

1. History

1.1 Perceptron

Threshold Unit

$f[w, b](x) = \text{sign}(x \cdot w + b)$

where $x \cdot w := \sum_{i=1}^n x_i w_i$

Decision Boundary

$x \cdot w + b \neq 0 \Leftrightarrow \frac{x \cdot w}{\|w\|} + \frac{b}{\|w\|} \neq 0$

$x \cdot w + b = \begin{bmatrix} x \\ 1 \end{bmatrix} \cdot \begin{bmatrix} w \\ b \end{bmatrix} =: \tilde{x} \cdot \tilde{w}, \quad \tilde{x}, \tilde{w} \in \mathbb{R}^{n+1}$

Geometric Margin

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

Maximum Margin Classifier

$(w^*, b^*) \in \operatorname{argmax}_{w, b} \gamma[w, b](S)$

with $\gamma[w, b](S) := \min_{(x, y) \in S} \gamma[w, b](x, y)$

Perceptron Learning

if $f[w, b](x) \neq y$

update $w \leftarrow w + yx$, and $b \leftarrow b + y$

$w_0 \in (x_1, \dots, x_s) \Rightarrow w_t \in (x_1, \dots, x_s) \ (\forall t)$

Convergence

$\exists w, \|w\| = 1$, that $\gamma[w](S) = \gamma > 0 \Rightarrow w_t \cdot w \geq t\gamma$.

$R = \max_{x \in S} \|x\| \Rightarrow \|w_t\| \leq R\sqrt{t}$

$\cos \angle(u, w_t) = \frac{u \cdot w_t}{\|w_t\|} \geq \frac{\sqrt{t}\gamma}{tR} = \frac{\gamma}{R\sqrt{t}} \leq 1 \Rightarrow t \leq \frac{R^2}{\gamma^2}$

Covers Theorem

$C(s + 1, n) = 2 \sum_{i=0}^{n-1} \binom{s}{i}$

$C(S, n)$: Number of ways to separate S with n dimensions

$C(s, n) = 2s$ for $s \leq n$

Phase transition at $s = 2n$. For $s > 2n$ empty version space is the exception, otherwise the rule.

1.3 Hopfield Networks

Hopfield Model

$E(X) = -\frac{1}{2} \sum_{i \neq j} w_{ij} X_i X_j + \sum_i b_i X_i$

where $X_i \in \{-1, +1\}$

$w_{ij} = w_{ji} \ (\forall i, j)$, $w_{ii} = 0 \ (\forall i)$: Interaction strengths

Hebbian Learning

$x^t \in \{\pm 1\}^n \quad (1 \leq t \leq s)$

$w_{ij} = \frac{1}{n} \sum_{t=1}^s x_i^t x_j^t$

$W = \frac{1}{n} \sum_{t=1}^s x^t (x^t)^\top$

2. Feedforward Networks

2.1 Linear Models

Linear regression

$h[w](S) = \frac{1}{2s} \|Xw - y\|^2$

$\nabla h = 2X^\top Xw - 2X^\top y$

Moore-Penrose inverse solution

$w^* = X^+ y \in \operatorname{argmin}_w h[w]$

where $X^+ := \lim_{\epsilon \rightarrow 0} (X^\top X + \epsilon I)^{-1} X^\top$

Stochastic gradient descent update

$w_{t+1} := w_t + \underbrace{\eta (y_{i_t} - w_t^\top x_{i_t})}_{\text{residual}} x_{i_t}$

with $i_t \stackrel{\text{iid}}{\sim} \text{Uniform}(1, \dots, s)$

Gaussian noise model

$y_i = w^\top x_i + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2)$

Least squares equivalent to negative log likelihood of gaussian noise model

Ridge regression

$h_\lambda[w] := h[w] + \frac{\lambda}{2} \|w\|^2$

$w^* = (XX^\top + \lambda I)^{-1} X^\top y$

Logistic function

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

$\sigma(z) + \sigma(-z) = 1$

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

Cross entropy loss

$\ell(y, z) = -y \log \sigma(z) - (1 - y) \log(1 - \sigma(z))$
 $= -\log \sigma((2y - 1)z)$

Logistic regression with cross entropy loss

$\nabla \ell_i = [\sigma(w^\top x_i) - y_i] x_i$

2.2 Feedforward Networks

Generic feedforward layer definition

$F: \underbrace{\mathbb{R}^{m(n+1)}}_{\text{parameters}} \times \underbrace{\mathbb{R}^n}_{\text{input}} \rightarrow \underbrace{\mathbb{R}^m}_{\text{output}}$

$F[\theta](x) := \varphi(Wx + b), \quad \theta := (\vec{W}, b)$

Composition of layers

$G = F_L[\theta_L] \circ \dots \circ F_1[\theta_1]$

where $F_l[W^l, b^l](x) := \varphi_l(W^l x + b^l)$

Layer activations

$x^l := (F_l \circ \dots \circ F_1)(x) = F_l(x^{l-1})$

identifying $x^0 = x$, $x^L = F(x)$

Softmax function

$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}, \quad \text{softmax}(A)_{ij} = \frac{e^{A_{ij}}}{\sum_k e^{A_{ik}}}$

$\ell(y; z) = \left[-zy + \log \sum_j e^{z_j} \right] \frac{1}{\ln 2}$

Residual layer definition

$F[W, b](x) = x + [\varphi(Wx + b) - \varphi(0)]$

therefore $F[0, 0] = \text{id}$

Skip connection: Concatenate previous layer back in

2.3 Sigmoid Networks

Sigmoid activation

$\varphi(z) := \sigma(z) = \frac{1}{1 + e^{-z}}$

Hyperbolic tangent activation

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

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

Barron's Theorem: Approximation error

For f with finite $C_f := \int \|\omega\| |\hat{f}(\omega)| d\omega < \infty$ there exists MLP g with one hidden layer of width m that:

