

1 Connectionism

1.1 Perceptron

Threshold Unit $f[w, b](x) = \text{sign}(x \cdot w + b)$ with **Decision Boundary** $x \cdot w + b = 0$. $-\frac{b}{\|w\|}$ is the signed distance of the hyperplane from 0.

Geometric Margin $\gamma[w, b](x, y) = \frac{y(x \cdot w + b)}{\|w\|}$.

Maximum Margin Classifier

$(w^*, b^*) \in \arg\max_{w, b} \gamma[w, b](\mathcal{S})$, with margin $\gamma[w, b](\mathcal{S}) := \min_{(x, y) \in \mathcal{S}} \gamma[w, b](x, y)$.

Perception Learning

if $f[w, b](x) \neq y$: $w \leftarrow w + yx, b \leftarrow b + y$.

Aims to find some solution; assumes version space is non-empty. Does not aim for small err. $w_0 \in \text{span}(x_1, \dots, x_s) \Rightarrow w_t \in \text{span}(x_1, \dots, x_s) \forall t$

Convergence - Novikoff's Theorem

1. If $\exists w^*, \|w^*\| = 1$, s.t. $\gamma[w^*](\mathcal{S}) = \gamma > 0 \Rightarrow w_t \cdot w^* \geq \gamma$.
2. Let $R = \max_{x \in \mathcal{S}} \|x\|$. Then $\|w_t\| \leq R\sqrt{\gamma}$. $\cos \angle(w^*, w_t) = \frac{w^* \cdot w_t}{\|w^*\| \|w_t\|} \geq \frac{\gamma}{\sqrt{\gamma}R} = \sqrt{\frac{\gamma}{R}} \leq 1 \Rightarrow t \leq \frac{R^2}{\gamma}$.

Cover's Theorem for $\mathcal{S} \subset \mathbb{R}^n, |\mathcal{S}| = s$

$C(\mathcal{S}, n)$: # of ways to separate \mathcal{S} in n dimensions. Position of pts does not matter (as long as they are in general position). $C(s+1, n) = 2 \sum_{i=0}^{n-1} s_i$, $C(s, n) = 2^s$ for $s \leq n$. Phase transition at $s = 2n$. For $s < 2n$ empty version space is the exception, otherwise the rule.

1.2 Hopfield Networks

$E(X) = -\frac{1}{2} \sum_{i \neq j} w_{ij} X_i X_j + \sum_i b_i X_i$, where $X_i \in \{\pm 1\}$. $w_{ij} = w_{ji}, w_{ii} = 0$.

Hebbian Learning

Choose patterns $\{x^t\}_{t=1}^s \in \{\pm 1\}^n$, compute weights once using them: $w_{ij} = \frac{1}{n} \sum_{t=1}^s x_i^t x_j^t$; $w_{ii} = 0$. For inference, update X iteratively: $X_i^{t+1} = \text{sign}\left(\sum_j w_{ij} X_j^t + b_i\right)$ asynchronously. Capacity for random, uncorrelated patterns: $s_{\max} \approx 0.138n$. Requiring pattern to be retrieved with high probability: $s \leq \frac{n}{2 \log_2 n}$.

If $X = \text{diag}(1, \dots, 1)$, no reconstruction happens. Under async update step, any Hopfield network is guaranteed to converge.

2 Feedforward Networks

2.1 Linear Models

Linear regression (MSE)

$$L[w](X, y) = \frac{\|Xw - y\|^2}{2n}, \quad \nabla_w L = \frac{X^\top Xw - X^\top y}{n}.$$

Moore-Penrose inverse solution

$w^* = X^*y \in \arg\min_w L[w](X, y)$, where $X^* = \lim_{\delta \rightarrow 0} (X^\top X + \delta I)^{-1} X^\top$ Moore-Penrose inverse.

Stochastic gradient descent update $w_{t+1} = w_t + \eta(y_{i_t} - w_t^\top x_{i_t})x_{i_t}, i_t \sim \mathcal{U}([1, n])$.

Gaussian noise model $y_i = w^\top x_i + \varepsilon_i, \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$, LSQ equivalent to NLL of gaussian noise model.

Ridge regression $h_\lambda[w] = h[w] + \frac{\lambda}{2} \|w\|^2, w^* = (X^\top X + \lambda I)^{-1} X^\top y$.

Logistic function

$$\sigma(z) = \frac{1}{1+e^{-z}}, \sigma(z) + \sigma(-z) = 1.$$

$$\sigma' = \sigma(1-\sigma), \sigma'' = \sigma(1-\sigma)(1-2\sigma)$$

Cross entropy loss

$$\ell(y, z) = -y \log \sigma(z) - (1-y) \log(1-\sigma(z))$$

$$\ell(y, z) = -y \log \sigma(z) - (1-y) \log(1-\sigma(z))$$

Logistic regression with Cross Entropy loss:

$$L[w] = \frac{1}{n} \sum_{i=1}^n \ell_i(y_i, w^\top x_i), \nabla \ell_i = [\sigma(w^\top x_i) - y_i] x_i.$$

2.2 Feedforward Networks

Generic feedforward layers

$$F : \underbrace{\mathbb{R}^{m(n+1)}}_{\text{parameters}} \times \underbrace{\mathbb{R}^n}_{\text{input}} \rightarrow \underbrace{\mathbb{R}^m}_{\text{output}}, F[\theta](x) = \varphi(Wx + b).$$

Layer composition

$$G = F^L[\theta^L] \circ \dots \circ F^1[\theta^1].$$

Layer activations

$$x^l = F^l[\theta^l] \circ \dots \circ F^1[\theta^1](x) = F^l(x^{l-1}), x^0 = x, x^L = F(x).$$

Softmax (z) $_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$, CE-loss in terms of logits $\ell(y, z) = \frac{1}{\ln(2)} \left[-y_z + \ln \sum_j e^{z_j} \right]$. CE between two pmfs: $l(p; q) = -\sum_i p_i \log q_i$. CE with hard labels is NLL-loss.

Residual layer $F[W, b](x) = x + (\varphi(Wx + b) - \varphi(0))$, therefore $F[0, 0] = \text{id}$. Link that propagates x forward is called a **skip connection**. Composing residual layers: number of paths grows exponentially, can include projections for flexibility of changing dimensionality.

2.3 Sigmoid Networks

Sigmoid activation $\sigma(z) = \frac{1}{1+e^{-z}}$.

Hyperbolic tangent activation

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = 2\sigma(2z) - 1.$$

$$\tanh'(z) = 1 - \tanh^2(z).$$

Smooth function approximation

Polynomials, ridge functions ($\varphi(a^\top x + b)$) and MLPs with C^∞ activations are universal approximators.

Weierstrass: Polynomials are universal approximators of $C(\mathbb{R})$ on any given compact I .

Barron's Theorem: For $f : \mathbb{R}^d \rightarrow \mathbb{R}$ with $C_f = \int \|\omega\| |\hat{f}(\omega)| d\omega < \infty$, \exists width- m MLP g_m s.t.: $\int_B |f - g_m|^2 dx \leq O(\frac{1}{m})$.

2.4 ReLU(z) = $\max(0, z)$ networks

Zalavsky's Thoerem: Activation patterns

m ReLU neurons in \mathbb{R}^n . Each neuron's hyperplane $\{w_i^\top x = 0\}$ partitions \mathbb{R}^n into $R(m)$ connected regions of constant activation pattern. $R(m) \leq \sum_{i=0}^{\min(n, m)} \binom{m}{i} \ll 2^m$.

Montufar: Connected linear regions in ReLU network $R(m, L) \geq R(m) \lfloor \frac{m}{n} \rfloor^{n(L-1)}$, L : layers, m : width.

Shekhtman: Piecewise linear functions are dense

in $C([0, 1])$. **Lebesgue**: Piecewise linear function with m pieces can be written $g(x) = ax + b + \sum_{i=1}^{m-1} c_i(x - x_i)_+$; $m+1$ parameters, a, b, c_i .

ReLU networks with 1 hidden layer are universal approximators.

Wang and Sun: Every continuous piecewise linear function $g : \mathbb{R}^n \rightarrow \mathbb{R}$ can be written as a signed sum of k -Hinges with $k \leq n+1$. A k -Hinge is a function $g(x) = \max_{j=1}^k \{w_j^\top x + b_j\}$, generalizes ReLU, known as Maxout unit.

Linear Autoencoder: Optimal $A = DE$, s.t. frobenius norm reconstruction err of AX is minimized, is $D = U_k, E = U_k^\top$, not jointly convex in E and D , but individually. $\hat{X}^* = \arg\min_{\text{rank } (\hat{X})=k} \|X - \hat{X}\|_F^2 = U \Sigma_k V^\top$ SVD.

Implies $\lambda_i(\nabla^2 f(x)) \geq \mu$ for all EVs λ_i of $\nabla^2 f(x)$.

