Regression: N: data-size D: dim (features) $D+1$: Model param. with offset w_0

→ relates x_n to y_n. Regression Funct°: $y_n \approx f(x_n) = f_w(x_n)$ → 2 goals: Prediction, Interpret°

Linear Regression: Univariate: $y_n \approx f(x_n) := w_0 + w_1 x_{n1}$

Multivariate: $y_n \approx f(x_n) := w_0 + w_1 x_{n1} + \dots + w_D x_{nD} = w_0 + x_n^T \begin{pmatrix} w_1 \\ \vdots \\ w_D \end{pmatrix} =: \tilde{x}_n^T \tilde{w}$

Note: If $D > N$, we say the model is overparametrized → Solution: Regularization $= \begin{pmatrix} 1 \\ x_n \end{pmatrix} \tilde{w} \in \mathbb{R}^{D+1}$

Cost Funct°: $\mathcal{L}(w) = \frac{1}{N} \sum_{n=1}^{N} \text{Loss}(e_n)$ (sum over training samples), $e_n = y_n - f_w(x_n)$ | 2 desirables pr(o)p(er)ties:

1) symmetric around 0 2) Penalize large and huge mistakes the same (outliers robust)

$MSE(w) := \frac{1}{N} \sum_{n=1}^{N} [y_n - f_w(x_n)]^2$ $MAE(w) := \frac{1}{N} \sum_{n=1}^{N} |y_n - f_w(x_n)|$ more robust to outliers than MSE

Convexity: A function $h(u)$ with $u \in \mathbb{R}^D$ is convex if for any $u, v \in \mathbb{R}^D$ and any $0 \le \lambda \le 1$: $\boxed{h(\lambda u + (1-\lambda) v) \le \lambda h(u) + (1-\lambda) h(v)}$ desirable computat-ional property

A strict. convex function has a unique global min. For convex functions, every local min is a global min

- Sum of convex functions are convex - $f: \mathbb{R}^D \to \mathbb{R}$ convex iff $\nabla^2 f$ = PSD

- Local opt: $\mathcal{L}(w^*) \underset{\text{strict}}{\le} \mathcal{L}(w), \quad \forall w \text{ with } \|w - w^*\| < \epsilon$ | Global: $\mathcal{L}(w^*) \underset{\text{strict}}{\le} \mathcal{L}(w), \quad \forall w \in \mathbb{R}^D$

LS: Solving Lin Reg with MSE analytically. 2 steps for global opt. 1) Show convexity 2) $\nabla \mathcal{L}(w) = 0$

Normal eq: $\mathcal{L}(w) = \frac{1}{2N} \sum_{n=1}^{N} (y_n - x_n^T w)^2 = \frac{1}{2N} e^T e$ with $e = y - Xw$

1) Cost fct is convex bc it's combination of convex fcts Cost: $\Theta(ND^2 + D^3)$

2) $\nabla \mathcal{L}(w) = -\frac{1}{N} X^T e \overset{!}{=} 0 \Rightarrow X^T X w^* = X^T y \Rightarrow w^* = (X^T X)^{-1} X^T y$ iff Gram matrix $X^T X \in \mathbb{R}^{D \times D}$ is invertible (if $\text{rank}(X) = D$) Predict°n on unseen data:

→ $\hat{y}_m := x_m^T w^* = x_m^T (X^T X)^{-1} X^T y$. • if $D > N$: $\text{rank}(X) < D$ • if $D \le N$, but some col $x_{:d}$ are (nearly) collinear: ill-cond. ⇒ Solution: Linear Solver: $X^T X w^* = X^T y$

Underfitting: Can't find fit, too limited **Overfitting:** Fits noise too, too rich

Augment linear models: $y_n \approx w_0 + w_1 x_n + w_2 x_n^2 + \dots + w_M x_n^M =: \phi(x_n)^T w$

Avoid overfit: Increase N, keep M fixed (add data, same model complexity)

