow$ Convolutional structure with small filters.
- 2) Respect certain invariances / equivariance / symmetries.

$\rightarrow$ weight sharing !

$\rightarrow$ data augmentation, training.

- 3) Hierarchical Structures (Parts makes wholes)

自顶向下结构

$\rightarrow$ depth

$\rightarrow$ multiresolution / filterbank.

Key Idea (Getting it to work):

- 1) Normalization layers
- 2) Dropout
- 3) Residual / Skip Connections

Recall: Convolution (LTI System)

$$y(t) = \int_{-\infty}^{\infty} x(\tau) h(t-\tau) d\tau = \int_{-\infty}^{\infty} h(\tau) x(t-\tau) d\tau$$

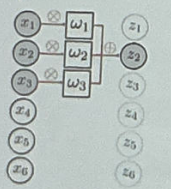
其中 $x(t)$: 输入信号

$h(t)$: 系统响应。
 $\Rightarrow$ 输出取决于当前输入 + 所有过去输入。
且满足交换率。

Recall 1D convolution (LTI systems... e.g. from EECS 170, 123)

$$y(t) = \int_{-\infty}^{\infty} x(\tau) h(t-\tau) d\tau = \int_{-\infty}^{\infty} h(\tau) x(t-\tau) d\tau$$
$$y[n] = \sum_{\tau=-\infty}^{\infty} x[n-\tau] h[\tau] = \sum_{\tau=-\infty}^{\infty} h[\tau] x[n-\tau] \xRightarrow{\text{FIR}} \sum_{\tau=-k}^{+k} h[\tau] x[n-\tau]$$

In Deep Learning, we don't 'flip'

a) 

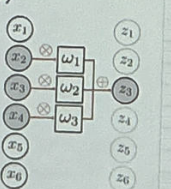
b) 

Figure 10.7 in Prince.

Notice: No Flip
Can also have a bias

差异在于: 不需翻转核。

Weight not share
across channels

If we want more channels at output of the layers.
can repeat with fresh weights & biases for each output channel.

怎么确保不同 channel 学到不同特征?

1. Initialize differently.
2. Backpropagation 会自动调整 (show later).

Special Case: 1×1 conv with C_{in} input channels and C_{out} output channels

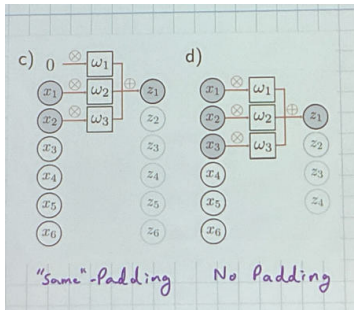
Input is a vector C_{in}
(from 1 pixel)

Output is a vector C_{out} .
(for this pixel)

激活函数, 如 ReLU.

$$\vec{y} = \text{act} \{ W \vec{x} + \vec{b} \}.$$

Padding



Other :

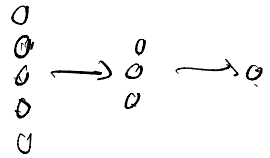
wrap-around.

average.

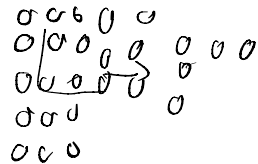
copy.

mirror.

但 0-padding 最常见.



}



结论: 对于一个只使用 $K \times K$ 卷积核的网络,

第 L 层的感受野为 $R_L = R_{L-1} + (K-1)$.

每层增加 $K-1$.

从头开始算, 则为 $(K-1)L + 1$.

Lecture 9.

Today: Convolutional Neural Network.

$k \times k$ cov with C_{in} - channel input and 1 output

How many weights? $= k^2 C_{in}$

bias? $= 1!$ (C_{in} 个相加, 乘以 1 个即可)

$k \times k$ cov with C_{in} - channel input and C_{out} output

weights: $k^2 C_{in} \times C_{out}$

bias: C_{out}

Backprop

We share para. across different pixels.