For $f : \mathbb{R} \rightarrow \mathbb{R}$, these become: $L \geq f''(x) \geq \mu \quad \forall x$.

If f μ -strongly convex, L -smooth, GD iterates x_t with $0 < \eta \leq \frac{1}{L}$ converge to unique minimizer x^* at rate $\|x_t - x^*\|^2 \leq (1-\eta\mu)^k \|x_0 - x^*\|^2$. If f convex, diff and L -smooth, with $\eta \leq \frac{1}{L}$, $f(x_t) - f(x^*) \leq \frac{1}{2\eta t} \|x_0 - x^*\|^2$. **Non-Convex case**: If f diff, L -smooth, with minimum f^* , GD iterates with $\eta \leq \frac{1}{L}$ satisfy $\min_{i=0}^t \|\nabla f(x_i)\|^2 \leq \frac{2(f(x_0) - f^*)}{\eta(t+1)}$.

If f diff and L -smooth: $f(x) - f(x^*) \geq \frac{1}{2L} \|\nabla f(x)\|^2$.

Polyak-Lojasiewicz condition

$$\frac{1}{2} \|\nabla f(x)\|^2 \geq \mu(f(x) - \min f) \text{ (forall } x\text{).}$$

μ -strong convex $\Rightarrow \mu$ -PL.

GD Convergence Rates & Learning Rates

L-smooth only: $\eta^* = \frac{1}{L}$. To reach ε -stationary point ($\|\nabla f\| \leq \varepsilon$) needs at most $\frac{2L}{\varepsilon^2}(f(x_0) - \min f)$ steps.

μ -PL + L-smooth: Use $\eta^* = \frac{2}{L+\mu}$. Convergence: $f(x_t) - f(x^*) \leq (1 - \frac{\mu}{L})^t (f(x_0) - f(x^*))$.

3 Gradient-Based Learning

Forward mode is more memory efficient, but backward mode is more runtime efficient. Fwd is $O(\# \text{params})$, reverse is $O(d_{\text{out}})$.

Numerator layout:

For

$f : \mathbb{R}^n \rightarrow \mathbb{R}^m$,

$\left(\frac{\partial y}{\partial x}\right)_{ij} = \frac{\partial y_i}{\partial x_j} \in \mathbb{R}^{m \times n}$ and $f : \mathbb{R}^{n_1 \times n_2} \rightarrow \mathbb{R}$,

$\nabla f(X)_{ij} = \frac{\partial f}{\partial X_{ij}} \in \mathbb{R}^{n_1 \times n_2}$.

3.1 Backpropagation

$x^\ell = \varphi(W^\ell x^{\ell-1} + b^\ell), \quad \frac{\partial \mathcal{L}}{\partial \theta^\ell} = \delta^\ell, \quad \delta^\ell = \frac{\partial \mathcal{L}}{\partial x^\ell} \odot \varphi'(W^\ell x^{\ell-1} + b^\ell), \quad \frac{\partial \mathcal{L}}{\partial \theta^{\ell+1}} = (W^{\ell+1})^\top \delta^{\ell+1}$.

$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z} \frac{\partial z}{\partial x}$. For $x \in \mathbb{R}^n$ different z : $\frac{\partial (Wx)}{\partial x} = W$, element wise f gives: $\frac{\partial f(x)}{\partial x} = \text{diag}(f'(x)), \frac{\partial \hat{y} - y}{\partial y} = 2(\hat{y} - y)^\top, \frac{\partial L}{\partial \hat{y}} = h \cdot \frac{\partial L}{\partial y}$.

$\frac{d}{dx_j} \text{softmax}(x)_i = \text{sm}(x)_i (\delta_{ij} - \text{sm}(x)_j)$

3.2 Gradient Descent

Update: $x_{t+1} = x_t - \eta \nabla f(x_t)$.

Gradient flow ODE $\frac{dx}{dt} = -\nabla f(x)$ gives ideal trajectory to be approximated by gradient descent.

Newton's method gives optimal step for quadratic model: $\Delta x = -[\nabla^2 f(x)]^{-1} \nabla f(x)$.

$\nabla^2_x [x^\top Ax + b^\top x + c] = A + A^\top$

SGD variance

$V[\theta](S) = \frac{1}{s} \sum_{i=1}^s \|\nabla f[\theta](S) - \nabla f[\theta](x_i, y_i)\|^2$.

Polyak averages: $\bar{x}_{k+1} = \frac{k}{k+1} \bar{x}_k + \frac{1}{k+1} x_{k+1}$

SGD convergence rate with Polyak averaging and $\eta_k \propto \frac{1}{k}$

$\mathbb{E}[f(\bar{\theta}_t)] - \min f \leq O\left(\frac{1}{\sqrt{t}}\right)$ (general)

$\mathbb{E}[f(\bar{\theta}_t)] - \min f \leq O\left(\frac{\log t}{t}\right)$ (strongly convex)

$\mathbb{E}[f(\bar{\theta}_t)] - \min f \leq O\left(\frac{1}{t}\right)$ (additionally smooth)

Minibatch SGD: Variance \downarrow by $\propto r$. Can $\uparrow \eta \propto r$.

Var. Reduction with SVRG w/ occasional snapshot $\bar{\theta}$: $\theta_{t+1} = \theta_t - \eta [\nabla f_i(\theta_t) - \nabla f_i(\bar{\theta}) + \nabla f(\bar{\theta})]$.

3.4 Acceleration and Adaptivity

Heavy ball momentum update

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

Nesterov acceleration

$\hat{\theta}_{t+1} = \theta_t + \beta(\theta_t - \theta_{t-1})$

$\theta_{t+1} = \hat{\theta}_{t+1} - \eta \nabla h(\hat{\theta}_{t+1})$

More theoretical grounding than heavy ball.

AdaGrad updates

$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\gamma_t + \varepsilon}} \odot \nabla f(\theta_t)$,

$\gamma^t = \gamma^{t-1} + \nabla f(\theta_t) \odot \nabla f(\theta_t)$.

Adam updates

$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla f(\theta_t), \hat{m}_t = \frac{m_t}{1 - \beta_1^t}$

$v_t = \beta_2 v_{t-1} + (1 - \beta_2) (\nabla f(\theta_t))^2, \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$

$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \varepsilon}}$

RMSprop: Adam without momentum term.

signSGD: $\theta_{t+1} = \theta_t - \eta \text{ sign}(\nabla f_{I_k}(\theta_t))$.

x^4 is strictly convex but not strongly convex, since near 0 the growth of x^4 is slower than x^2 , violating the uniform lower bound on curvature. With f L -smooth and μ -PL, GD with optimal step-size $\arg\min_\eta f(\theta_t - \eta \nabla f(\theta_t))$ converges globally at linear rate. $f(w) = \frac{\|xw - y\|^2}{2} + \lambda \|w\|^2$ satisfies PL-condition. **Muon**: Orthogonalize gradient, should increase the scale of other rare directions which have small magnitude in update but are important.

$\Delta W = -\|\nabla L(W)\|_* \cdot \frac{d_{\text{out}}}{d_{\text{out}}} \cdot U^\top V$ ($\nabla L(W) = U \Sigma V^\top$) minimizes RHS of: $L(W + \Delta W) \leq L(W) + \langle \nabla W L(W), \Delta W \rangle_F + \frac{1}{2} \frac{d_{\text{out}}}{d_{\text{out}}} \|\Delta W\|_F^2$. GD Trajectory always orthogonal to level set.

4 Convolutional Networks

Convolution ($f * g$) $(u) = (g * f)(u)$

$$= \int_{-\infty}^{\infty} f(t) g(u-t) dt = \int_{-\infty}^{\infty} f(u-t) g(t) dt.$$

$(f * g) = \text{Toeplitz-Matrix}(g)f$.

Fourier transform convolution property

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g).$$

Cross-correlation:

$$(g * f)[u] = \sum_t g[t] f[u+t].$$

$$(f(t) * g(t))(y) = (f(-t) * g(t))(y).$$

$$(f(t) * g(t))(-y) = (f(-t) * g(t))(y) = (f(t) * g(-t))(y).$$

Translation-Equivariant Operators = Convolutions.

For $x \in \mathbb{R}^D, h \in \mathbb{R}^K$, $x * h = W_h x$. **Toeplitz Matrix**: $W_h \in \mathbb{R}^{(D-K+1) \times D}$

$$(W_h)_{k,j} = \begin{cases} h_{k-j+k-1} & \text{if } k \leq j \leq k+K-1 \\ 0 & \text{otherwise} \end{cases} \text{ for } h \in \mathbb{R}^K.$$

$$\nabla_w v^\top \text{vec}(\sigma(x * w)) = \text{flip}[x] * [\text{mat}(v) \odot \sigma'(x * w)], \text{ flip rows and columns.}$$

Normal convolution of image $x \in \mathbb{R}^{d \times d}$, kernel $w \in \mathbb{R}^{q \times q} x * w$ requires $\mathcal{O}((d-q)^2 q^2)$. If w separable st $w = uv^\top, u \in \mathbb{R}^q, v \in \mathbb{R}^p, \mathcal{O}(dq(d-q))$

$$\text{Output size: } H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} + 2P - K}{S} \right\rfloor + 1, \text{ Height, Input size, padding, Kernel size, stride.}$$

4.1 Convolutional Networks

Conventions for Padding: Add zeros around input.