$\int_B (f(x) - g_m(x))^2 \mu(dx) \leq O\left(\frac{1}{m}\right)$

2.4 ReLU Networks

ReLU activation

$\varphi(z) := (z)_+ := \max\{0, z\}$

ReLU networks are universal function approximators

Zaslavsky: Connected regions

$R(H) \leq \sum_{i=0}^{\min\{n, m\}} \binom{m}{i} := R(m)$

Montufar: Connected regions in ReLU network

$R(m, L) \geq R(m) \left\lfloor \frac{m}{n} \right\rfloor^{n(L-1)}$

L : layers, m : width

3. Gradient-Based Learning

3.1 Backpropagation

Parameter derivatives for ridge function layers

$\frac{\partial x_i^l}{\partial w_{ij}^l} = \dot{\varphi}_i^l x_j^{l-1}, \quad \dot{\varphi}_i^l := \varphi'^l \left((w_i^l)^\top x^{l-1} + b_i^l \right)$

$\frac{\partial x_i^l}{\partial b_i^l} = \dot{\varphi}_i^l$

Loss derivatives

$\frac{\partial h[\theta](x, y)}{\partial w_{ij}^l} = \frac{\partial h^l[\theta](x^l, y)}{\partial x_i^l} \frac{\partial x_i^l}{\partial w_{ij}^l} = \delta_i^l \dot{\varphi}_i^l x_j^{l-1}$

$\frac{\partial h[\theta](x, y)}{\partial b_i^l} = \frac{\partial h^l[\theta](x^l, y)}{\partial x_i^l} \frac{\partial x_i^l}{\partial b_i^l} = \delta_i^l \dot{\varphi}_i^l$

with $\delta_i^l = \frac{\partial h}{\partial x_i^l} \dot{\varphi}_i^l$

3.2 Gradient Descent

Gradient descent update

θ_{t+1} = θ_t - η ∇h(θ_t)

Gradient flow ODE

dθ/dt = -∇h(θ)

L-smoothness

||∇h(θ_1) - ∇h(θ_2)|| ≤ L ||θ_1 - θ_2|| (∀θ_1, θ_2)

λ_max(∇^2h) ≤ L

ℓ(w) - ℓ(w') ≤ ∇ℓ(w')^T (w - w') + (L/2) ||w - w'||_2^2

ℓ''(x) ≤ L

Polyak-Łojasiewicz condition

(1/2) ||∇h(θ)||^2 ≥ μ(h(θ) - min h) (∀θ)

Convergence rate

η = 1/L ⇒ t = (2L/ε^2) (h(θ^0) - min h) for ε-critical point ⇒ h(θ^t) - min h ≤ (1 - μ/L)^t (h(θ^0) - min h)

3.3 Acceleration and Adaptivity

Heavy ball momentum update

θ_{t+1} = θ_t - η ∇h(θ_t) + β(θ_t - θ_{t-1})

Nesterov acceleration

θ̃_{t+1} = θ_t + β(θ_t - θ_{t-1})

θ_{t+1} = θ̃_{t+1} - η ∇h(θ̃_{t+1})

More theoretical grounding than heavy ball

AdaGrad updates

θ^i_{t+1} = θ^i_t - η^i_i ∂_i h(θ^t)

ν^i_t = ν^i_{t-1} + [∂h/∂θ^i_t(θ^t)]^2, η^i_t = η / sqrt(ν^i_t + ε)

Adam updates

g^i_t = β g^i_{t-1} + (1 - β) ∂_i h(θ^t)

ν^i_t = α ν^i_{t-1} + (1 - α) [∂_i h(θ^t)]^2

θ^i_{t+1} = θ^i_t - η^i_t g^i_t, η^i_t := η / sqrt(ν^i_t + ε)

RMSprop: Adam without momentum term

3.4 Stochastic Gradient Descent

Stochastic gradient descent update

θ_{t+1} = θ_t - η ∇h(θ^t)(x_{i_t}, y_{i_t})

SGD variance

V[θ](S) = (1/s) ∑_{i=1}^s ||∇h[θ](S) - ∇h[θ](x_i, y_i)||^2

SGD convergence rate

E[h(θ̄^t)] - min h ≤ O((1/sqrt(t))) (general)

E[h(θ̄^t)] - min h ≤ O((log t)/t) (strongly convex)

E[h(θ̄^t)] - min h ≤ O(1/t) (additionally smooth)

3.5 Function properties

Convexity

ℓ(λw + (1 - λ)w') ≤ λℓ(w) + (1 - λ)ℓ(w')

ℓ''(x) ≥ 0 ∀x

Convexity and differentiability

ℓ(w) ≥ ℓ(w') + ∇ℓ(w')^T (w - w')

Implies convexity for differential functions and vice versa

Strong convexity and differentiability

ℓ(w) ≥ ℓ(w') + ∇ℓ(w')^T (w - w') + (μ/2) ||w - w'||_2^2

ℓ''(x) ≥ μ ∀x

4. Convolutional Networks

4.1 Convolutions

Convolution definition

(f * g)(u) := ∫_{-∞}^∞ g(u - t) f(t) dt = ∫_{-∞}^∞ f(u - t) g(t) dt

Fourier transform convolution property

F(f * g) = F(f) · F(g)

Discrete convolution

(f * g)[u] := ∑_{t=-∞}^∞ f[t] g[u - t]

Cross-correlation

(g ★ f)[u] := ∑_{t=-∞}^∞ g[t] f[u + t]

Toeplitz matrices

(f * g) = Toeplitz-Matrix(g) f

4.2 Convolutional Networks

Conventions

Padding: Add zeros around input

Stride: Step size of convolution

Max-Pooling

Take maximum value in windows (size r)

ConvNets for Images

y[r][s, t] = ∑_u ∑_{s', t'} w[r, u][s', t'] · x[u][s + s', t + t']

r: output channel, u: input channel

Number of parameters of a convolutional layer

D = (fully connected) · (window size)

4.3 Natural Language Processing with ConvNets

Word embedding

word ω ↦ x_ω ∈ R^n

Conditional log-bilinear model

Prediction of output word ν given word ω in neighborhood

P(ν|ω) = exp[x_ω^T y_ν] / ∑_μ exp[x_ω^T y_μ]

h({x_ω}, {y_ν}) = ∑_{(ω, ν)} ℓ_{ων}

ℓ_{ω, ν} = -x_ω^T y_ν + ln ∑_μ exp[x_ω^T y_μ]

Negative sampling

ℓ̃_{ω, ν} = -ln σ(x_ω^T y_ν) - β E_{μ ~ D} ln(1 - σ(x_ω^T y_μ))