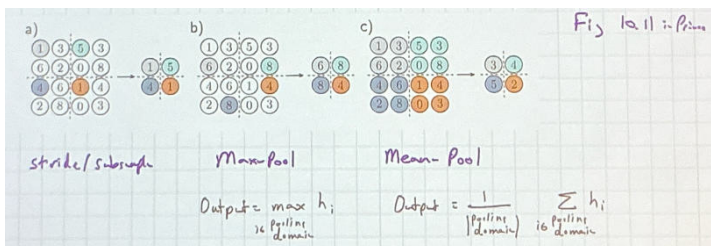
$\Rightarrow$ How can we compute $\frac{\partial L}{\partial w}$ when same w appears in different places?

Easy Answer: Don't worry, PyTorch will handle it. 😊

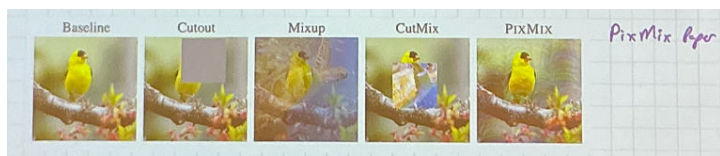
Underlying Answer: 1) Pretend the weight w was actually $\tilde{w}_1, \tilde{w}_2, \dots$ in all the different places and just happened to be that $\tilde{w}_1 = \tilde{w}_2 = \tilde{w}_3 = \dots = w$

2) Now to get $\frac{\partial L}{\partial w} = \sum_i \frac{\partial L}{\partial \tilde{w}_i}$
i.e. Add them up

Pooling



Data Augmentation

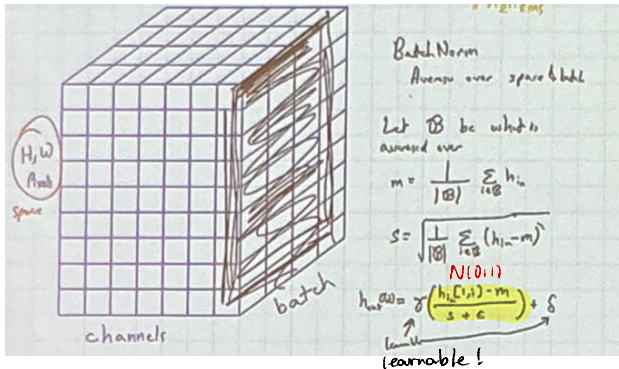


在 20 分钟内完成。

Lecture 10

Training: Stability / Effectiveness.

→ Add Normalize Layers.



RMS - Norm. don't recenter

• 标准的层归一化 (Layer Normalization) 通常包含两个主要步骤:

1. 重新中心化 (Re-centering): 减去均值, 使得数据的中心变为零 ($x - \mu$).
2. 重新缩放 (Re-scaling): 除以标准差, 将数据的方差归一化。

• RMSNorm 的设计者认为, 对于训练稳定性的贡献, 重新缩放 (即控制数值的“大小”或“尺度”) 是其中最关键的部分。而重新中心化 (即调整数值的“偏移”) 对性能的贡献相对较小, 却同样消耗计算资源。

因此, 通过移除均值的计算和减法操作, RMSNorm 极大地简化了整个归一化流程。

What to do at testing / inference.

→ layer norm : no issue

→ Batch norm : Need m, s 但推理时没有 batch !! 怎么办

How to get? 1) Just get an average of m & s from last epoch
2)

方法二 (标准方法): 2) Can be ... (笔记未写完)

• 解读与补充: 虽然笔记只写了个开头, 但它几乎肯定是指业界标准的解决方案: 使用指数移动平均 (Exponential Moving Average, EMA)。

• 具体做法:

1. 在训练过程中, 除了使用每个批次的 m 和 s 来归一化当前批次的数据外, Batch Norm 层还会维护两个额外的变量: $running_mean$ (运行均值) 和 $running_variance$ (运行方差)。
2. 每处理一个批次, 就会用该批次的 m 和 s 来平滑地更新 $running_mean$ 和 $running_variance$ 。
3. 当训练结束时, 这两个“运行统计量”就相当于整个数据集分布的一个非常好的近似估计。
4. 在推理时, Batch Norm 层就直接使用这两个保存下来的、固定的 $running_mean$ 和 $running_variance$ 来对输入进行归一化。

Dropout a data arguementation inside our network data training.