ConvNets for Images (r out channel, u in channel)

$$y[r][s, t] = \sum_u \sum_{i,j} w[r, u][i, j] * x[u][s+i, t+j].$$

Number of parameters of a convolutional layer

$$(|r| \times |u|) \cdot (|i| \times |j|) : \text{fully connected} \times \text{patch-size}.$$

4.2 Word2Vec

Per word ω , have input embedding x_ω and output embedding y_ω .

Predict context word ν given center word ω :
 $P(\nu | \omega) = \frac{\exp(x_\omega^\top y_\nu)}{\sum_u \exp(x_\omega^\top y_u)}$.

NLL loss: $\ell_{\omega,\nu} = -x_\omega^\top y_\nu + \ln \sum_\mu \exp(x_\omega^\top y_\mu)$.
 Total: $h(\{x_\omega\}, \{y_\nu\}) = \sum_{(\omega, \nu)} \ell_{\omega,\nu}$ over observed pairs. Use only input embeddings after training.

5 Geometric Deep Learning

Group is set G with a binary operation s.t.:
 1) $(gh)f = g(hf)$, 2) $\exists e \in G$ s.t. $ef = fe = f$,
 3) $\forall g \exists g^{-1} \in G$ s.t. $gg^{-1} = g^{-1}g = e$, 4) $gh \in G \forall g, h$. Abelian if $gh = hg$.

5.1 Sets and Points

Order-invariance property:

$$f(x_1, \dots, x_M) = f(x_{\pi(1)}, \dots, x_{\pi(M)}) \quad (\text{perturbations}).$$

(Permutation) Equivariance property:

$$\begin{aligned} f(x_1, \dots, x_M) &= (y_1, \dots, y_M) \\ f(x_{\pi(1)}, \dots, x_{\pi(M)}) &= (y_{\pi(1)}, \dots, y_{\pi(M)}) \end{aligned}$$

Deep Sets model (invariant layer):
 $f(x_1, \dots, x_M) = \rho(\sum_{m=1}^M \varphi(x_m))$.

Equivariant map construction:

$$\rho : \mathbb{R} \times \mathbb{R}^N \rightarrow Y, (x_m, \sum_{k=1}^M \varphi(x_k)) \mapsto y_m$$

5.2 Graph Convolutional Networks

Feature and adjacency matrices

$$X = \text{mat}(x_1^\top; \dots; x_M^\top), A = (a_{nm}) \text{ with } a_{nm} = 1 \Leftrightarrow \{v_n, v_m\} \in E.$$

Permutation matrix constraints

$$P \in \{0, 1\}^{M \times M} \text{ with single 1 in each row and col.}$$

Graph invariance definition

$$f(X, A) \stackrel{!}{=} f(PX, PAP^\top), \forall P \in \Pi_M.$$

Graph equivariance definition

$$f(X, A) = Pf(PX, PAP^\top), \forall P \in \Pi_M.$$

Node neighborhood features

$$X_m = \{\{x_n : \{v_n, v_m\} \in E\}\}, \quad \{\cdot\} = \text{multiset}$$

Message passing scheme

$$\varphi(x_m, X_m) = \varphi(x_m, \bigoplus_{x \in X_m} \psi(x)), \quad \oplus \text{ is some permutation-invariant operation.}$$

Normalized adjacency matrix

$$\bar{A} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}, \quad D = \text{diag}(d_1, \dots, d_M), d_m = 1 + \sum_{n=1}^M a_{nm}.$$

One GCN layer

$$X^+ = \sigma(\bar{A}XW), W \in \mathbb{R}^{M \times N}.$$

5.2.1 Spectral Graph Theory

Laplacian operator

$$\Delta f = \sum_{n=1}^N \frac{\partial^2 f}{\partial x_n^2}, f : \mathbb{R}^N \rightarrow \mathbb{R}.$$

Graph Laplacian

$$L = D - A, (Lx)_n = \sum_{m=1}^M a_{nm}(x_n - x_m). \quad x^\top Lx = \frac{1}{2} \sum_u \sum_v A_{uv}(x_u - x_v)^2 \geq 0 \text{ (psd).}$$

Normalized Laplacian

$$\tilde{L} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}.$$

Graph Fourier transform

$$L = D - A = U\Lambda U^\top, \hat{f} = U^\top f, f = U\hat{f}. \quad \Lambda := \text{diag}(\lambda_1, \dots, \lambda_M), \lambda_i \geq \lambda_{i+1}.$$

Convolution: $x * y = U((U^\top x) \odot (U^\top y))$.

Filtering operation

$$G_\theta(L)x = UG_\theta(\Lambda)U^\top x$$

Polynomial kernels

$$U(\sum_{k=0}^K \alpha_k \Lambda^k)U^\top = \sum_{k=0}^K \alpha_k L^k$$

Polynomial kernel network layer

$$\begin{aligned} x_i^{l+1} &= \sum_j p_{ij}(L) x_j^l + b_i, \\ p_{ij}(L) &= \sum_{k=0}^K \alpha_{ijk} L^k \end{aligned}$$

GNNs cannot distinguish between certain graphs that are topologically different. Unconstrained set architectures are more powerful. If **WL-test** says graphs are different, they are; but if it says they're the same, they might still be different.

6 Theory of DNNs

6.1 Statistical Learning Theory

Risk decomposition

f^* : optimal predictor over all functions,

\hat{f}_H : $\underset{f \in H}{\text{argmin}} \mathcal{R}(f)$, \hat{f}_H : learned from finite data.

$$\mathcal{R}(\hat{f}_H) - \mathcal{R}(f^*) = \underbrace{\mathcal{R}(\hat{f}_H) - \mathcal{R}(f_H)}_{\text{excess risk}} + \underbrace{\mathcal{R}(f_H) - \mathcal{R}(f^*)}_{\text{estimation error}} + \underbrace{\mathcal{R}(f^*) - \mathcal{R}(\hat{f}_H)}_{\text{approximation error}}$$

Rademacher Complexity $G = \{g_h \mid h \in H\}$:

For $\sigma \in \{-1, 1\}$, measures how well G can fit random noise: $\hat{\mathcal{R}}_{D_n}(G) = \mathbb{E}_\sigma [\sup_{g \in G} \frac{1}{n} \sum_{i=1}^n \sigma_i g(z_i)]$.

$$\mathbb{E}[\sup_{h \in H} \mathcal{R}(h) - \hat{\mathcal{R}}_{D_n}(h)] \leq 2\hat{\mathcal{R}}_{D_n}(G), \quad \mathbb{E}[\mathcal{R}(\hat{h}_H)] \leq \mathcal{R}(h^*) + 2\hat{\mathcal{R}}_{D_n}(G).$$

Double descent:

Beyond the interpolation point, models eventually may level out at a lower generalization error.

Implicit bias towards min norm solutions:

Any convergent algorithm with iterates in $\text{span}\{x_1, \dots, x_n\}$ finds the minimum norm solution.

6.1.1 A PAC-Bayesian result

P prior distribution over functions before seeing data, Q posterior after training.

PAC-Bayesian theorem

Bounds generalization gap for stochastic

classifiers ($f \sim Q$): $E_Q[\mathcal{R}(f)] - E_Q[\hat{\mathcal{R}}_n(f)] \leq \sqrt{\frac{2}{n} [\text{KL}(Q||P) + \ln(2\sqrt{n}/\varepsilon)]}$

• P : prior, Q : posterior (learned). Rate $\tilde{O}(1/\sqrt{n})$

• $\text{KL}(Q||P)$: “information cost” of moving $P \rightarrow Q$

• Insight: generalization depends on **distance moved**, not parameter count

PAC-Bayesian for DNNs

$$P = \mathcal{N}(0, \lambda I), Q = \mathcal{N}(\theta, \text{diag}(\sigma_i^2)), \quad \text{KL}(Q||P) = \sum_i \left[\log \frac{\lambda}{\sigma_i} + \frac{\sigma_i^2 + \theta_i^2 - 1}{2\lambda^2} - \frac{1}{2} \right]$$

Minimize directly: $E_Q[\hat{\mathcal{R}}] + \sqrt{\frac{2}{n} [\text{KL}(Q||P) + \dots]}$

\Rightarrow encourages wide/flat minima (perturbations $\theta + \varepsilon$ must also perform well)

Implementation: Reparameterization: $\tilde{\theta} = \theta + \text{diag}(\sigma_i)\eta, \eta \sim \mathcal{N}(0, I)$, Backprop through θ and σ .

6.2 Linearized DNNs and NTK

Training neural network $f(\theta)(x)$ can be approximated by **linearizing** around initialization θ_0 when parameters change slowly.