MLE: Gaussian RV in $\mathbb{R}^D$: $\mathcal{N}(y \mid \mu, \Sigma) = (2\pi)^{-D/2} \det(\Sigma)^{-1/2} \exp[-0.5 (y-\mu)^T \Sigma^{-1} (y-\mu)]$ (Σ PSD)

Note: 2 RV indep when $P(X,Y) = P(X)P(Y)$ in $\mathbb{R}$: $P(y \mid \mu, \sigma^2) = \mathcal{N}(y \mid \mu, \sigma^2) = (2\pi\sigma^2)^{-1/2} \exp[-(y-\mu)^2 / 2\sigma^2]$

Probabilistic model for LS: assume data $y_n = x_n^T w + \epsilon_n$ (unknown) with $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$

Likelihood: $p(y \mid X, w) = \prod_{n=1}^{N} p(y_n \mid x_n, w) = \prod_{n=1}^{N} \mathcal{N}(y_n \mid x_n^T w, \sigma^2)$ (independent; assumption on noise) Goal: to max it, and it's equivalent as min $\mathcal{L}_{MSE}$

Log: $\mathcal{L}_{LL}(w) := \log p(y \mid X, w) = -\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - x_n^T w)^2 + \text{cnst.}$

⇒ $\arg\min_w \mathcal{L}_{MSE}(w) = \arg\max_w \mathcal{L}_{LL}(w)$

MLE can also be interpreted as finding the model under which the observed data is most likely to have been generated from (probabilistically)

Properties:

• MLE is a sample approximation to the expected log-likelihood: $\mathcal{L}_{LL}(w) \approx \mathbb{E}_{p(y,x)}[\log P(y \mid x, w)]$

• MLE is consistent, i.e., it will give us the correct model assuming that we have a sufficient amount of data.

• $W_{MLE} \xrightarrow{P} W_{true}$ • MLE is efficient, i.e. it achieves the Cramer-Rao lower bound

Replace Gaussian with Laplace: $P(y_n \mid x_n, w) = (2b)^{-1} \exp[-\frac{1}{b} |y_n - x_n^T w|]$

Generalization: Data Model: unknown distrib D with range $X \times Y$

Dataset: $S = \{(x_n, y_n)\}_{n=1}^{N} \sim D$ iid   learning algo: $A(S) = f_S$ (output) (S: input)

Expect. error: $L_D(f) = \mathbb{E}_{(x,y) \sim D}[\ell(y, f(x))]$ (loss) But D is unknown   Empirical error: LLN: $L_S \xrightarrow{|S| \to \infty} L_D$

$L_S(f) = 1/|S| \sum_{(x_n,y_n) \in S} \ell(y_n, f(x_n))$ Training error: $L_S(f) = 1/|S| \sum_{(x_n,y_n) \in S} \ell(y_n, f_S(x_n))$

But could diverge from L_D bc of overfitting

Split data (test error): $L_{S_{test}}(f_{S_{train}}) = \frac{1}{|S_{test}|} \sum_{(x_n,y_n) \in S_{test}} \ell(y_n, f_{S_{train}}(x_n)) \approx L_D(f_{S_{train}})$

Generalization error: $P\left[\underbrace{|L_D(f) - L_{S_{test}}(f)|}_{\text{Gener. Gap}} \ge \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2|S_{test}|}} \right] \le \delta$ Error decreases as $\Theta(1/|S_{test}|^{1/2})$ w # of test points

(a,b) bounds on the loss upper bound

Hoeffding's: $P\left[\left| \frac{1}{N} \sum_{n=1}^{N} \Theta_n - E(\Theta) \right| \ge \epsilon \right] \le 2e^{-2N\epsilon^2/(b-a)^2}$ 1) With $\Theta = \ell(y_n, f(x_n))$

2) $\delta = 2e^{-2|S_{test}|\epsilon^2/(b-a)^2} \to \epsilon = \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2|S_{test}|}}$

Model Selection: Split data, train K times for each value, compute error.

