6.390

Calculus, Gradient Descent, Feature Transformations, Neural Networks

$$\frac{\partial y}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \vdots \\ \frac{\partial y}{\partial x_n}\end{bmatrix}, \quad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n}\end{bmatrix}, \quad \frac{\partial y}{\partial \mathbf{X}} = \begin{bmatrix}\frac{\partial y}{\partial X_{11}} & \cdots & \frac{\partial y}{\partial X_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial X_{n1}} & \cdots & \frac{\partial y}{\partial X_{nm}}\end{bmatrix}, \quad \left(\frac{\partial \mathbf{A}\mathbf{x}}{\partial \mathbf{x}}\right)^\top = \frac{\partial \mathbf{x}^\top \mathbf{A}}{\partial \mathbf{x}} = \mathbf{A},$$

$$\frac{\partial v\mathbf{u}}{\partial \mathbf{x}} = v\frac{\partial \mathbf{u}}{\partial \mathbf{x}} + \frac{\partial v}{\partial \mathbf{x}}\mathbf{u}^\top, \quad \frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{x}} = \frac{\partial \mathbf{u}}{\partial \mathbf{x}}\frac{\partial \mathbf{g}(\mathbf{u})}{\partial \mathbf{u}}, \quad \frac{\partial \mathbf{u}^\top \mathbf{v}}{\partial \mathbf{x}} = \frac{\partial \mathbf{u}}{\partial \mathbf{x}}\mathbf{v} + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}\mathbf{u}, \quad f(\mathbf{u}+\mathbf{du}) \approx f(\mathbf{u}) + \nabla f_{\mathbf{u}}(\mathbf{u})^\top \mathbf{du}$$

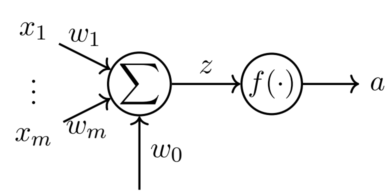
$$\tilde{X} = \begin{bmatrix} —x^{(1)}—(+1) \\ \vdots \\ —x^{(n)}—(+1)\end{bmatrix}, \quad \tilde{Y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(n)}\end{bmatrix}, \quad \theta^* = \left(\tilde{X}^\top\tilde{X} + n\lambda I\right)^{-1}\tilde{X}^\top\tilde{Y}.$$

Batch-Gradient-Descent$(\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, T)$:
Let $\Theta^{(0)}, t \leftarrow \Theta_{\text{init}}, 0$. Repeat $\Theta^{(t)} \leftarrow \Theta^{(t-1)} - \eta(t)\nabla_\Theta f\left(\Theta^{(t-1)}\right)$ until $\left|f\left(\Theta^{(t)}\right) - f\left(\Theta^{(t-1)}\right)\right| < \varepsilon$. Return $\Theta^{(t)}$.

Stochastic-Gradient-Descent$(\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, T)$:
Let $\Theta^{(0)} \leftarrow \Theta_{\text{init}}$. For $t = 1 \cdots T$, randomize $i \in \{1, \cdots, n\}$ and $\Theta^{(t)} \leftarrow \Theta^{(t-1)} - \eta(t)\nabla_\Theta f_i\left(\Theta^{(t-1)}\right)$. Return $\Theta^{(t)}$.

	Loss Function $\mathcal{L}(g,a)$
$\mathcal{L}_{\text{sq}}$	$(g-a)^2$
$\mathcal{L}_{01}$	$\mathbf{1}(g \neq a)$
$\mathcal{L}_{\text{nll}}$	$-a\log g - (1-a)\log(1-g)$
$\mathcal{L}_{\text{nllm}}$	$-\sum_{k=1}^{K} a_k \log g_k$

	Activation Function $f(z)$
ReLU	$z \cdot \mathbf{1}(z \geq 0)$
σ	$1/(1+\exp(-z))$
SM	$\begin{bmatrix}\exp(z_1)/\sum_i \exp(z_i) \\ \vdots \\ \exp(z_K)/\sum_i \exp(z_i)\end{bmatrix}$
Others	z, $\tanh(z)$, $\mathbf{1}(z \geq 0)$, etc