Linearization \rightarrow Kernel Regression:

Taylor approximation: $h(\beta)(x) = f(\theta_0)(x) + \beta \cdot \nabla f(\theta_0)(x), \quad \beta = \theta - \theta_0$.

With residuals $\tilde{y}_i = y_i - f(\theta_0)(x_i)$, training becomes **linear regression** with features $\nabla f(\theta_0)(x_i)$: $\min \|\tilde{y}_i - \beta \cdot \nabla f(\theta_0)\|$

Neural Tangent Kernel (NTK):

Definition: $k_\theta(x, x') := \nabla f(\theta)(x) \cdot \nabla f(\theta)(x')$.

Dual representation:

$$h(\alpha)(x) = f(\theta_0)(x) + \sum_{i=1}^n \alpha_i k_{\theta_0}(x_i, x).$$

Optimization problem: $\min_{\alpha} \frac{1}{2} \|K_{\theta_0} \alpha - \tilde{y}\|^2$

Optimal solution (kernel regression):

$$\alpha^* = K_{\theta_0}^\dagger (y - f(\theta_0)), h^*(x) = k_{\theta_0}(x)^\top \alpha^*$$

Functional Gradient Flow

Training dynamics in function space:

$$\dot{f}(\theta) = K(\theta)(y - f(\theta))$$

• If $K(\theta)$ constant \rightarrow **linear ODE** with closed-form solution

• If $K(\theta)$ evolves \rightarrow **nonlinear dynamics**, feature learning

Infinite-Width Limit

Initialization: $w_{ij}^{(\ell)} \sim \frac{\sigma_w}{\sqrt{m_\ell}} \mathcal{N}(0, 1)$. Result: As width $m \rightarrow \infty$: $k_{\theta(t)} \rightarrow k_\infty$ (constant during training).

• Kernel becomes **deterministic** (depends only on architecture/init scheme)

• Training = kernel regression with frozen k_∞

• No feature learning

Finite-width: $\|K(\theta_0) - K(\theta(t))\| = \mathcal{O}(\frac{1}{m})$

Why Kernel Stays Constant:

Kernel grad $\nabla K = \nabla^2 f(x) \nabla f(z) + \nabla^2 f(z) \nabla f(x)$,

at $m = \infty$: $\nabla^2 f \rightarrow 0 \Rightarrow \nabla K \rightarrow 0 \rightarrow$ kernel frozen.

Lazy Training (NTK Regime)	Feature Learning (Rich Regime)
$m \rightarrow \infty$, small LR	Finite width, normal LR
K const \rightarrow linear dy.	K evolves \rightarrow nonlinear nematics
No feature learning	Learns representations
Theoretically tractable	SOTA performance

For $y = f(x) + \varepsilon, f(x) = x^\top w, \varepsilon \sim \mathcal{N}(0, \sigma^2)$, $p(y \mid X, w) = \mathcal{N}(X^\top w, \sigma^2 I)$. If $w \sim \mathcal{N}(0, \Sigma)$, $p(w \mid X, X) = \mathcal{N}(\Sigma_{\text{post}}^{-\frac{1}{2}} X^\top w, \Sigma_{\text{post}})$, where $\Sigma_{\text{post}}^{-1} = \Sigma^{-1} + \frac{1}{\sigma^2} X X^\top$. Maximizing log $p(w \mid X, X)$ is same as minimizing least-squares with ℓ_2 penalty $\frac{1}{2} w^\top \Sigma^{-1} w$, ridge when $\Sigma = \lambda^{-1} I$. **Predictive:** $f_* \mid x_*, y, X = x_*^\top w \sim \mathcal{N}(x_*^\top \mu_{\text{post}}, x_*^\top \Sigma_{\text{post}} x_*)$. $y_* \mid x_*, y, X$ adds σ^2 . $f = x_*^\top w$ with $w \sim \mathcal{N}(0, \Sigma)$ is GP w $k(x, x') = x_*^\top \Sigma x'$.

6.3.2 NNGPs

Setup: Random 1-hidden-layer NN with m units: $f(x) = v_0 + \frac{1}{\sqrt{m}} \sum_{j=1}^m v_j \varphi(\theta_j^\top x)$. Random init: $v_0 \sim \mathcal{N}(0, \sigma_0^2), \mathbb{E}[v_j^2] = \sigma_v^2, \text{Cov}(\theta_j) = \Sigma_\theta$.

Result: As $m \rightarrow \infty$, $f(\cdot) \rightarrow$ GP($0, k$) where $k(x, x') = \sigma_v^2 + \sigma_v^2 \mathbb{E}_\theta[\varphi(\theta^\top x) \varphi(\theta^\top x')]$

Monte Carlo approximation: Sample B random NNs $\{f_b\}_{b=1}^B$, define features:

- $\varphi(x) = \frac{1}{\sqrt{B}} (f_1(x), \dots, f_B(x))^\top$
- $\Phi = [f_1 \dots f_B] \in \mathbb{R}^{n \times B}$ (feature matrix)
- $\hat{K} = \Phi \Phi^\top$ (approximate kernel matrix)

GP regression: Posterior mean and variance: $\mathbb{E}[f(x) \mid y] = \varphi(x)^\top (\Phi^\top \Phi + \sigma^2 I_B)^{-1} \Phi^\top y$
 $\text{Var}[f(x) \mid y] = \sigma^2 \varphi(x)^\top (\Phi^\top \Phi + \sigma^2 I_B)^{-1} \varphi(x)$

Key advantage: Inverts $B \times B$ matrix instead of $n \times n$ when $B \ll n$.

7 Generative Models

7.0.1 Linear Autoencoders

Setup. Encoder $C \in \mathbb{R}^{k \times d}$, decoder $D \in \mathbb{R}^{d \times k}$, data $X \in \mathbb{R}^{d \times n}$ (centered cols): $\min_{C, D} \|X - DCX\|_F^2$.

Optimal Solution (PCA). Let $S = XX^\top$ with eigendecomposition $S = Q\Lambda^2 Q^\top, \lambda_1 \geq \dots \geq \lambda_d \geq 0$. Optimal reconstruction via rank- k projection: $\hat{X} = U_k^* U_k^{*\top} X$ where $U_k^* = Q_{[:, 1:k]}$ are top- k eigenvectors of S (equiv, top- k left singular vectors of X).

- Any $C = U_k^* U_k^{*\top}, D = A^{-1}(U_k^*)^\top$ is optimal ($\forall A$)
- Reduces to truncated SVD: $\hat{X} = U^* \Lambda_k V^\top$ with $\Lambda_k = \text{Diag}(\lambda_1, \dots, \lambda_k, 0, \dots, 0)$
- Convex objective with no spurious local minima (gradient descent finds global optimum)
- Singular vectors may not be uniquely identified

7.0.2 Factor analysis

Latent Variable Models are a generic way to describe generative models. Latent variable $z \sim p(z)$, conditional models for observables $x, p(x|z)$, observed data model: $p(x) = \int p(x|z)p(z) dz$.

Mixture models: simple discrete models: $z \in [K], p(z)$ mixing proportions, $p(x|z)$ condit. densities.

$$x \sim \mathcal{N}(\mu, \Sigma): p(x; \mu, \Sigma) = \frac{\exp[-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)]}{\sqrt{(2\pi)^n \det(\Sigma)}}$$

7.0.3 Linear Factor Analysis

- Latent prior: $z \sim \mathcal{N}(0, I)$, $z \in \mathbb{R}^m$
 - Observation: $x = \mu + Wz + \eta$, $\eta \sim \mathcal{N}(0, \Sigma)$
 - Independence: $\eta \perp z$
 - Typically $m < n$ (fewer factors than features)
- Marginal distribution:** $x \sim \mathcal{N}(\mu, WW^\top + \Sigma)$
- WW^\top : shared variance (low-rank, explained by latent factors)
 - Σ : unique variance (diagonal, observation-specific)
- Non-identifiability: $(WQ)(WQ)^\top = WQQ^\top W^\top = WW^\top$ for any orthogonal Q . Factors only identifiable up to rotations/reflections. \Rightarrow Use factor rotations (varimax, etc.) for interpretability.

MLE estimation: $\theta = (\mu, W) \xleftarrow{\text{max}} \log p(X; \mu, W)$

- $\hat{\mu} = \frac{1}{s} \sum_{i=1}^s x_i$ (closed form)
- No closed form for $W \rightarrow$ use GD or EM algorithm

Posterior (encoder): $p(z|x) = \frac{p(x|z)p(z)}{p(x)}$.

$$\mu_{z|x} = W^\top (WW^\top + \Sigma)^{-1}(x - \mu).$$

$$\Sigma_{z|x} = I - W^\top (WW^\top + \Sigma)^{-1}W.$$

Probabilistic PCA: Special case $\Sigma = \sigma^2 I$. Optimal i -th column: $w_i = \rho_i u_i$, $\rho_i^2 = \max\{0, \lambda_i - \sigma^2\}$. $W = U_m L_m$ where (λ_i, u_i) is i -th eigenpair of data covariance.