Idea: Just remove / zero-out random activations

⇒ Encourages internal representation to be redundant

Typically used for MLP-style layers

$\vec{h}_{in} \rightarrow$ Cin channels

Bernoulli. $\tilde{h}_e[i] = \begin{cases} 0 & \text{if coin is head} \\ h_e[i] & \text{if --- tail.} \end{cases}$

Training — Dropout randomly. → 0.5 p 的输出得到保留.

What about Testing?

Try: Do nothing? → Fail

Why? → Messes up scale!

eg.

假设网络中有一层，它有 4 个神经元。在某个训练样本下，它们的输出（激活值）是：

• $h = [10, 20, 30, 40]$

再假设下一层只有一个神经元，它做的事情很简单：就是把上一层的这 4 个输出加起来。

我们设定 Dropout 的保留概率 $p = 0.5$ (即一半的神经元会被丢弃)。

1. 训练阶段 (Training Phase)

在训练时，Dropout 是随机的。我们模拟几次随机丢弃的过程：

• 第一次前向传播：随机丢弃了第 2 和第 4 个神经元。

• 激活值变为：[10, 0, 30, 0]

• 下一层接收到的和是：10 + 0 + 30 + 0 = 40

• 第二次前向传播：随机丢弃了第 1 和第 3 个神经元。

• 激活值变为：[0, 20, 0, 40]

• 下一层接收到的和是：0 + 20 + 0 + 40 = 60

• 第三次前向传播：随机丢弃了第 1 和第 4 个神经元。

• 激活值变为：[0, 20, 30, 0]

• 下一层接收到的和是：0 + 20 + 30 + 0 = 50

我们可以看到，下一层接收到的输入值总是在变，但它的“期望值（平均水平）”是多少呢？

原始总和是 10 + 20 + 30 + 40 = 100。因为平均有一半的神经元在工作，所以期望的总和是：

$E[\text{sum}] = 100 \times p = 100 \times 0.5 = 50$

在成千上万的训练中，下一层会慢慢适应，它习惯了接收一个平均值在 50 左右的输入。

Do nothing.

得到 100 的输出！

超出强度 → 模型 fail.

怎么办？

输出 $\times p$!

ResNet

Residual / Skip connections.

存在一个反常现象, 更深的网络居然 underfitting!

→ Degradation (退化).

问题出现在 Backprop, gradient 从后向前, 在每一层都 $\times$ 权重

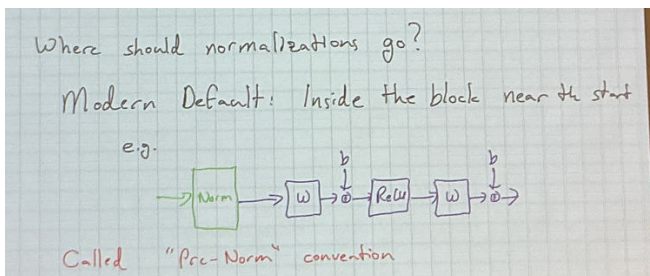
→ 变得很小 $\rightarrow 0$

Vanishing Gradients

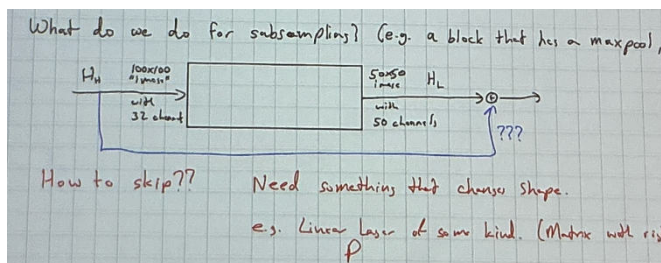
What goes inside a block that's skipped over?

INTUITION: Pre-ResNet Perspective: $f_i(f_i(f_3(\dots f_1(x))\dots))$
ResNet Perspective: $f_{i+1}(x) = f_i(x) + g_i(f_i(x))$
Feels like $\frac{d}{dx} x(u) = g(x(u))$ ODE!

Pre-Norm Convention

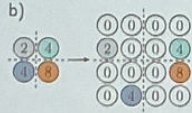


Subsampling.