	Objective Function $J(\theta,\theta_0)$	Gradient $\nabla_\theta J$	$\partial J/\partial\theta_0$
J_{mse}	$\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\text{sq}}\left(\theta^\top x^{(i)}+\theta_0, y^{(i)}\right) \overset{\theta_0=0}{=} \frac{\lvert\tilde{X}\theta-\tilde{Y}\rvert^2}{n}$	$\frac{2}{n}\tilde{X}^\top\left(\tilde{X}\theta-\tilde{Y}\right) \quad (\theta_0=0)$	
J_{r}	$J_{\text{mse}}(\theta,\theta_0)+\lambda\lVert\theta\rVert^2$	$\nabla_\theta J_{\text{mse}}+2\lambda\theta$	
J_{lr}	$\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\text{nll}}\left(\sigma\left(\theta^\top x^{(i)}+\theta_0\right), y^{(i)}\right)+\lambda\lVert\theta\rVert^2$	$\frac{1}{n}\sum_{i=1}^{n}\left(g^{(i)}-y^{(i)}\right)x^{(i)}+2\lambda\theta$	$\frac{1}{n}\sum_{i=1}^{n}\left(g^{(i)}-y^{(i)}\right)$
J_{mlr}	$\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}_{\text{nllm}}\left(\text{SM}\left(\theta^\top x^{(i)}\right), y^{(i)}_{\text{onehot}}\right)+\lambda\lVert\theta\rVert^2$	$\frac{1}{n}\sum_{i=1}^{n}x^{(i)}\left(g^{(i)}-y^{(i)}_{\text{onehot}}\right)^\top+2\lambda\theta$	

Poly basis: $\Phi_n(x) = [x_{i_1}\cdots x_{i_k} \mid 0 \leq k \leq n \text{ and } 1 \leq i_j \leq d]^\top$
Radial basis: $\Phi(x) = [f_{x^{(1)}}(x), \cdots, f_{x^{(n)}}(x)]^\top$, $f_p(x) = e^{-\beta\lVert p-x\rVert^2}$
Standardization: $\Phi(x) = \frac{x-\overline{x}}{\sigma}$. Other discrete transformations: **Numeric**, **One-hot**, **Binary**, etc.

$$Z^\ell = W^{\ell\top}A^{\ell-1} + W_0^\ell$$
$$A^\ell = f^\ell(Z^\ell)$$
$$W^\ell : m^\ell \times n^\ell$$
$$W_0^\ell : n^\ell \times 1$$
$$\frac{\partial \mathcal{L}_{\text{nllm}}(\text{softmax}(z),y)}{\partial z} = \text{softmax}(z) - y$$

$$\frac{\partial \text{loss}}{\partial W^\ell} = A^{\ell-1}\cdot\left(\frac{\partial \text{loss}}{\partial Z^\ell}\right)^\top$$
$$\frac{\partial \text{loss}}{\partial W_0^\ell} = \frac{\partial \text{loss}}{\partial Z^\ell}$$
$$\frac{\partial \text{loss}}{\partial Z^\ell} = \frac{\partial A^\ell}{\partial Z^\ell}\cdot W^{\ell+1}\cdots\frac{\partial A^{L-1}}{\partial Z^{L-1}}\cdot W^L\cdot\frac{\partial A^L}{\partial Z^L}\cdot\frac{\partial \text{loss}}{\partial A^L}$$

Convolutional Neural Networks

Conv Layer $k^\ell \times k^\ell \times m^\ell \times m^{\ell-1}$. (Number of filters: m^ℓ. Filter size: $k^\ell \times k^\ell \times m^{\ell-1}$. Input size: $n^{\ell-1} \times n^{\ell-1} \times m^{\ell-1}$).
Stride s^ℓ: Number of pixels shifted in each step.
Padding p^ℓ: Number of layers of pixels added around the edge.
Max-Pooling: Compressing by selecting local maxima.