As $\sigma \rightarrow 0$: $\mu_{z|x} \rightarrow W^\dagger(x - \mu)$ (standard PCA). If W has orthogonal columns, then $W^\dagger = W^\top$.

7.1 Variational Autoencoders

$z \in \mathbb{R}^d$ is learned embedding of x . For generation, $z \sim \mathcal{N}(0, I)$, decoder $p_\theta(x|z)$ maps latent to data.

Problem: $p_\theta(x) = \int p(z)p_\theta(x|z) dz$ intractable. **Solution:** Maximize ELBO instead: $\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\text{KL}}(q_\phi(z|x) \parallel p(z))$

- Reconstruction:** Encode $x \rightarrow z$, decode back
- KL term:** Keep encoder output close to prior $p(z) \sim \mathcal{N}(0, I)$, ensures generation using latents

Encoder $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \text{diag}(\sigma_\phi^2(x)))$.

KL closed form: $D_{\text{KL}}(q_\phi(z|x) \parallel p(z)) = \frac{1}{2} \sum_{j=1}^d (\sigma_{\phi,j}^2 + \mu_{\phi,j}^2 - 1 - \log \sigma_{\phi,j}^2)$.

$\text{KL}(\mathcal{N}(\mu_0, \sigma_o^2) \parallel \mathcal{N}(\mu_1, \sigma_1^2)) = \frac{1}{2} \left(\frac{\sigma_1^2}{\sigma_o^2} + \frac{(\mu_0 - \mu_1)^2}{\sigma_o^2} - 1 + \log \frac{\sigma_1^2}{\sigma_o^2} \right)$

$$\text{KL}(p\parallel q)) = \mathbb{E}_p [\log \frac{p(x)}{q(x)}]$$

- Fwd KL:** $q_1^* = \arg \min_{q \in Q} \text{KL}(p \parallel q)$
- Rev KL:** $q_2^* = \arg \min_{q \in Q} \text{KL}(q \parallel p)$

Rev KL: Mode-seeking ($p = 0 \Rightarrow q = 0$), FwdKL: Mean-seeking ($p \neq 0 \Rightarrow q \neq 0$). MLE minimizes fwd KL to empirical \mathcal{D} .

$$\log p_\theta(x|z) = -\frac{1}{2\sigma^2} \|x - \mu_\theta(z)\|_2^2 - \frac{d}{2} \log(2\pi\sigma^2).$$

Reparameterization trick: $z = \mu_\phi(x) + \sigma_\phi(x) \odot \varepsilon$, $\varepsilon \sim \mathcal{N}(0, I)$, enables backprop through sampling.

$$\log p_\theta(x) - \text{ELBO} = D_{\text{KL}}(q_\phi(z|x) \parallel p_\theta(z|x)), \text{ tight when } q_\phi = \text{true posterior.}$$

Monte

Carlo estimation: $E_{q_\phi(z|x)}[\log p_\theta(x|z)] \approx -\frac{1}{2\sigma^2 K} \sum_{k=1}^K \|x - \mu_\theta(z_k)\|_2^2 - \frac{d}{2} \log(2\pi\sigma^2)$.

Generative Classifiers Given $y \in \{0, 1\}$, $p(y=1) = p(y=0) = \frac{1}{2}$, $p(x|y) = \mathcal{N}(x; \mu_y, I_d)$, where $x \sim \mathcal{N}(\mu_0, \mu_1 \in \mathbb{R}^d)$, $p(y=1|x) = \frac{p(y=1)p(x|y=1) + p(y=0)p(x|y=0)}{p(y=1)p(x|y=1) + p(y=0)p(x|y=0)} = \frac{\frac{1}{2}(2\pi)^{-\frac{d}{2}} \exp(-\frac{1}{2}\|x - \mu_1\|^2)}{\frac{1}{2}(2\pi)^{-\frac{d}{2}} \exp(-\frac{1}{2}\|x - \mu_1\|^2) + \frac{1}{2}(2\pi)^{-\frac{d}{2}} \exp(-\frac{1}{2}\|x - \mu_0\|^2)} = \frac{1 + \exp(\frac{1}{2}\|x - \mu_1\|^2 - \frac{1}{2}\|x - \mu_0\|^2)}{1 + \exp(-(\frac{1}{2}\|x - \mu_1\|^2 + \frac{1}{2}\|x - \mu_0\|^2))}$ equiv to logistic regression where $p(y=1|x) = \sigma(w^\top x + b)$ with $w = \mu_1 - \mu_0$, $b = \frac{1}{2}(\|\mu_0\|^2 - \|\mu_1\|^2)$.

ELBO for Hierarchical VAEs: Model x by decoding from latents $z = (z_1, \dots, z_L)$. $p_\theta(x, z) = p_\theta(z_1) \prod_{i=1}^{L-1} p_\theta(z_i|z_{i+1}) p(z_L)$. Inference top-down: $q_\phi(z|x) = q_\phi(z_L|x) \prod_{i=1}^{L-1} q_\phi(z_i|z_{i+1})$.

ELBO for HVAE: $\mathcal{L}(x) = \mathbb{E}_{z|x \sim q_\phi} [\log \frac{p_\theta(x,z)}{q_\phi(z|x)}]$ $= \mathbb{E}_{z|x \sim q_\phi} [\log p_\theta(x|z_1) + \log \frac{p_\theta(z_1|z_2)}{q_\phi(z_1|z_2)} + \dots + \sum_{i=2}^{L-1} \log \frac{p_\theta(z_i|z_{i+1})}{q_\phi(z_i|z_{i+1})} + \log \frac{p_\theta(z_L)}{q_\phi(z_L|z_{L-1})}]$.

Change of variables spherical to 3D euclidian $(x, y, z) \mapsto (r \cos \theta \cos \phi, r \cos \theta \sin \phi, r \sin \theta)$. Lengths of the three sides of an infinitesimal cuboid whose diagonally opposite vertices are at r, θ, ϕ and $(r+dr, \theta+d\theta, \phi+d\phi)$ are $(dr, rd\theta, r \cos \theta d\phi)$. Volume is $r^2 \cos \theta dr d\theta d\phi$. Determinant of jacobian $\left| \frac{\partial(x,y,z)}{\partial(r,\theta,\phi)} \right| = r^2 \cos \theta$.

Density on spherical coordinates $p(r, \theta, \phi) \rightarrow$ density on Euclidian coordinates is $p(x, y, z) = p(r, \theta, \phi) \left| \frac{\partial(x,y,z)}{\partial(r,\theta,\phi)} \right|^{-1}$. Infinitesimal probability mass of the cuboid above is equal to the mass of a Euclidian cuboid of size (dx, dy, dz) at (x, y, z) .

7.2 Normalizing Flows

Transform simple distribution $z \sim \mathcal{N}(0, I)$ through invertible map T to get complex $x = T(z)$. Exact likelihood (no ELBO like VAEs), easy sampling.

Change of Variables Formula:

$$p_x(x) = p_z(T^{-1}(x)) \cdot \det J_{T^{-1}}(x), \quad |\det J_{T^{-1}}(x)| = \frac{1}{|\det J_T(T^{-1}(x))|}.$$

Diffeomorphism: T is bijective, differentiable, with differentiable inverse. Guarantees $\det J_T \neq 0$.

Computational problem: Computing $\det J$ is $O(d^3)$ for dense Jacobian. **Solution:** Design T s.t. Jacobian is **triangular**, then only $O(d)$!

Two architectures with triangular Jacobians:

	MAF	IAF
Fast / parallel	Density eval	Sampling
Slow / sequential	Sampling	Density

Coupling layers: Trick that makes both directions fast, at the cost of being less expressive per layer.

7.3 Autoregressive Models

$$p(x) = \prod_{i=1}^d p(x_i | x_{<i}).$$

7.4 Generative Adversarial Networks

Likelihood-free generative model: train via adversarial game between two networks: **Generator** G_θ maps latent $z \sim p_z$ (typically Gaussian) to fake samples; **Discriminator** D_φ : outputs prob that input is real.

GAN

Objective: $\min_\theta \max_\varphi \mathbb{E}_{x \sim p_r} [\log D_\varphi(x)] + \mathbb{E}_{z \sim p_z} [\log (1 - D_\varphi(G_\theta(z)))]$

- Discriminator** maximizes: correctly classify real (high D) and fake (low D)
- Generator** minimizes: fool discriminator (make $D(G(z))$ high)

Common alternative objective for the generator is to maximize $\mathbb{E}_{z \sim p(z)} [\log D(G(z))]$ instead of minimizing $\mathbb{E}_{z \sim p(z)} [\log (1 - D(G(z)))]$ to help mitigate vanishing gradient problem when discriminator becomes good, i.e. $D(G(z)) \rightarrow 0$. For a fixed generator G , **optimal discriminator** D^* is given by $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}$.