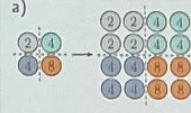


Upsampling

What if we wanted the reverse? Go from low-res to high-res? Upsampling



Max-unpool
Must remember which position
was the max.



Duplication - the opposite of mean

Transpose Conv.

Lecture 11

Recall Concepts / date:

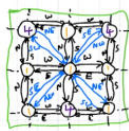
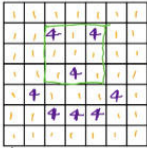
- Weight-sharing to respect symmetries/invariances in domain
- Convolutions — local processing
- Combined with additional channels and depth/nonlinearities for expressivity
- Data Augmentations
- Dropout as an augmentation for MLP-like Elements in hidden layers
- Normalizations for avoiding explosions in activations or gradients
- Residual Connections for successful gradient flow to all layers
- Downsampling/Pooling to get the big picture faster
- Upsampling and U-Nets to restore appropriate detail at original scale

One loose end: stochastic depth regularization

Generalizing CNNs. $\rightarrow$ GNN

First Conceptual Step: Image

Nodes = Pixels



Directed Graph

Edges labeled with one of $\{N, S, E, W, NE, SE, NW, SW\}$

(Always a special relation to self)

A 3×3 Conv can be viewed as receiving messages from neighbors, each of whom has their own labeled edge to me, multiplying by an edge-label-specific weight matrix, and adding those up together with a special self-specific matrix acting on this node's info and a bias.

Note: Implicit Zero-padding and Summation

3×3 Convs are local to a node

Matrices W_i and biases $\vec{b}$ are specific to image of processing

Recall Picture:

3×3 examples

W_{NW}	W_N	W_{NE}
W_W	W_C	W_E
W_{SW}	W_S	W_{SE}

$\text{cin} \rightarrow \{W_i, \vec{b}\} \text{ cin}$

$$\vec{h}_{\text{out}} = \vec{b} + \sum_i W_i \vec{h}_{\text{in},i}$$

Image $\rightarrow$ Graph.

pixel $\rightarrow$ node
pixel $\leftrightarrow$ edge

Taking Stock: We have, in principle, an immediate recipe to generalise CNNs to directed graphs with the same property: Every edge labeled from a discrete finite set.

What Generalizes Immediately: Weight-sharing (Nodes do the same things)
 3x3 convs (As above, matrices for edge-labels)
 Additional Channels, Residual Connections, Dropout for MLPs, Strides Right

What does not generalize: Pooling/Downsampling and Upsampling

What needs discussion: Normalization.

Normalization.

① Batch $\rightarrow$ minibatch 假设每个图都是 iid. Graph 不成立!

② Layer $\rightarrow$ 对于 Node. 丢失连接.

So are we done?	No.	Edge Labels (Finite)	Edge Labels (Infinite)	None
Kinds of Graphs:	Directed	✓		
	Undirected			???

First Attempt: Reduce to already understood case. $A-B$ 变为 $A \geq B$.

1) Treat each undirected edge as both directions $\rightarrow$ replaces $\rightarrow$
 2) Pick a single dummy label for every edge.

What happens? Conv: $\vec{b} + \sum W \vec{h}_i$
 $= \vec{b} + W \sum \vec{h}_i$

So, we have.

- $h_0 \rightarrow$ 自身特征向量
- $n \rightarrow$ How many neighbor
- $\{h_1, \dots, h_n\} \rightarrow$ 邻居

Architectural Constraints! 置换不变性.

So, for each output channel, I can compute.

$$f_w(\vec{h}_0, \sum g_{w_i}(\vec{h}_i)) \quad \text{with learnable } w_1, w_2.$$

$\downarrow$
 other choice π .