Transformers

Query, key, value vectors: $q_i = W_q^\top x^{(i)}, k_i = W_k^\top x^{(i)}, v_i = W_v^\top x^{(i)}$ $(Q = XW_q, K = XW_k, V = XW_v)$.

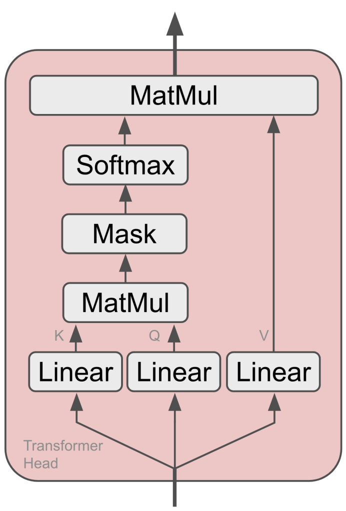
$x^{(i)}$ have dimension d, the QKV vectors have dimension d_k.

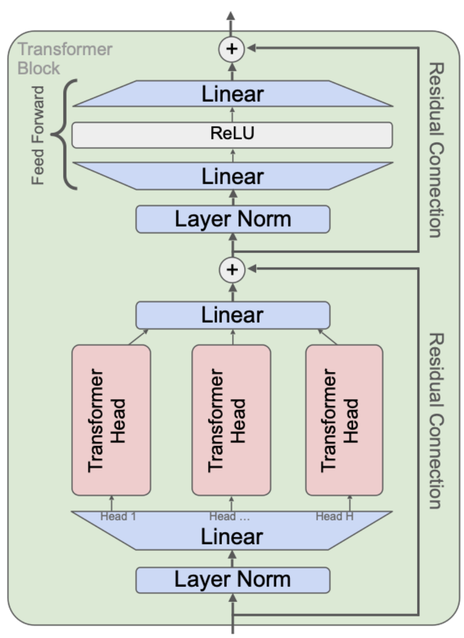
Probability distribution of keys given query: $p(k \mid q) = \mathrm{SM}([q^\top k_1, \cdots, q^\top k_n])$.

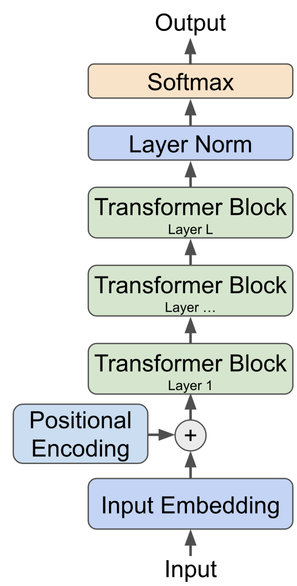
Self-attention matrix: $A = \begin{bmatrix} \mathrm{SM}([q_1^\top k_1, \cdots, q_1^\top k_n]/\sqrt{d_k}) \\ \vdots \\ \mathrm{SM}([q_n^\top k_1, \cdots, q_n^\top k_n]/\sqrt{d_k}) \end{bmatrix} \implies$ Self-attention output: $Y = AV$

Masking: Taking lower triangle of matrix to ignore future values.

$\mathrm{LayerNorm}(z; \gamma, \beta) = \gamma \dfrac{z - \mu_z}{\sigma_z} + \beta$.