If **discriminator optimal**, GAN objective reduces to $V(D^*, G) = 2D_{\text{JS}}(p_{\text{data}} \parallel p_G) - \log 4$.

7.4.1 Theoretical Foundation

Binary classification with $p(y=1) = p(y=0) = \frac{1}{2}$:

- $y=1$: sample from real $p_{r(x)}$
- $y=0$: sample from generator $p_{\theta(x)}$

Bayes Optimal Classifier (prob that x is real): $p_\theta(x) = P(y=1|x) = \frac{p_{r(x)}}{p_{r(x)} + p_{\theta(x)}}$.

Generator Logistic Objective = JS Divergence:

$$\ell^*(\theta) = \mathbb{E}_{\tilde{p}_\theta(x,y)} [y \ln q_\theta(x) + (1-y) \ln(1 - q_\theta(x))] = \text{JS}(p_r \parallel p_\theta) - \ln 2.$$

Jensen-Shannon Divergence: $\text{JS}(p_r \parallel p_\theta) = \frac{1}{2} D_{\text{KL}}(p_r \parallel p_m) + \frac{1}{2} D_{\text{KL}}(p_\theta \parallel p_m)$, $p_m = \frac{p_r + p_\theta}{2}$.

Bounded: $0 \leq \text{JS}(p_r \parallel p_\theta) \leq \log 2$.

Jensen Inequality

If φ convex: $\varphi(\mathbb{E}[X]) \leq \mathbb{E}[\varphi(X)]$. If concave, other way around.

7.4.2 Training

Alternating SGD (heuristic, may diverge!). Training is **Saddle-point problem**, notoriously unstable!

JS Divergence Saturates when distributions don't overlap. If p_r and p_θ have disjoint supports: discriminator perfect, no gradient for generator!

Wasserstein Distance (Earth Mover's Distance):

$$W(p_r, p_\theta) = \inf_{\gamma \in \Pi(p_r, p_\theta)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|]$$

Minimum total "work" to transport mass from p_r to p_θ . Provides meaningful gradients even without overlap.

Dual (Kantorovich-Rubinstein):

$$W(p_r, p_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p_r} [f(x)] - \mathbb{E}_{x \sim p_\theta} [f(x)].$$

Maximize gap between avg score of real vs fake samples w.r.t. Lipschitz constraint. Max achievable gap = Wasserstein distance.

WGAN uses critic f_w (not classical discriminator!):

$$\min_\theta \max_w \mathbb{E}_{x \sim p_r} [f_w(x)] - \mathbb{E}_{z \sim p_z} [f_w(G_\theta(z))].$$

Enforcing Lipschitz:

- Weight clipping** (original): crude, problematic
- Gradient penalty:** add $\lambda \mathbb{E}_{\hat{x}} [\| \nabla_{\hat{x}} f_w(\hat{x}) \|_2 - 1]^2$

Mode Collapse: Generator produces only few samples that fool discriminator, ignoring full distribution diversity.

7.5 Diffusion Models

Forward process (fixed): Gradually add Gaussian noise over T steps until data becomes pure noise.

Fwd step: $q(x_t | x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$. Full fwd process: $q(x_{1:T} | x_0) = \prod_{t=1}^T q(x_t | x_{t-1})$.

Noise schedule: $\{\beta_t \in (0, 1)\}_{t=1}^T$ noise added at each step. Define: $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$.

Direct sampling (reparameterization trick):

$$q(x_t | x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I). \quad x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I).$$

Reverse process (learned): Train NN to denoise step by step: $p_\theta(x_{t-1} | x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I)$.

For small β_t , the reverse $q(x_{t-1} | x_t)$ is also Gaussian.

$\log p_\theta(x_0)$ intractable, so derive **Variational Lower Bound (VLB)**: $-\log p_\theta(x_0) \leq \mathcal{L}_{\text{VLB}} =$

$$\mathbb{E}_q \left[\log \frac{q(x_{1:T} | x_0)}{p_\theta(x_{1:T} | x_0)} \right]. \quad \text{Decomposition into 3 terms: } \mathcal{L}_{\text{VLB}} = \underbrace{D_{\text{KL}}(q(x_T | x_0) \parallel p(x_T))}_{L_T} +$$

$$\underbrace{\sum_{t=2}^T \mathbb{E}_{q(x_t | x_0)} [D_{\text{KL}}(q(x_{t-1} | x_t, x_0) \parallel p_\theta(x_{t-1} | x_t))]}_{L_{t-1}} - \underbrace{\mathbb{E}_{q(x_1 | x_0)} [\log p_\theta(x_0 | x_1)]}_{L_0}$$

- L_T : Is $q(x_T | x_0) \approx \mathcal{N}(0, I)$? Not optimized.
- L_{t-1} : Match learned reverse to true reverse
- L_0 : Reconstruction term

Tractable Reverse Posterior $q(x_{t-1} | x_t, x_0)$ is Gaussian with closed form (product of Gaussians): $q(x_{t-1} | x_t, x_0) = \mathcal{N}(x_{t-1}; \mu_{q,t}(x_t, x_0), \sigma_t^2 I)$,

$$\mu_{q,t}(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \varepsilon_0 \right), \quad \sigma_t^2 = \frac{1}{1 - \alpha_{t-1}} \beta_t.$$

7.6 Noise Prediction Parameterization

Predict the noise $\varepsilon_\theta(x_t, t)$ instead of mean directly. Parameterize learned mean to mirror true posterior: $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \varepsilon_\theta(x_t, t) \right)$. Since both distributions are Gaussian with same variance: $D_{\text{KL}}(\mathcal{N}(\mu_q, \sigma_t^2 I) \parallel \mathcal{N}(\mu_p, \sigma_t^2 I)) = \frac{1}{2\sigma^2} \|\mu_q - \mu_p\|^2$. This simplifies L_{t-1} to comparing noise: $L_{t-1} = \mathbb{E}_{x_0, \varepsilon_0} \left[\frac{(1 - \alpha_t)}{2\alpha_t(1 - \alpha_t)\sigma_t^2} \|\varepsilon_0 - \varepsilon_\theta(x_t, t)\|^2 \right]$. $\mathcal{L}_{\text{simple}} = \mathbb{E}_{t \sim [1, T], x_0, \varepsilon_0} \|\varepsilon_0 - \varepsilon_\theta(x_t, t)\|^2$.

Training	Sampling
<ol style="list-style-type: none"> Sample real image $x_0 \sim q(x_0)$ Sample random timestep $t \sim \text{Uniform}\{1, \dots, T\}$ Sample noise $\varepsilon \sim \mathcal{N}(0, I)$ Compute noisy image: $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon$ Grad step on $\nabla_\theta \mathcal{L} = \varepsilon_\theta(x_t, t)^2$ 	<ol style="list-style-type: none"> Sample $x_T \sim \mathcal{N}(0, I)$ For $t = T, \dots, 1$: <ul style="list-style-type: none"> $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$ $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}}\varepsilon_\theta(x_t, t)) + \sigma_t z$ Return x_0

Cosine noise schedule performs better than linear.

Used architecture is U-Net. **Input:** Noisy image $x_t + \text{timestep } t$; **Output:** Predicted noise $\varepsilon_\theta(x_t, t)$.

Model **conditional distribution** $p_\theta(x_{0:T} | y)$ where y is condition (class, text, image). Extend denoiser to take y as input.

Latent Diffusion Models (LDM)

run diffusion in **compressed latent space** instead of pixel space.

8 Tricks

Short Connections in DNs: Add less deep paths to a very deep network. **Residual connections**:

shortcut and add back in. Skip connections: concatenate.

For $y_1 = \alpha f(x, \Theta_1) + x$, $x \in \mathbb{R}^d$, $y_i \in \mathbb{R}^d$, $\Theta_i \in \mathbb{R}^{d \times d}$ and $y_{i \geq 2} = \alpha f(y_{i-1}, \Theta_i) + y_{i-1}$, $l = L(y_n)$ it holds that $\frac{\partial y_k}{\partial y_{k-1}} = \alpha \frac{\partial f(y_{k-1}, \Theta_k)}{\partial y_{k-1}} + I_d$ and $\frac{\partial y_k}{\partial \Theta_k} = \alpha \frac{\partial f(y_{k-1}, \Theta_k)}{\partial \Theta_k}$. By applying the chain rule we have $\frac{\partial l}{\partial \Theta_k} = \frac{\partial L(y_n)}{\partial y_n} \frac{\partial y_n}{\partial y_{n-1}} \dots \frac{\partial y_{k+1}}{\partial y_k} \frac{\partial f(y_{k-1}, \Theta_k)}{\partial \Theta_k}$. Set $\alpha^2 = \frac{\alpha}{n}$ for $a > 0$ s.t. $\lim_{n \rightarrow \infty} \mathbb{E}\|y\|^2 < \infty$.