Bound: $P\left[\max_k |L_D(f_k) - L_{S_{test}}(f_k)| \ge \sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2|S_{test}|}} \right] \le \delta$ Same Θ

Let $k^* = \arg\min_k L_D(f_k)$ and $\hat{k} = \arg\min_k L_{S_{test}}(f_k)$ ⇒ $P\left[L_D(f_{\hat{k}}) \ge L_D(f_{k^*}) + 2\sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2|S_{test}|}} \right] \le \delta$

K-fold Cross-Validation: 1. Randomly partion data into K groups.

2. Train K times, leaving 1 group for test and K-1 for train.

3. Average K results. → unbiased estimate of the gen. error and its var.

Logistic reg: model proba's {0,1} $\sigma(\eta) = (e^{\eta})(1+e^{\eta})^{-1}$, $1-\sigma(\eta) = (1+e^{\eta})^{-1}$ Robust to outliers and unbalanced data

$\sigma'(\eta) = \sigma(\eta)(1-\sigma(\eta))$, $P(1 \mid x) = \sigma(x^T w + w_0)$, $P(0 \mid x) = 1 - P(1 \mid x)$ | if $P(1 \mid x) \ge \frac{1}{2}$ → class 1

MLE: assume $X \perp w$: $\mathcal{L}(w) \propto \prod_{n=1}^{N} \sigma(x_n^T w)^{y_n} [1 - \sigma(x_n^T w)]^{1-y_n}$ LL: $w_* = \arg\min_w L(w)$

$:= \frac{1}{N} \sum_{n=1}^{N} -y_n x_n^T w + \log(1 + e^{x_n^T w})$ ⇒ Note: if {0,1}: $\ell(y, g(x)) = -y g(x) + \log(1 + \exp(g(x)))$ if {-1,1}: → $\log(1 + \exp(-y g(x)))$ ⇒ $\nabla \mathcal{L}(w) = \frac{1}{N} \sum_{n=1}^{N} (\sigma(x_n^T w) - y_n) x_n = \frac{1}{N} X^T (\sigma(Xw) - y)$ → No CS solution but convex! | GD: slow $\Theta(N)$ SGD: fast but converges slow

Newton's: $L(w) \sim L(w_t) + \nabla L(w_t)^T (w - w_t) + \frac{1}{2} (w - w_t)^T \nabla^2 L(w_t)(w - w_t)$ →

$w_{t+1} = w_t - \gamma_t \nabla^2 L(w_t)^{-1} \nabla L(w_t)$ ⇒ intensive! Regularized: issue if data lin separable. ⇒ $\frac{1}{N} \sum_{n=1}^{N} -y_n x_n^T w + \log(1 + e^{x_n^T w}) + \frac{\lambda}{2} \|w\|_2^2$

Optimization: $w^* = \arg\min_w \mathcal{L}(w)$ s.t. $w \in \mathbb{R}^D$ → but very sensitive to ill-conditioning → Solution: normalize

Smooth: 1) Gridsearch: Get the lowest loss over all W. Brute-force | Exponential complexity | No guarantee to find an optimum

2) Gradient descent: A gradient (at a point) is the slope of the tangent to the function (at that point). It points to the direction of largest increase of the function.

Vector: $\nabla \mathcal{L}(w) := \left[\frac{\partial \mathcal{L}(w)}{\partial w_1}, \dots, \frac{\partial \mathcal{L}(w)}{\partial w_D} \right]^T \in \mathbb{R}^D$ | update rule: $w^{(t+1)} := w^{(t)} - \gamma \nabla \mathcal{L}(w^{(t)})$ $\gamma > 0$ is the step-size or LR Popular: $\gamma = \frac{cst}{t}$ → smaller and smaller..