Unsupervised Learning

k-means **Clustering** loss: $J_{\mathrm{km}}(\mu, y) = \sum_{j=1}^{k} \sum_{i=1}^{n} \mathbf{1}\left(y^{(i)} = j\right) \left\| x^{(i)} - \mu^{(j)} \right\|^2$ (μ^j is the mean of points in cluster j)

K-MEANS$(k, \tau, \{x^{(1)}, \cdots, x^{(n)}\})$:
Randomly initialize μ, y. For $t = 1 \cdots \tau$, set $y_{\mathrm{old}} \leftarrow y$ and

- $y^{(i)} \leftarrow \mathrm{argmin}_j \left\| x^{(i)} - \mu^{(j)} \right\|^2$ for all $1 \le i \le n$.
- $\mu^{(j)} \leftarrow \mathrm{mean}_j \left\{ x^{(i)} \mid y^{(i)} = j \right\}$ for all $1 \le j \le k$.

until $y = y_{\mathrm{old}}$. Return μ, y.

Autoencoders: $x \xrightarrow{\text{encoder}} a \xrightarrow{\text{decoder}} \tilde{x}$. (Goal: Compress x into a lower-dimensional latent space $\{a\}$ using NN.)

Markov Decision Processes and Reinforcement Learning

$$S = \{\text{states}\},\ A = \{\text{actions}\},\ T(s,a,s') = \Pr\left[S_t = s' \mid S_{t-1} = s, A_{t-1} = a\right],\ R(s,a) = \text{reward},\ \gamma = \text{disc. factor}$$

Value iteration for a policy π (take $h \to \infty$ if infinite-horizon):

$$V_\pi^0(s) = 0, \qquad V_\pi^h(s) = R(s,\pi(s)) + \gamma \sum_{s' \in S} T(s,\pi(s),s') V_\pi^{h-1}(s')$$

Value iteration to find optimal policy (take $h \to \infty$ if infinite-horizon):

$$Q^0(s,a) = 0, \qquad Q^h(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,\pi(s),s') \max_{a' \in A} Q^{h-1}(s',a'), \qquad \pi_h^*(s) = \operatorname*{argmax}_{a \in A} Q^h(s,a)$$

Q-LEARNING$(S, A, s_0, \gamma, \alpha, \varepsilon)$:
Initialize $Q(s,a) \leftarrow 0$ for all $s \in S, a \in A$. Let $s \leftarrow s_0$. Repeat enough times:

- With probability $1 - \varepsilon$, let $a \leftarrow \operatorname*{argmax}_{a \in A} Q(s,a)$ (*exploitation*), otherwise pick $a \in A$ uniformly (*exploration*).
- Execute a on s. Let the result be r and new state be s'.
- Update $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \left(r + \gamma \max_{a' \in A} Q(s',a') \right)$ and make $s' \leftarrow s$.

Non-Parametric Models

BUILDTREE(I, k):
If $|I| \le k$, return LEAF(value $= \text{mean}_{i \in I}\ y^{(i)}$). Else, for each dimension j and splitting value s,

- Denote $I_{j,s}^+ = \left\{ i \in I : x_j^{(i)} \ge s \right\}$ and $I_{j,s}^- = \left\{ i \in I : x_j^{(i)} < s \right\}$.
- Set $E_{j,s} = \sum_{I_{j,s}^+} (\text{squared distance to mean}) + \sum_{I_{j,s}^-} (\text{squared distance to mean})$

Set $(j^*, s^*) = \operatorname*{argmin}_{j,s} E_{j,s}$ and return the root (j^*, s^*) branched into BUILDTREE(I_{j^*,s^*}^+, k) and BUILDTREE(I_{j^*,s^*}^-, k).

Entropy: $H(I) = -\sum_{\text{class } k} p_k \log_2 p_k$ (where p_k is the proportion of elements in I classified as class k).

Weighted average entropy: $\hat{H}_{j,s} = \dfrac{\left|I_{j,s}^+\right|}{\left|I_{j,s}^+\right| + \left|I_{j,s}^-\right|} H(I_{j,s}^+) + \dfrac{\left|I_{j,s}^-\right|}{\left|I_{j,s}^+\right| + \left|I_{j,s}^-\right|} H(I_{j,s}^-)$

Bootstrap Aggregation: Sample B data sets of size n with replacement, construct B predictors, make the bootstrap predictor the average of the B predictors. For classification, use majority vote.