8.1 Weight Decay & Early Stopping

L2 regularization

$$\mathcal{R}_\Omega(\theta; \mathcal{S}) = \mathcal{R}(\theta; \mathcal{S}) + \Omega(\theta), \quad \Omega_\mu(\theta) = \frac{\mu}{2} \|\theta\|^2, \mu \geq 0.$$

Only penalize weights, not biases. **GD upd w/ WD:** $\Delta\theta = -\eta \nabla \mathcal{R}(\theta) - \eta \nabla \Omega_\mu(\theta) = -\eta \nabla \mathcal{R}(\theta) - \eta \mu \theta$.

Geometric interpretation (local quadratic approx): Regularized optimum: $\theta_\mu^* = (H + \mu I)^{-1} H \theta^*$, where $H = Q^\top \Lambda Q$ gives $\theta_\mu^* = Q \text{diag}\left(\frac{\lambda_i}{\lambda_i + \mu}\right) Q^\top \theta^*$.

$\lambda_i \gg \mu: \frac{\lambda_i}{\lambda_i + \mu} \approx 1 \rightarrow$ weak shrinkage (important dirs).

$\lambda_i \ll \mu: \frac{\lambda_i}{\lambda_i + \mu} \approx 0 \rightarrow$ strong shrinkage (flat dirs).

Adaptively shrinks based on loss geometry, preserves important dirs, removes unnecessary complexity.

Early stopping: Rather than training to convergence, stop when validation performance plateaus. Analysis shows that this is approximately equivalent to L2 regularization. GD trajectories can be approximated as $\theta(k) = [I - (I - \eta \Lambda)^k] \theta^*$. For small step sizes, behaves like weight decay when $k = \frac{1}{\eta \mu}$.

L^1 regularized second-order approximation of an arbitrary loss function around optimal θ^* is $R_{L^1}(\theta) \approx R(\theta^*) + \frac{1}{2}(\theta - \theta^*)^\top H(\theta - \theta^*) + \lambda \|\theta\|_1$. Assuming $H = \text{diag}(h_1, \dots, h_d)$, we get $R_{L^1}(\theta) \approx \sum_{i=1}^d \left[\frac{h_i}{2} (\theta_i - \theta_i^*)^2 + \lambda |\theta_i| \right] + \text{const}$ so we need to minimize $f(a) = \frac{1}{2}(a - b)^2 + \beta |a|$ and this gives $a^* = \text{sgn}(b) \max\{0, |b| - \beta\}$, so $\theta_i^* = \text{sgn}(\theta_i) \max\{0, |\theta_i| - \frac{\lambda}{h_i}\}$. For L^2 regularization we get $\theta_i^* = \frac{h_i}{h_i + \lambda} \theta_i^*$. **Connecting L2 w/ early stopping:** $R(w) \approx R(w^*) + \frac{1}{2}(w - w^*)^\top H(w - w^*)$. $\nabla R(w) = H(w - w^*)$. GD update: $w' = w^{t-1} - \eta H(w^{t-1} - w^*)$ gives $w^t - w^* = (I_d - \eta H)(w^{t-1} - w^*)$. Using $H = Q \Lambda Q^\top$: $Q^\top (w^t - w^*) = (I_d - \eta \Lambda) Q^\top (w^{t-1} - w^*)$. If $w^0 = 0$, $Q^\top (w^t - w^*) = (I_d - \eta \Lambda)^t Q^\top (0 - w^*) \Rightarrow Q^\top w^t = [I_d - (I_d - \eta \Lambda)^t] Q^\top w^*$. Optimal w under L^2 reg gives $Q^\top w = [I_d - \lambda(\Lambda + \lambda I_d)^{-1}] Q^\top w^*$. Matching both gives $t \approx \frac{1}{\lambda}$. **Weight normalization** is like Batch-Norm, with the covariance matrix replaced by the identity matrix.

8.2 Ensemble Methods

Bagging: Create K bootstrap samples of Data (sampling with replacement), train separate models, and average predictions: $p(y|x) = \frac{1}{K} \sum_{k=1}^K p(y|x; \theta_k)$

Dropout: Randomly drop units during training with probability $1 - \pi$. Creates an exponential ensemble of sub-networks sharing weights. Test time: Scale weights by π to approximate the ensemble average.

8.3 Normalization

Batch Norm: Normalize activations across mini-batch: $\tilde{z} = \frac{z - \mu_{\text{batch}}}{\sigma_{\text{batch}}}$, $\hat{z} = \alpha \tilde{z} + \beta$, $\mu_{\text{batch}} = \frac{1}{b} \sum_{i=1}^b z_i$, $\sigma_{\text{batch}} = \sqrt{\frac{1}{b} \sum_{i=1}^b (z_i - \mu_{\text{batch}})^2}$.

Layer Norm: Normalize features in a layer instead; particularly effective for RNNs (batch statistics are less stable).

8.4 Data / Task Augmentation

Augment Data by applying valid transformations.

Semi-supervised Learning: Train jointly on labeled and unlabeled data w/ combined loss. Pre-training & Fine-tuning. Multi-task Learning. Self-supervised Learning: Create free supervision from data.

9 Recurrent Neural Networks

Evolution: $z_t = F[\theta](z_{t-1}, x_t)$, with $z_0 = 0$. Optional output: $y_t = G[\theta](z_t)$.

Simple RNN: $z_t = \phi(W z_{t-1} + U x_t)$ where $W \in \mathbb{R}^{m \times m}$, $U \in \mathbb{R}^{m \times n}$

Backpropagation Through Time (param sharing): $\frac{\partial R}{\partial w_{ij}} = \sum_t \frac{\partial R}{\partial z_t^i} \cdot \dot{\phi}_t^i \cdot z_{t-1}^{t-1}$.
 $\frac{\partial R}{\partial u_{ik}} = \sum_t \frac{\partial R}{\partial z_t^i} \cdot \dot{\phi}_t^i \cdot x_t^k \cdot \dot{\phi}_t^i = \phi'(F_i(z^{t-1}, x^t))$.

Gradient flow backward through time:

$$\nabla_{x_t} \mathcal{R} = [\prod_{r=t+1}^s W^\top S(z^r)] \cdot J_G \cdot \nabla_y \mathcal{R}$$

Spectral analysis: $\|\prod W^\top S(z^r)\|_2 \leq [\sigma_{\max}(W)]^{s-t}$

Root cause: Repeated matmul through time.
 \Rightarrow Simple RNNs cannot learn long dependencies.

Deep RNNs stack layers vertically:

$$z^{t,\ell} = \varphi(W_\ell z^{t-1,\ell} + U_\ell z^{t,\ell-1}) \text{ where } z^{t,0} = x_t.$$

For RNN with $z_{t+1} = \varphi(U z_t + V x_{t+1})$, $L = \sum_{t=1}^T \ell(\hat{y}_t, y_t)$, where \hat{y}_t depends on z_t . Then $\frac{\partial L}{\partial U} = \sum_{t=1}^T \frac{\partial L}{\partial z_t} \cdot (\varphi'_t \cdot z_t)$,
 $\frac{\partial L}{\partial V} = \sum_{t=1}^T \frac{\partial L}{\partial z_t} \cdot (\varphi'_t \cdot x_{t+1})$.

Weight Sharing in RNNs (LSTM):

$$\frac{\partial L}{\partial W} = \sum_{t=1}^T \frac{\partial L}{\partial W_t}$$

Proof idea: Introduce dummy parameters $\tilde{W}_i = f(W)$ for each time step. By chain rule: $\frac{\partial L}{\partial W} = \sum_i \frac{\partial L}{\partial \tilde{W}_i} \frac{\partial \tilde{W}_i}{\partial W}$. With constraint $\tilde{W}_i = W$, we have $\frac{\partial \tilde{W}_i}{\partial W} = I$, giving the sum.

Initialization of bias in RNNs: Use 1.

9.1 Long Short-Term Memory (LSTM)

- C_t : cell state (internal memory, protected highway)
- z_t : hidden state (external output, filtered view)

$$C_t = \underbrace{\sigma(F \tilde{x}^t) \odot C_{t-1}}_{\text{forget}} + \underbrace{\sigma(G \tilde{x}^t) \odot \tanh(V \tilde{x}^t)}_{\text{input}}$$

$$z_t = \underbrace{\sigma(H \tilde{x}^t)}_{\text{output}} \odot \tanh(C_t), \text{ where } \tilde{x}^t = [x_t, z_{t-1}]$$

9.2 Gated Recurrent Unit (GRU)

Single state z_t . Input: $\tilde{x}^t = [x_t, z_{t-1}]$.

$$u_t = \sigma(U \tilde{x}^t), \quad r_t = \sigma(R \tilde{x}^t),$$

$$z_t = u_t \odot z_{t-1} + (1 - u_t) \odot \tanh(W[r_t \odot z_{t-1}, x_t])$$

Often comparable to LSTM with fewer resources. Gating creates identity paths \rightarrow better gradient flow.