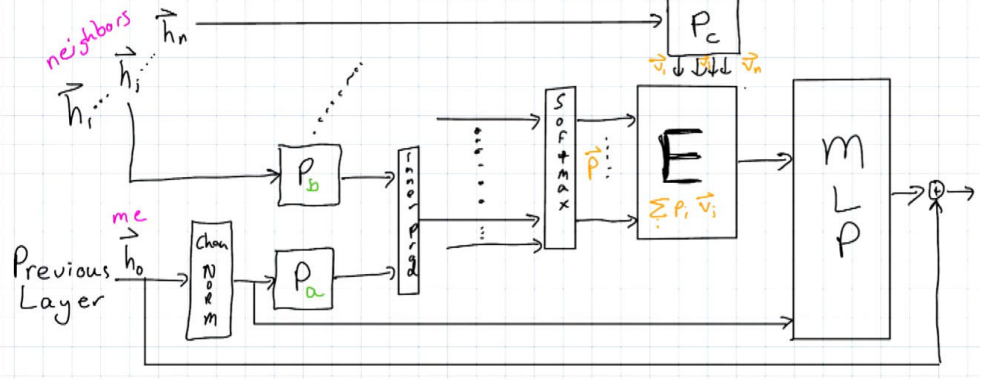
min/max. median

norms

average, variance.

...

Natural Example: Design Choices: Inner Product For Similarity (Learned)
 Expectation-style for "sum" (Learned Linear Function)
 MLP for combining



Lecture 12

Oversmoothing: 过平滑, $\rightarrow$ GNN 层数过多, node representation 趋于一个值.

Pooling / Downsampling & Upsampling.

$\rightarrow$ Not generating new things, but change the scale.

$\rightarrow$ We can add a fake global node connected to everything.

如何实现 GNN 的 pooling? (基于内容, Many $\rightarrow$ Few).

Like Clustering (聚类)!

Recall Classic Clustering Approach: k-Means

0) 随机初始化.

1) Calculate cluster

2) Map point to centers

3) go to 1, 2, repeat.

In GNN: Diffpool!

pooling $\rightarrow$ smaller graph but fully connected.

Make each point belong to **every cluster**! with some prob.

Input : Graph + Embedded vector.

Output : k -length vector for each node

$$S_i = [p_1, \dots, p_k]$$

代表 Node i 属于第 $1, 2, \dots, k$ 个 cluster 的概率。

补充说明！！

Lecture 13

RNN : 用于语言 / NLP

Recurrent Neural Network.

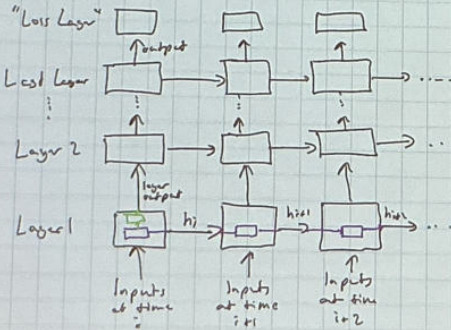
A signal processing perspective...

SP	Neural Nets	Weight Sharing
FIR Filter	CNN	Across Space.
IIR Filter (依赖于过去 input)	RNN	Across Time.

Inside IIR :

$$\begin{cases} \vec{h}_{t+1} = W \vec{h}_t + B \vec{u}_{t+1} \\ \vec{y}_{t+1} = C \vec{h}_{t+1} + D \vec{u}_{t+1} \end{cases}$$

Approach: Keep all the design ideas & principles from CNNs.
Just replace space with time and enforce causality.