5. Recurrent Networks

5.1 Simple Recurrent Networks

Time evolution equation

z_t := F[θ](z_{t-1}, x_t), z_0 := 0, (∀t)

Output map

ŷ_t := G[ψ](z_t)

RNN parameterization

F[U, V](z, x) := φ(Uz + Vx)

G[W](z) := Φ(Wz), W ∈ R^{q × m}

Backpropagation through time

∂h/∂z^s_i = ∑_{s=t}^T ∑_k δ^s_k ∑_{j=1}^m ∂ŷ^s_k/∂z^s_j ∂z^s_j/∂z^s_i, ∂ŷ^s_k/∂z^s_j = Φ^s_k w_{kj}

∂h/∂v_{ij} = ∑_{t=1}^T ∂h/∂z^t_i φ^t_i x^t_j

∂h/∂u_{ij} = ∑_{t=1}^T ∂h/∂z^t_i φ^t_i z^{t-1}_j

Spectral norm

||A||_2 = max_{x: ||x||=1} ||Ax||_2 = σ_1(A)

Gradient norms

∂z_T/∂z_0 = Φ^T U ... Φ^1 U

The norm of gradients either:

- 1. Vanishes exponentially if σ_1(U) < 1/κ: ||∂z_t/∂z_0||_2 ≤ (κσ_1(U))^t → ∞
- 2. Explodes if σ_1(U) is too large

Bidirectional RNNs

ŷ_t = Φ(Wz_t + W̃z_t)

5.2 Gated Memory

LSTM

z_t := (forget gate) ⊙ z_{t-1} + (old) (storage gate) ⊙ tanh(new) (Vx_t)

x̃_t := [x_t, ℓ_t], ℓ_{t+1} = σ(Hx̃_t) ⊙ tanh(Uz_t)

GRU

z_t = (1 - σ) ⊙ z_{t-1} + σ ⊙ z̃_t, σ := σ(G[x_t, z_{t-1}])

z̃_t := tanh(V[ℓ_t ⊙ z_{t-1}, x_t])

ℓ_t := σ(H[z_{t-1}, x_t])

5.3 Linear Recurrent Models

Linear state evolution

z_{t+1} = Az_t + Bx_t

Diagonal form

A = P\Lambda P^{-1}, \quad \Lambda := \text{diag}(\lambda_1, \dots, \lambda_m), \quad \lambda_i \in \mathbb{C}

Stability condition

\max_j |\lambda_j| \leq 1

Initialization

\lambda_i = \exp(-\exp(\xi_i) + i\theta_i), \quad e^{\xi_i} = -\ln r_i

\theta_i \sim \text{Uni}[0; 2\pi], \quad r_i \sim \text{Uni}[I], \quad I \subseteq [0; 1]

Advantages

- (i) clear modeling of long/short range dependencies
- (ii) no channel mixing required
- (iii) parallelizable training

6. Attention and Transformers

6.1 Attention

Attention mixing

y_s := \sum_t a_{st} W x_t, \quad a_{st} \geq 0, \quad \sum_t a_{st} = 1

A = (a_{st}) \in \mathbb{R}^{T \times T}, \quad \text{s.t. } Y = WXA^\top

Query-key matching

Q = U_Q X, \quad K = U_K X \quad (U_Q, U_K \in \mathbb{R}^{q \times n})

Q^\top K = X^\top \underbrace{U_Q^\top U_K}_{\text{rank} \leq q} X \quad (Q^\top K \in \mathbb{R}^{T \times T})

Softmax attention

A = \text{softmax}(\beta Q^\top K), \quad a_{st} = \frac{e^{\beta [Q^\top K]_{st}}}{\sum_r e^{\beta [Q^\top K]_{sr}}}

usually $\beta = 1/\sqrt{q}$

Feature transformation

X \mapsto Y \mapsto F(Y), \quad F(\theta)(Y) = (F(y_1), \dots, F(y_T))

Positional encoding

p_{tk} = \begin{cases} \sin(t\omega_k) & k \text{ even} \\ \cos(t\omega_k) & k \text{ odd} \end{cases}, \quad \omega_k = C^{k/K}

Transformer architecture

Self-attention: attend to its own values in the past

Cross-attention: E.g. decoder attends to encoder output (query from decoder, key and value from encoder)

Vision transformer patch embedding

\mathbb{R}^{p \times p \times q} \ni \text{patch}_t \mapsto x_t := V(\vec{\text{patch}}_t) \in \mathbb{R}^n

with $V \in \mathbb{R}^{n \times (qp^2)}$

GELU activation

\varphi(z) = z \Pr(z \leq Z), \quad Z \sim \mathcal{N}(0, 1)

7. Geometric Deep Learning

7.1 Sets and Points

Function over sets

\{x_1, \dots, x_M\} \subseteq \mathbb{R}, \quad f: 2^{\mathbb{R}} \rightarrow Y

Order-invariance property

f(x_1, \dots, x_M) = f(x_{\pi_1}, \dots, x_{\pi_M}) \quad \forall \pi \in S_M

Equivariance property

f(x_1, \dots, x_M) = (y_1, \dots, y_M)

\Rightarrow f(x_{\pi_1}, \dots, x_{\pi_M}) = (y_{\pi_1}, \dots, y_{\pi_M})

Permutation invariant sum

\sum_{m=1}^M x_m = \sum_{m=1}^M x_{\pi_m}, \quad \forall M, \forall \pi \in S_M

Deep Sets model

f(x_1, \dots, x_M) = f^l \left(\sum_{m=1}^M \varphi(x_m) \right)

Max pooling variant

f(x_1, \dots, x_M) = f^l \left(\max_{m=1}^M \varphi(x_m) \right)

Equivariant map construction

f^l: \mathbb{R} \times \mathbb{R}^N \rightarrow Y, \quad \left(x_m, \sum_{k=1}^M \varphi(x_k) \right) \mapsto y_m

7.2 Graph Convolutional Networks

Feature and adjacency matrices

X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_M^\top \end{bmatrix}, \quad A = (a_{nm})

\text{with } a_{nm} = \begin{cases} 1 & \text{if } \{v_n, v_m\} \in E \\ 0 & \text{otherwise} \end{cases}