9.3 Linear Recurrent Models

RNNs not parallelizable during training. LRU has linear dynamics: $z_{t+1} = A z_t + B x_t$. Diagonalize to $A = P \Lambda P^{-1}$, $\lambda_i \in \mathbb{C}$, change basis $\zeta_t = P^{-1} z_t$. Then: $\zeta_{t+1} = \Lambda \zeta_t + C x_t$. Each dimension evolves independently (no channel mixing). Compensate with expressive output: $y_t = \text{MLP}(\text{Re}(G \zeta_t))$.

Stability: Require $\max|\lambda_j| \leq 1$ (spectral radius ≤ 1).

Parameterization: $\lambda_i = \exp(-\exp(\nu_i) + i \varphi_i)$ ensures $|\lambda_i| \in (0, 1)$ automatically, $|\lambda_i| \approx 1$: Long-term memory, $|\lambda_i| \approx 0$: Short-term patterns.

Provably universal as sequence-to-sequence map.

9.3.1 Connectionist Temporal Classification

Problem: Unsegmented sequences (e.g., speech).

Solution: RNN outputs prob distribution over vocabulary at each time step. Model all

alignments with blank symbol “-”: $p(\ell|x) = \sum_{\pi \in B^{-1}(\ell)} \prod_t y_{\pi_t}$.
 \mathcal{B} removes blanks and repeated symbols.

9.4 Sequence Learning

Teacher Forcing: $p(y^t)$ depends on $y^{1:t-1}$ only through z^t , means during autoregressive generation, model doesn't see its own predictions.

Solution: Add feedback connections from y^{t-1} to z^t : $z^t = \text{RNN}(z^{t-1}, x^t, y^{t-1})$, now model conditions on its own previous predictions \rightarrow more coherent gen.

Professor Forcing: Train two networks (teacher-forced + free-running), discriminator matches hidden states \rightarrow improved generalization.

Exposure bias: Model relies on itself where inputs come from the previous output because of the non-availability of the ground truth.

Seq2Seq: Input and output sequences have different lengths: Use encoder-decoder framework.

Gradients in bi-directional RNNs are computed by making a forward and backward run, then at timestep t we combine (concatenate/add) and continue with the backpropagation. This happens at every bidirectional layer.

To initialize points uniformly on the unit ring in \mathbb{C} between circles of radii $r_2 > r_1$, sample from $\exp(\frac{1}{2} \log(u_1(r_2^2 - r_1^2) + r_1^2) + i 2\pi u_2)$ w $u_1, u_2 \sim \mathcal{U}(0, 1)$.

10 Attention and Transformers

Seq2Seq with Attention: Encoder generates hidden state sequence. Decoding RNNs output attends to encoder states and gets used as input in next step.

Attention: Learn to index, multiplicative gating to combine bottom-up and top-down information.

KV - attention map:
 $F(\xi, ((x_1, z_1), \dots, (x_s, z_s))) = [z_1, \dots, z_s]$.
 $f(\xi, (x_1, \dots, x_s))$ consisting of a query ξ (what to look for?), keys x_i (index) and values z_i .

Scaled Dot-Product Attention: $f(\xi, x) = \frac{\xi \cdot x}{\|\xi\| \cdot \|x\|}$

Multi-headed attention: $G(\xi, (x^t, z^t)_{t=1}^s) = W \begin{bmatrix} F_1(\xi, (x^t, z^t)) \\ \vdots \\ F_h(\xi, (x^t, z^t)) \end{bmatrix}$,

where

$$F_j(\xi, (x^t, z^t)) = F \left(W_j^q \xi, (W_j^k x^t, W_j^v z^t) \right).$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q K^\top}{\sqrt{d_k}}\right) V,$$

where

$$Q = X W_Q, \quad K = X W_K, \quad V = X W_V \quad \text{and}$$

$$X \in \mathbb{R}^{T \times d_{\text{model}}}, \quad W_Q, W_K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \quad W_V \in \mathbb{R}^{d_{\text{model}} \times d_v}, \quad V \in \mathbb{R}^{T \times d_v}.$$

Positional encodings necessary since no use of recurrence. Can use predefined approaches, or learned.

Self attention used in encoder, masked self-attention in decoder. Can also add cross-attention to decoder.

ELMo consists of multiple layers of 2 LSTMs working in opposite directions. Can then be used to collapse all layers $2L+1$ and train in task-specific manner.

BERT is trained on two simultaneous tasks (Masked Language Modeling and Binary Prediction) whether sentence B follows A). Has bidirectional encoder, and task-specific heads. Can do FT for various tasks.

Vision Transformers split images into patches, add pos embeddings and a [CLS] token, then process with a standard transformer encoder.

Complexity metrics for different layer types in terms of input sequence length n .

Key Metrics:

- Complexity per Layer:** Total computational operations per layer
- Sequential Operations:** Number of operations needed to connect any two input positions
- Maximum Path Length:** Longest path between any two input positions in the network

Self-Attention:

- Complexity per Layer: $\mathcal{O}(n^2 d)$ – quadratic in sequence length due to all-pairs attention
- Sequential Operations: $\mathcal{O}(1)$ – fully parallelizable
- Maximum Path Length: $\mathcal{O}(1)$ – direct connections between all positions

RNN:

- Complexity per Layer: $\mathcal{O}(nd^2)$ – linear in sequence length
- Sequential Operations: $\mathcal{O}(n)$ – must process sequentially
- Maximum Path Length: $\mathcal{O}(n)$ – information flows through entire sequence

Trade-off: Self-attention enables parallel processing and direct long-range connections but has quadratic complexity, while RNNs have linear complexity but require sequential processing.

11 Ethics

Adversarial examples (given $f(x) = y$ correctly):

- Untargeted: $\|\delta\| \leq \epsilon$ s.t. $f(x + \delta) \neq y$.
- Optimize $\max_{\|\delta\| \leq \epsilon} L(f(x + \delta), y)$.
- Targeted: $\|\delta\| \leq \epsilon$ s.t. $f(x + \delta) = t \neq y$.
- Optimize $\min_{\|\delta\| \leq \epsilon} L(f(x + \delta), t)$.

Linear Binary ($y \in \{-1, 1\}$, $f(x) = w^\top x + b$):

Correct: $y(w^\top x + b) > 0$.
Adf. flips when $y w^\top \delta \leq -y(w^\top x + b)$ cross hyperplane. L2 optimal: $\delta^* = -\frac{w^\top x + b}{\|w\|^2} w$, $\|\delta^*\|_2 = \frac{\|w\|_2}{\|w\|_2}$.

L_∞ optimal: $\delta = -\epsilon \text{ sign}(yw)$.

Multiclass: $f_k(x) = w_k^\top x + b_k$, use $\arg\max_k f_k(x)$.

Margin to class j : $m_j(x) = (w_j - w_j)^\top x + (b_j - b_j)$.

$f_j(x) = f_j(x) \Leftrightarrow m_j(x) = 0$. Correct if $f_j(x) > f_j(x + \delta) \forall j \neq y$, adversarial if $\exists j \neq y$ s.t. $f_j(x + \delta) < f_j(x + \delta)$. Distance to boundary: $\frac{m_j(x)}{\|w_j - w_j\|}$.

Adversarial attacks for NNs: Approximate boundary by $f(x + \delta) \approx f(x) + \nabla f(x)^\top \delta$. FGSM is a one-step L_∞ attack: $\delta = \epsilon \text{ sign}(\nabla_x L(f(x), y))$. PGD is multi-step $\delta_{t+1} = \text{Proj}_{\|\delta\| \leq \epsilon}(\delta_t + \alpha \text{ sign}(g_t))$.

Distributionally Robust Optimization:

$$\min_f \sup_{Q \in U(P)} Q_L[f(Q)], \text{ where } U \text{ means close. Can use upper bound on Wasserstein distance e.g.}$$

$$\mathbb{E}[\max_{Q \in S} L(f(x + \delta), y)].$$

Robust training min $_\delta \mathbb{E}[\max_{Q \in S} L(f(x + \delta), y)]$. Adversarial training can be viewed as robustness to distribution shift measured by Wasserstein distance.

Interpretability: Local - explain pred for specific x , Global - explain model behaviour on avg over data.

Local: Ceteris paribus (vary x_j , fix x_{-j}). Sensitivity ($\partial_{x_j} f(x)$, missing info ($f(x) - \mathbb{E}[f(X) | X_{-j} = x_{-j}]$)). **Global:** Mutual info ($I(X_j; Y | X_{-j})$)). Predictive util (train $f(w)$ and $w \cdot x_j$). For log-loss predictive util \approx conditional mutual information. SHAP attributes predictions, while SAGE attributes risk reduction.

A protected attribute, Y target outcome, \hat{Y} prediction. Demographic Parity: $\hat{Y} \perp A$; Equalized Odds: $\hat{Y} \perp A | Y$, Equality of Opportunity: $\hat{Y} \perp A | Y = 1$.