Many Ideas Just Carry Over
Vertical Residual Connections
Vertical "1x1" MLPs

Vertical Normalization
Note: Can't average with the future
Just local channels

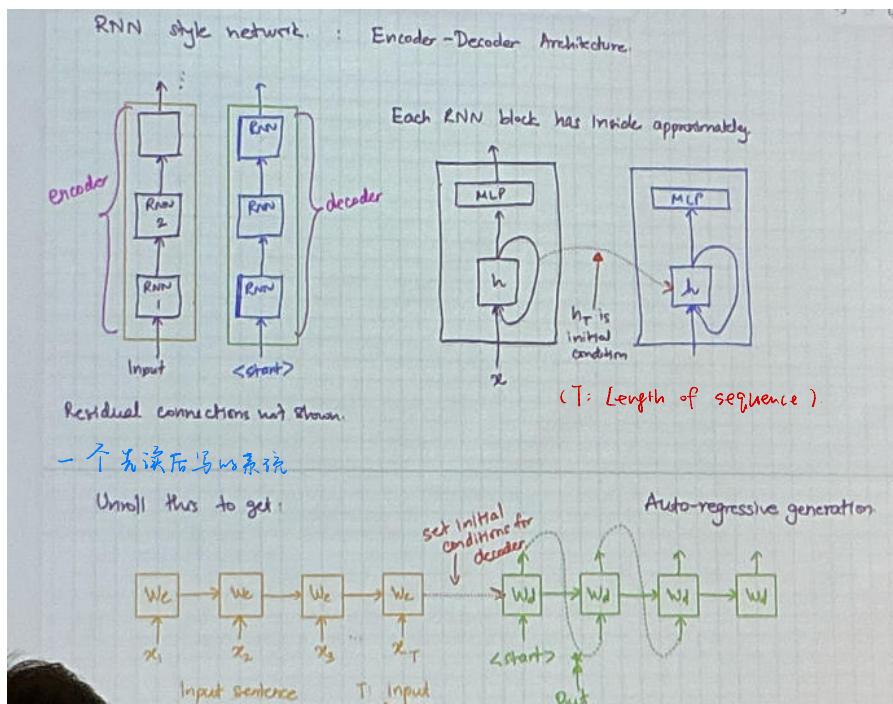
Key Design Questions:

RNN - Quick View

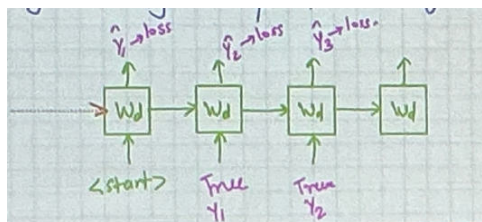
此处銀乙帝深

Lecture 18. Attention.

Encoder & Decoder.



Teacher - Forcing. → 用标准答案作为下一步的输入，即前一步的输出。



Challenge: 对于图，一个 data 的 correct location 很显然
但在 language 中没那么如此。
怎么找到它们的合适位置？

Solution: Hash Table

Idea - 1: Look for the exact match of query q among the stored keys, and then return value corresponding to exact match.

→ Problem: Can't find exact match!

Idea 0: Return the closest one.

→ Problem: Bad gradients. 小扰动不带来优化.

Idea 1: Look at the top l - matches.

So, we have l values.

Try: average. $\frac{1}{l} \sum_{i=1}^l \vec{v}_i$

Next try: weighted avg. $\frac{1}{l} \sum_{i=1}^l \alpha_i \vec{v}_i$

Lecture

11/4

RoPE - Rotary Positional Embedding.

Modify $\vec{k}$, $\vec{q}$ based on their absolute position.

s.t. the score depends on the relative position.

$$\langle \vec{k}, \vec{q} \rangle = \|\vec{k}\|_2 \|\vec{q}\|_2 \cos \theta.$$

Rotation Matrix -

Recall: 旋转矩阵 $R_\theta = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$. 可以将 (x, y) 旋转 θ .

在 RoPE 中: θ 与绝对位置 m 相关.

$$R_m = \begin{pmatrix} \cos(m\theta) & -\sin(m\theta) \\ \sin(m\theta) & \cos(m\theta) \end{pmatrix}. \quad \text{where } \theta = 2\pi f_i \cdot t.$$

应用此矩阵后: $\vec{q}_m = R_m \vec{q}_m$

fixed, not learned.

$$\vec{k}_n = R_n \vec{k}_n$$

$$\langle \vec{q}_m, \vec{k}_n \rangle = (R_m \vec{q}_m)^T (R_n \vec{k}_n) = \vec{q}_m^T R_m^T R_n \vec{k}_n$$

Rotation Matrix 性质: $(R_m)^T$ 等于 R_{-m} (反向旋转 $m\theta$).

$$(R_m)^T R_n = R_{-m} R_n = R_{n-m}$$

$$\Rightarrow \langle \vec{q}_m, \vec{k}_n \rangle = (\vec{q}_m)^T R_{n-m} (\vec{k}_n)$$

RoPE 不是在 d 维进行复杂的高维 Rotation

而是在 $\frac{d}{2}$ 个 2D 平面进行 $d/2$ 次 2D 旋转.

GPT3 Architecture.

Auto - regressive Generation.