Permutation matrix constraints

P \in \{0, 1\}^{M \times M} \quad \text{s.t.} \quad \sum_{n=1}^M p_{nm} = \sum_{n=1}^M p_{mn} = 1 \quad (\forall m)

Graph invariance definition

f(X, A) \stackrel{!}{=} f(PX, PAP^\top), \quad \forall P \in \mathcal{M}

Graph equivariance definition

f(X, A) \stackrel{!}{=} Pf(PX, PAP^\top), \quad \forall P \in \mathcal{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 \left(x_m, \bigoplus_{X_m} \Phi(x) \right)

\oplus is a permutation-invariant operation

Normalized adjacency matrix

\bar{A} = D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}

D = \text{diag}(d_1, \dots, d_M), \quad d_m = 1 + \sum_{n=1}^M a_{nm}

GCN layer

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

Two-layer GCN

Y = \text{softmax}(\bar{A}\sigma(\bar{A}XW^0)W^1)

7.3 Spectral Graph Theory

Laplacian operator

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

Graph Laplacian

L = D - A, \quad (Lx)_n = \sum_{m=1}^M a_{nm}(x_n - x_m)

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

\Lambda := \text{diag}(\lambda_1, \dots, \lambda_M), \quad \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 \left(\sum_{k=0}^K \alpha_k \Lambda^k \right) U^\top = \sum_{k=0}^K \alpha_k L^k

Polynomial kernel network layer

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

7.4 Attention GNNs

Attention coupling matrix

Q = (q_{ij}), \quad q_{ij} = \text{softmax}(f^l(u^\top(Vx_i; Vx_j; x_{ij})))

s.t. $\sum_j A_{ij}q_{ij} = 1$

Attention propagation

X^+ = \sigma(QXW)

Weisfeiler-Lehman test

8. Tricks of the Trade

8.1 Initialization

Random initialization

$$\theta_i^0 \sim \mathcal{N}(0, \sigma_i^2), \quad \text{or} \\ \theta_i^0 \sim \text{Uniform}[-\sqrt{3}\sigma_i; \sqrt{3}\sigma_i]$$

LeCun initialization

$$w_{ij} \stackrel{\text{iid}}{\sim} \text{Uniform}[-a; a], \quad a := 1/\sqrt{n}, \quad b_i = 0$$

Stabilizes variance

Glorot initialization

$$w_{ij} \stackrel{\text{iid}}{\sim} \text{Uniform}[-\sqrt{3}\epsilon; \sqrt{3}\epsilon], \quad \epsilon := \frac{2}{n+m}$$

Stabilizes variance of gradients in backpropagation

He initialization

$$w_{ij} \sim \mathcal{N}(0, \epsilon) \quad \text{or} \\ w_{ij} \sim \text{Uniform}[-\sqrt{3}\epsilon; \sqrt{3}\epsilon], \quad \epsilon := \frac{2}{n}$$

In ReLU networks typically only $n/2$ units active

Orthogonal initialization

$$\frac{1}{\sqrt{m}}W \sim \text{Uniform}(O(m))$$

s.t. $W^\top W = WW^\top = mI$

8.2 Weight Decay

L_2 regularization

$$\mu(\theta) = \frac{\mu}{2} \|\theta\|^2, \quad \mu \geq 0$$

Gradient descent with weight decay

$$\dot{\theta} = -\eta \nabla E(\theta) - \eta \nabla \mu(\theta) = -\eta \nabla E(\theta) - \eta \mu \theta$$

Weight decay for multiple layers

$$\theta = (\vec{W}^1), (\vec{W}^2), \dots, (\vec{W}^L)$$

$$\mu(\theta) = \sum_{l=1}^L \mu_l \|W^l\|_F^2$$

Local loss landscape

$$\theta_\mu^* = (H + \mu I)^{-1} H \theta^*, \quad H = Q \Lambda Q^\top$$

$$(\Lambda + I)^{-1} = \text{diag} \left(\frac{\lambda_i}{\lambda_i + \mu} \right)$$

The minimum θ^* is shrunk along directions with small eigenvalues

Generalization

$$\mu = \frac{\sigma^2}{u^2}, \quad u: \text{teacher signal}$$

Optimal weight decay inverse proportional to the signal-to-noise ratio

8.4 Dropout

Probability ϕ_i of keeping a unit

Dropout as Ensembling

$$p(y|x) = \sum_{b \in \{0,1\}^R} p(b) p(y|x; b)$$

with $p(b) = \prod_{i=1}^R \phi_i^{b_i} (1 - \phi_i)^{1-b_i}$

Weight scaling for inference

$$\tilde{w}_{ij} \leftarrow \phi_{ij} w_{ij}$$

8.5 Normalization

Batch normalization

\mathbb{E} and \mathbb{V} from minibatches or population statistics

$$\bar{f} = \frac{f - \mathbb{E}[f]}{\sqrt{\mathbb{V}[f]}}, \quad \mathbb{E}[\bar{f}] = 0, \quad \mathbb{V}[\bar{f}] = 1$$

$$\bar{f}[\mu, \epsilon] = \mu + \epsilon \bar{f}$$

Weight normalization

$$f(v, \epsilon)(x) = \varphi(w^\top x), \quad w := \frac{\epsilon}{\|v\|_2} v$$

Gradient descent with respect to decoupled ϵ and v :

$$\partial_\epsilon E = \nabla_w E \cdot \frac{v}{\|v\|_2}$$

$$\nabla_v E = \frac{\epsilon}{\|v\|} \left(I - \frac{w w^\top}{\|w\|_2^2} \right) \nabla_w E$$

Layer normalization

$$\tilde{f}_i = \frac{f_i - \mathbb{E}[f]}{\sqrt{\mathbb{V}[\tilde{f}]}}$$

$$\mathbb{E}[f] = \frac{1}{m} \sum_{i=1}^m f_i$$

$$\mathbb{V}[f] = \frac{1}{m} \sum_{i=1}^m (f_i - \mathbb{E}[f])^2$$

Using population averages across units in a layer

8.7 Model Distillation

Tempered cross entropy loss for distillation

$$\ell(x) = \sum_{y=1}^K \frac{q \exp[F_y(x)/T]}{\sum_{\nu=1}^K \exp[F_\nu(x)/T]} \left[\frac{1}{T} G_y(x) - \ln \sum_{\nu=1}^K \exp[G_\nu(x)/T] \right]$$