$\nabla \mathcal{L} = \frac{1}{2N} \sum_{n=1}^{N} (-2(y_n - w_0)) = w_0 - \bar{y} \overset{!}{=} 0$

Example: GD for 1-param and MSE: $w_0^{(t+1)} := (1-\gamma) w_0^{(t)} + \gamma \bar{y}$ (Converges for $\gamma \in (0, 2)$)

• GD for linear MSE: $y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$, $X = \begin{bmatrix} x_{11} & x_{12} \dots & x_{1D} \\ \vdots & \ddots & \\ x_{N1} & & x_{ND} \end{bmatrix}$, $e = y - Xw$ → $\mathcal{L}(w) = \frac{1}{2N} \sum_{n=1}^{N} (y_n - x_n^T w)^2 = \frac{1}{2N} e^T e$ | For MAE: $\nabla \mathcal{L}(w) = -\frac{1}{N} X^T \text{sign}(e)$

→ $\nabla \mathcal{L}(w) = -\frac{1}{N} X^T e$ Cost of GD per step: $\Theta(N.D)$

Loss of 1 data point

3) SGD: 1) Take 1 random n 2) $w^{(t+1)} := w^{(t)} - \gamma \nabla \mathcal{L}_n(w^{(t)})$ $\mathbb{E}_n[\nabla \mathcal{L}_n(w)] = \sum_{n=1}^{N} \frac{1}{N} \nabla \mathcal{L}_n(w)$ (Noisy) Cost of SGD per step $\Theta(D)$

Cheap and unbiased estimate of the gradient: $\mathbb{E}[\nabla \mathcal{L}_n(w)] = \nabla \mathcal{L}(w)$ $= \nabla \frac{1}{N} \sum_n \mathcal{L}_n(w) = \nabla \mathcal{L}(w)$ → stable

4) Mini-batch SGD: $w^{(t+1)} := w^{(t)} - \gamma g$ with $g = 1/|B| \sum_{n \in B} \nabla \mathcal{L}_n(w^{(t)})$, where B is a subset. → B=1: Classic SGD

B=N: classic GD → g easily parallelized

4.1) SGD with momentum: $w^{(t+1)} := w^{(t)} - \gamma m^{(t+1)}$ with $m^{(t+1)} := \beta_1 m^{(t)} + (1-\beta_1) g$

4.2) ADAM: $w_i^{(t+1)} := w_i^{(t)} - \frac{\gamma}{\sqrt{v_i^{(t+1)}}} m_i^{(t+1)} \quad \forall i$ with $m^{(t+1)} := \beta_1 m^{(t)} + (1-\beta_1) g$, $v_i^{(t+1)} := \beta_2 v_i^{(t)} + (1-\beta_2)(g_i)^2 \quad \forall i$

- faster forgetting of older weights
- is a momentum variant of Adagrad
- coordinate-wise adjusted learning rate
- strong performance in practice, e.g. for self-attention networks

4.3) Sign-SGD: $w_i^{(t+1)} := w_i^{(t)} - \gamma \, sign(g_i) \quad \forall i$ only use the sign (one bit) of each gradient entry → communication efficient for distributed training (but convergence issue)

Non-smooth: Sub

Vector where function is above is a subgradient if $\mathcal{L}$ convex and differentiable: $g = \nabla \mathcal{L}(w)$ Cost of GD per step: $\Theta(N.D)$

1) Subgradient D: Vector $g \in \mathbb{R}^D$ such that $\mathcal{L}(u) \ge \mathcal{L}(w) + g^T(u - w) \quad \forall u$

⇒ $w^{(t+1)} := w^{(t)} - \gamma g$ Trick: if $\mathcal{L}(w) = h(q(w))$ with h non-diff. and q diff., ⇒ subg of $\mathcal{L}$ at w is $g \in \partial h(q(w)) \cdot \nabla q(w)$ (set of subg of h) Cost of SGD per step $\Theta(D)$