$T > 0$, F_y : teacher logits, G_y : student logits

Gradient of distillation loss

$$\frac{\partial \ell}{\partial G_y} = \frac{1}{T} \left[\frac{e^{q F_y/T}}{\sum_{\nu} e^{F_\nu/T}} - \frac{q e^{G_y/T}}{\sum_{\nu} e^{G_\nu/T}} \right]$$

9. Theory

9.1 Neural Tangent Kernel

Linearized DNN taylor approximation

$$h(\vartheta)(x) = f(x) + \vartheta \cdot \nabla f(x)$$

with $\vartheta \approx \theta - \theta_0$, $f(x) := f(\theta_0)(x)$

Kernel of gradient feature maps

$$k(x, y) = \nabla f(x) \cdot \nabla f(y), \quad \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$$

Dual representation

$$h(\alpha)(x) = f(x) + \sum_{i=1}^s \alpha_i \nabla f(x_i) \cdot \nabla f(x)$$

Squared loss

$$E(\alpha) = \frac{1}{2s} \sum_{i=1}^s \left[\sum_{j=1}^s \alpha_j \nabla f(x_j) \cdot \nabla f(x_i) + f(x_i) - y_i \right]^2$$

Optimal solution of linearized DNN

$$K = [k(x_i, x_j)]_{i,j=1}^n \in \mathbb{R}^{n \times n}$$

$$\alpha^* = K^+(y - f)$$

$$h^*(x) = k(x) K^+(y - f)$$

Neural Tangent Kernel NTK

$$k(\theta)(x, y) := \nabla f(\theta)(x) \cdot \nabla f(\theta)(y)$$

Quadratic loss

$$E(\theta) = \frac{1}{2} \|f(\theta) - y\|^2, \quad y := (y_1, \dots, y_s)^\top$$

Gradient flow ODE

$$\dot{\theta} := \frac{d\theta}{dt} = \sum_{i=1}^s (y_i - f_i(\theta)) \nabla f_i(\theta)$$

Functional gradient flow

$$\dot{f}_j = \nabla f_j \cdot \dot{\theta} = \sum_{i=1}^s (y_i - f_i) k(\theta)(x_i, x_j)$$

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

Infinite width limit

$$w_{ij}^l = \sqrt{\frac{\sigma_w}{m_l}} \xi_{ij}^l, \quad b_i^l = \sqrt{\frac{\sigma_b}{m_l}} \eta_i^l, \quad \xi_{ij}^l, \eta_i^l \stackrel{\text{iid}}{\sim} \mathcal{N}(0, 1)$$

$$k(\theta) \xrightarrow{P} k_\infty \text{ for } m_l \rightarrow \infty$$

Initial NTK converges to deterministic limit

NTK constancy

$$\frac{dk(\theta(t))}{dt} = 0$$

$$f_\infty(x) = k(x) K^+(y - f), \quad k = k_\infty$$

NTK remains constant when training in infinite width limit

Vanishing curvature

$$\frac{\|\nabla^2 f(\theta_0)\|_2}{\|\nabla f(\theta_0)\|_2^2} \ll 1$$

Near-constancy

$$\|k(\theta_0) - k(\theta_t)\|_F^2 \in O(1/m), \quad m = m_1 = \dots = m_L$$

9.2 Bayesian DNNs

Bayesian predictive distribution

f(x) = \int f(\theta)(x)p(\theta|S)d\theta

Bayes rule

p(\theta|S) = \frac{p(\theta)p(S|\theta)}{p(S)}, \quad p(S) = \int p(\theta)p(S|\theta)d\theta

Parameter priors (Gaussian)

p(\theta) = \prod_{i=1}^d p(\theta_i), \quad \theta_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \sigma_i^2)

-\log p(\theta) = \frac{1}{2\sigma^2} \|\theta\|^2 + \text{const}

Essentially a weight decay term

Likelihood (Gaussian noise)

-\log p(S|\theta) = \frac{1}{2\epsilon^2} \|y - f(\theta)\|^2 + \text{const.}

with y_i = f^*(x_i) + \eta_i, \eta_i \stackrel{\text{iid}}{\sim} \mathcal{N}(0, \epsilon^2)

Posterior

-\log p(\theta|S) = E(\theta) + \text{const}

E(\theta) = \frac{1}{2\epsilon^2} \|y - f\|^2 + \frac{1}{2\sigma^2} \|\theta\|^2

Bayesian ensembling (post hoc)

f(Y)(x) = \sum_{i=1}^n \frac{\exp[-E(\theta_i^{(Y)})]}{\sum_{j=1}^n \exp[-E(\theta_j^{(Y)})]} f(\theta_i)(x)

Relative posterior weighting

Markov chain monte carlo (MCMC)

\theta_0, \theta_1, \theta_2, \dots, \quad \theta_{t+1} | \theta_t \sim \Pi

p(\theta_1 | S) \Pi(\theta_2 | \theta_1) = p(\theta_2 | S) \Pi(\theta_1 | \theta_2)

Metropolis-Hastings

\Pi(\theta_1 | \theta_2) = \tilde{\Pi}(\theta_1 | \theta_2) A(\theta_1 | \theta_2)

A(\theta_1 | \theta_2) = \min \left\{ 1, \frac{p(\theta_1 | S) \tilde{\Pi}(\theta_2 | \theta_1)}{p(\theta_2 | S) \tilde{\Pi}(\theta_1 | \theta_2)} \right\}

Modified transition probability with acceptance step A

Hamiltonian monte carlo

E(\theta) = - \sum_{x,y} \log p(y|x;\theta) - \log p(\theta)

H(\theta, v) = E(\theta) + \frac{1}{2} v^\top M^{-1} v

with p(\theta, v) \propto \exp[-H(\theta, v)]

\dot{v} = -E(\theta), \quad \dot{\theta} = v

\theta_{t+1} = \theta_t + \eta v_t

v_{t+1} = v_t - \eta \nabla E(\theta^t)

Langevin dynamics

\dot{\theta} = v

dv = -\nabla E(\theta) dt - \underbrace{B v dt}_{\text{friction}} + \underbrace{\mathcal{N}(0, 2B dt)}_{\text{noise process}}

\theta_{t+1} = \theta_t + \eta v_t

v_{t+1} = \underbrace{(1 - \eta\beta)v_t}_{\text{friction}} - \underbrace{\eta \nabla \tilde{E}(\theta)}_{\text{stochastic}} + \underbrace{\sqrt{2\beta\eta} \mathcal{N}(0, I)}_{\text{extra noise}}

9.3 Gaussian Processes

Gaussian process

(f(x_1), \dots, f(x_s)) \sim \mathcal{N} \Leftrightarrow \sum_{i=1}^s \alpha_i f(x_i) \sim \mathcal{N}, \quad \forall \alpha \in \mathbb{R}^s

Mean and covariance functions

GPs are completely defined by first and second order statistics

\mu(x) := \mathbb{E}_x[f(x)]

k(x, y) := \mathbb{E}_{x,y}[f(x)f(y)] - \mu(x)\mu(y)

K_{\nu\mu} = k(x_\nu, x_\mu), \quad K \in \mathbb{R}^{s \times s}

Example kernels

k(x, y) = x^\top y, \quad k(x, y) = e^{-\epsilon \|x - y\|^2}

GPs in DNN

Treating parameters as random variables. Each unit in a DNN becomes a random function.

Linear Layer

w \sim \mathcal{N}\left(0, \frac{\sigma^2}{n} I_{n \times n}\right)

\mathbb{E}[y_i y_j] = \frac{\sigma^2}{n} x_i^\top x_j

Deep layers

W^{l+1} X^l, \quad l \geq 1

No longer normal as products break normality, but near-normal for high dimensional inputs.

Non-linear activations

\mu(x^{l+1}) = \mathbb{E}[\varphi(W^l x^l)]

Kernel recursion

K_{\mu\nu}^l = \mathbb{E}[\varphi(x_i^{l-1}(\mu))\varphi(x_i^{l-1}(\nu))] = \sigma^2 \mathbb{E}[\varphi(f_\mu)\varphi(f_\nu)]

f \sim \text{GP}(0, K^{l-1})

Kernel regression

Mean of bayesian predictive distribution

f^*(x) = k(x)^\top K^+ y

\mathbb{E}[(f(x) - f^*(x))^2] = K(x, x) - k(x)^\top K^+ k(x)

9.4 Statistical Learning Theory

VC learning theory

L_t = - \frac{\|m(x_t, x_{0:t}) - m_\theta(x_t, t)\|^2}{2\sigma_t^2} + \text{const.}

\text{VC-dim}(\mathcal{F}) := \max_s \sup_{|S|=s} \mathbb{1}[|\mathcal{F}(S)| = 2^s]

VC inequality

P\left(\sup_F |E(f) - \hat{E}(f)| > \epsilon\right) \leq 8|\mathcal{F}(s)|e^{-s\epsilon^2/32}

Double descent

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

Generalization gap

\Delta := \max(0, E - \hat{E})

E: expected population error, \hat{E}: empirical error

KL divergence

D_{KL}(p\|q) = \int p(x) \log \frac{p(x)}{q(x)} dx = \mathbb{E}_{x \sim p} \left[\ln \frac{p(x)}{q(x)} \right]

PAC-Bayesian theorem

For fixed E and any Q over s samples:

\mathbb{E}_Q[E(f)] - \mathbb{E}_Q[\hat{E}(f)] \leq \sqrt{\frac{2}{s} \left[KL(Q\|P) + \ln \frac{1}{2} \sqrt{\frac{s}{\delta}} \right]}

Ensures general rate \tilde{O}(1/\sqrt{s})

PAC-Bayesian bound

Q := \mathcal{N}(\theta, \text{diag}(\sigma_i^2))

KL(Q\|P) = \sum_i \log \frac{\lambda}{\sigma_i} + \frac{\sigma_i^2 + \theta_i^2}{2\lambda^2} - \frac{1}{2}

E_{PAC}(Q) := \mathbb{E}_Q[\hat{E}] + \sqrt{\frac{2}{s} \left[KL(Q\|P) + \ln \frac{1}{2} \sqrt{\frac{s}{\delta}} \right]}

Favours minima robust to parameter perturbations

PAC-bayesian learning implementation

\theta_{t+1} = \theta_t - \eta \nabla \mathbb{E}_Q[\hat{E}] = \theta_t - \eta \nabla \hat{E}(\tilde{\theta})

with \tilde{\theta} \sim Q(\theta, \sigma)

Gradient loss on perturbed parameters

Reparameterization trick

\tilde{\theta} = \theta + \text{diag}(\sigma_i)\eta, \quad \eta \sim \mathcal{N}(0, I)

Backpropagation to \theta and \sigma_i

10. Generative Models

10.1 Variational Autoencoders

Linear autoencoder

$$\begin{aligned}x &\mapsto z = Cx, & C &\in \mathbb{R}^{m \times n} \\z &\mapsto \hat{x} = Dz, & D &\in \mathbb{R}^{n \times m} \\E(C, D)(x) &= \frac{1}{2} \|x - \hat{x}\|^2 = \frac{1}{2} \|x - DCx\|^2 \\DCX &= \hat{X} = U_m \Sigma_m V^\top \\ \Sigma_m &= \text{diag}(\sigma_1, \dots, \sigma_m, 0, \dots, 0)\end{aligned}$$

For centered data equivalent to PCA, but generally has non-global minima

Linear factor analysis

Probability Model

$$p_X(x) = \int p_Z(z) p_{X|Z}(x|z) dz$$

Z: latent variables, X: observed variables

Linear observation model

$$\begin{aligned}x &= \mu + Wz + \eta \quad \text{with } \eta \sim \mathcal{N}(0, \Psi) \\x &\sim \mathcal{N}(\mu, WW^\top + \Psi) \text{ for } z \sim \mathcal{N}(0, I)\end{aligned}$$

Posterior mean and covariance

$$\begin{aligned}\mu_{z|x} &= W^\top(WW^\top + \Psi)^{-1}(x - \mu) \\ \Sigma_{z|x} &= I - W^\top(WW^\top + \Psi)^{-1}W\end{aligned}$$

Pseudoinverse limit

$$\begin{aligned}W^\top(WW^\top + \sigma^2 I)^{-1} &\rightarrow W^+ \in \mathbb{R}^{m \times n} \\ \mu_{z|x} &\rightarrow W^+(x - \mu), \quad \Sigma_{z|x} \rightarrow 0\end{aligned}$$

Maximum likelihood estimation

$$\mu, W \leftarrow \underset{\mu, W}{\operatorname{argmax}} \log p_{(\mu, W)}(S)$$

Optimality condition for W

$$w_i = \phi_i u_i, \quad \phi_i = \max\{0, \sqrt{\lambda_i - \sigma^2}\}$$

With (λ_i, u_i) eigenvalues and eigenvectors of covariance matrix. For $\sigma = 0$ equivalent to PCA.

Variational autoencoder (VAE)

$$\begin{aligned}z &\sim \mathcal{N}(0, I) \\ x &= F(\theta)(z) = (F_L \circ \dots \circ F_1)(z)\end{aligned}$$

Evidence lower bound (ELBO)

$$\begin{aligned}\log p(\theta)(x) &= \log \int p(\theta)(x|z) p(z) dz \\ &= \log \int q(z) \left[\frac{p(\theta)(x|z) p(z)}{q(z)} \right] dz \\ &\geq \int q(z) \log p(\theta)(x|z) dz - \underbrace{\int q(z) \log \frac{q(z)}{p(z)} dz}_{=D_{KL}(q\|p)} \\ &=: L(\theta, q)(x)\end{aligned}$$

Inference network

$$\begin{aligned}\theta &\xrightarrow{\max} L(\theta, q)(S) = \sum_{i=1}^s L(\theta, q)(x_i) \\ z &\sim \mathcal{N}(\mu(x), \Sigma(x)) \\ z &= \mu + \Sigma^{\frac{1}{2}} \eta, \quad \eta \sim \mathcal{N}(0, I) \\ \nabla_\mu \mathbb{E}[f(z)] &= \mathbb{E}[\nabla_z f(z)] \\ \nabla_\Sigma \mathbb{E}[f(z)] &= \frac{1}{2} \mathbb{E}[\nabla_z^2 f(z)]\end{aligned}$$

Integration by parts derivation

$$\begin{aligned}\nabla_\mu \mathbb{E}[f(z)] &= - \int \nabla_z \mathcal{N}(\mu, \Sigma) f(z) dz \\ &= \int \mathcal{N}(\mu, \Sigma) \nabla_z f(z) dz\end{aligned}$$

10.2 Generative Adversarial Networks

GAN objective

$$V(G, D) = \mathbb{E}_{x_r \sim p_{\text{data}}} D(x_r) + \mathbb{E}_{z \sim p_z} (1 - D(G(z)))$$

Discriminator Mixture Model

$$\tilde{p}_\theta(x, y) = \frac{1}{2} (yp(x) + (1-y)p_\theta(x)), \quad y \in \{0, 1\}$$

p: true probability, p_θ: model probability

Bayes-optimal classifier

$$q_\theta(x) := P\{y = 1|x\} = \frac{p(x)}{p(x) + p_\theta(x)}$$

To detect fake samples, y = 1 for real samples, y = 0 for fake samples

Logistic likelihood

$$\theta \xrightarrow{\min} \ell^*(\theta) := \mathbb{E}_{\tilde{p}_\theta} [y \ln q_\theta(x) + (1-y) \ln(1 - q_\theta(x))]$$

Jensen-Shannon as effective objective

$$\begin{aligned}\ell^* &= \mathbb{E}_{\tilde{p}_\theta} [y \ln q_\theta(x) + (1-y) \ln(1 - q_\theta(x))] \\ &= -\frac{1}{2} H(p) - \frac{1}{2} H(p_\theta) + H\left(\frac{1}{2}(p + p_\theta)\right) - \ln 2 \\ &= \text{JS}(p, p_\theta) - \ln 2\end{aligned}$$

Discriminator model

$$q_\psi : x \mapsto [0; 1], \quad \psi \in \Psi$$

Objective bounds

$$\begin{aligned}\ell^*(\theta) &\geq \sup_{\psi \in \Psi} \ell(\theta, \psi) \\ \ell(\theta, \psi) &:= \mathbb{E}_{\tilde{p}_\theta} [y \ln q_\psi(x) + (1-y) \ln(1 - q_\psi(x))]\end{aligned}$$

Saddle point optimization

$$\theta^* := \underset{\theta \in \Theta}{\operatorname{argmin}} \left\{ \sup_{\psi \in \Psi} \ell(\theta, \psi) \right\}$$

ψ: Generator, θ: Discriminator

Alternating gradient descent/ascent

$$\begin{aligned}\theta_{t+1} &= \theta_t - \eta \nabla_\theta \ell(\theta_t, \psi_t) \\ \psi_{t+1} &= \psi_t + \eta \nabla_\psi \ell(\theta_{t+1}, \psi_t)\end{aligned}$$

Extra-gradient steps

$$\begin{aligned}\theta_{t+1} &= \theta_t - \eta \nabla_\theta \ell(\theta_{t+0.5}, \psi_t) \\ \text{with } \theta_{t+0.5} &:= \theta_t - \eta \nabla_\theta \ell(\theta_t, \psi_t) \\ \psi_{t+1} &= \psi_t + \eta \nabla_\psi \ell(\theta_t, \psi_{t+0.5}) \\ \text{with } \psi_{t+0.5} &:= \psi_t + \eta \nabla_\psi \ell(\theta_t, \psi_t) \\ \text{Deconvolutional DNN} \\ \text{Upside-down ConvNet for image generation}\end{aligned}$$

10.3 Denoising Diffusion

Markov chains

$$\begin{aligned}x_{0:t-1} &\perp\!\!\!\perp x_{t+1:\infty} | x_t \quad (\forall t) \\ p(x_t | x_{t-1}) &= p(x_1 | x_0) \quad (\forall t)\end{aligned}$$

$$p(x_{s:t}) = p(x_t) \prod_{\tau=s+1}^t p(x_{\tau-1} | x_\tau)$$

$$p(x_{s:t}) = p(x_s) \prod_{\tau=s+1}^t p(x_\tau | x_{\tau-1})$$

$$\pi(x_{t+1}) = \int \pi(x_t) p(x_{t+1} | x_t) dx_t$$

Denoising diffusion

Forward (noise generation)

$$\pi^* = \xi_0 \mapsto \xi_1 \mapsto \dots \mapsto \xi_{T-1} \mapsto \xi_T = \pi$$

Backward (denoising)

$$\pi = \mu_\theta^T \mapsto \mu_\theta^{T-1} \mapsto \dots \mapsto \mu_\theta^1 \mapsto \mu_0 \overset{?}{\approx} \pi^*$$

Gaussian example

$$\pi \approx \mathcal{N}(0, I), \quad x_t | x_{t-1} \sim \mathcal{N}((1 - \beta_t)x_{t-1}, \beta_t I)$$

Forward SDE

$$dx_t = -\frac{1}{2} \beta_t x_t dt + \sqrt{\beta_t} d\omega_t$$

Backward SDE

$$dx_t = \left[-\frac{1}{2} \beta_t x_t - \underbrace{\beta_t \nabla_{x_t} \log q_t(x_t)}_{\text{score}} \right] dt + \underbrace{\sqrt{\beta_t} d\bar{\omega}_t}_{\text{wiener process}}$$

ELBO bound

$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I)$$

$$\ln p_\theta(x_0) = \ln \int q(x_{1:T} | x_0) \frac{p_\theta(x_{0:T})}{q(x_{1:T} | x_0)} dx_{1:T}$$

$$\geq \mathbb{E} \left[\ln \frac{p_\theta(x_{0:T})}{q(x_{1:T} | x_0)} \middle| x_0 \right] = \sum_{t=0}^T L_t$$

$$L_t := \begin{cases} \mathbb{E}[\ln p_\theta(x_0 | x_1)] & t = 0 \\ -D(q(x_T | x_0) \| \pi) & t = T \\ -D(q(x_{t-1} | x_t, x_0) \| p_\theta(x_{t-1} | x_t)) & \text{else} \end{cases}$$

Backward model assumption

$$x_{t-1} | x_t \sim \mathcal{N}(m(x_t, t), \Sigma(x_t, t))$$

Entropy bounds

$$H(x_t) \geq H(x_{t-1}) \Rightarrow H(x_t | x_{t-1}) \geq H(x_{t-1} | x_t)$$

Noise schedules

$$\bar{\alpha}_t = \prod_{\tau=1}^t (1 - \beta_\tau), \quad \bar{\beta}_t = 1 - \bar{\alpha}_t$$

$$\xi_t \approx \mathcal{N}(\sqrt{\bar{\alpha}_t}x_0, \bar{\beta}_t I) \xrightarrow{t \rightarrow \infty} \mathcal{N}(0, I)$$

Forward trajectory target

$$x_{t-1}|x_t, x_0 = \mathcal{N}(m(x_t, x_0, t), \tilde{\beta}_t I)$$
$$m(x_t, x_0, t) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t}x_0 + \frac{(1 - \bar{\alpha}_{t-1})\sqrt{1 - \bar{\beta}_t}}{1 - \bar{\alpha}_t}x_t$$

with $\tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\beta_t$

Fixed isotropic covariance

$$\Sigma(x_t, t) = \sigma_t^2 I, \quad \text{where } \sigma_t^2 \in \{\beta_t, \tilde{\beta}_t\}$$

Simplified ELBO

$$L_t = -\frac{\|m(x_t, x_0, t) - m_\theta(x_t, t)\|^2}{2\sigma_t^2} + \text{const.}$$

Reparameterization

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon \Rightarrow x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}x_t - \frac{\sqrt{1 - \bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}\epsilon$$
$$m(x_t, x_0, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t(x_0, \epsilon) - \frac{\sqrt{\beta_t}}{\sqrt{1 - \bar{\alpha}_t}}\epsilon\right)$$

with $\epsilon \sim \mathcal{N}(0, I)$

Expected squared error

$$\mathbb{E}_q[L_t|x_0] = \mathbb{E}_\epsilon \left[\phi_t \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\|^2 | x_0 \right]$$

with $\phi_t = \frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)}$

Final simplified criterion

$$h(\theta)(x) = \frac{1}{T} \sum_{t=1}^T \mathbb{E}[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\|^2]$$

11. Ethics

11.1 Adversarial Examples

Adversarial perturbation

$$f(x + \delta) \neq f(x) \quad \text{s.t.} \quad \|\delta\|_p \leq \epsilon$$

p-norm definitions

$$\|x\|_p = \left(\sum_i |x_i|^p\right)^{1/p}$$

$$\|x\|_\infty = \max_i |x_i|, \quad \|x\|_0 = |\{i : x_i \neq 0\}|$$

Optimal perturbation (linear binary classification)

$$\delta \propto \text{sign}(f_1(x) - f_2(x))(w_2 - w_1)$$

for $f_i = w_i^\top x + b_i$

Optimal perturbation (multiclass)

$$\delta = \underset{i>1}{\operatorname{argmin}} \frac{f_1(x) - f_i(x)}{\|w_1 - w_i\|_2^2} (w_i - w_1)$$

DeepFool iterative optimization

Iterate:

$$\underset{\delta}{\operatorname{argmin}} \|\delta\|_2 \quad \text{s.t.} \quad (\nabla f_1(x) - \nabla f_2(x))^\top \delta < f_1(x) - f_2(x)$$

Robust training

$$\ell(f(x), y) \rightarrow \max_{\delta: \|\delta\|_p \leq \epsilon} \ell(f(x + \delta), y)$$

Projected gradient ascent (*p* = 2)

$$\delta_{t+1} = \epsilon [\delta_t + \alpha \nabla_x \ell(f(x + \delta_t), y)]_\epsilon$$

$$[z]_\epsilon := \frac{z}{\|z\|_2}$$

Projected gradient ascent (*p* = ∞)

$$\delta_{t+1} = \epsilon [\delta_t + \alpha \text{sign}(\nabla_x \ell(f(x + \delta_t), y))]_\epsilon$$

$$[z]_\epsilon := \frac{z}{\|z\|_\infty}$$

Fast Gradient Sign Method (FGSM)

$$\delta = \epsilon \text{sign}(\nabla_x \ell(f(x), y))$$