2) S(S)GD: g = subgradient to $L_n(w)$

Conclusion: W local min if $\nabla \mathcal{L}(w) = 0$ (critical point) and $\nabla^2 \mathcal{L}(w) \succ 0$ (positive definite)

Regularization: Penalize complex models via: $\min_w \mathcal{L}(w) + \Omega(w)$ where $\Omega(w)$ is the regularizer $\lambda \to 0$ (no reg) → risk overfitting; $\lambda \to \infty$ (strong reg) → risk underfitting

L2: $\Omega(w) = \lambda \|w\|_2^2 = \lambda \sum_i w_i^2$ Ridge: L2 + MSE: $\min_w 1/2N \sum_{n=1}^{N} (y_n - x_n^T w)^2 + \lambda \|w\|_2^2$

Closed-form Sol: $w^*_{ridge} = (X^T X + \lambda' I)^{-1} X^T y$ → always invertible thanks to $\lambda' = 2N\lambda$ λ' lifts the EV so EV are at least λ'. Proof: $X^T X + \lambda' I = USU^T + \lambda' U I U^T = U[S + \lambda' I] U^T$

→ Fight ill-conditioning.

L1: $\Omega(w) = \lambda \|w\|_1 = \lambda \sum_i |w_i|$ Lasso: L1 + MSE: $\min_w 1/2N \sum_{n=1}^{N} (y_n - x_n^T w)^2 + \lambda \|w\|_1$ More powerful. Makes w sparse → few non-0 components

Note: $\{w : \|y - Xw\|^2 = \alpha\}$ is an ellipsoid. These are likely to touch a corner of the "L1-ball" ⇒ sparsity.

Notes: gradient of $\|w\|^2 = 2w$ Subgradient $\|w\|_1 = (\text{sign}(w_1), \dots, \text{sign}(w_D))$

Bias-Variance: $Y = f(x) + \epsilon$. Expected error: $\mathbb{E}_{(x,y) \sim D}((Y - f_S(x))^2)$ → True @ every point: $L(f_S) = \mathbb{E}_{\epsilon \sim D_\epsilon}((f(x_0) + \epsilon - f_S(x_0))^2)$

$\mathbb{E}_{S \sim D}[L(f_S)] = \mathbb{E}_{S \sim D, \epsilon \sim D_\epsilon}[(f(x_0) + \epsilon - f_S(x_0))^2] = \underbrace{\text{Var}_{\epsilon \sim D_\epsilon}[\epsilon]}_{\text{Noise var}} + \underbrace{(f(x_0) - \mathbb{E}_{S' \sim D}[f_{S'}(x_0)])^2}_{\text{Bias}} + \underbrace{\mathbb{E}_{S \sim D}[(\mathbb{E}_{S' \sim D}[f_{S'}(x_0)] - f_S(x_0))^2]}_{\text{Var}}$ (can't go below noise var)

Classification: unlike reg. Y ∈ discrete set: $y \in \{c_1, \dots, c_K\}$ Binary K=2 Classifier divides space into classes

KNN: Pros {No opt., easy, good for low dim} Cons {slow, bad for high dim, distance?} Max-Margin: HP which max margin Nonlinear classif: Feature aug. or Kernels Binary classif goal: $\min L_D(f) = \mathbb{E}_D(1_{y \ne f(x)}) = P_D(Y \ne f(x))$

Bayes classif: $f^* = \arg\min_f L_D(f) \to f^*(x) = \arg\max_{y \in \{-1,1\}} P(Y = y \mid X = x)$ unobtainable bc D not known

2 classes of classif algo 1) Non-Param: local AVG (KNN) 2) Param: min ERM Problem: Not convex. → Replace f and 1 by convex fct.

ERM: min ER instead True Risk: $\min_{f: x \to y} L_{train}(f) := \frac{1}{N} \sum_{n=1}^{N} 1_{f(x_n) \ne y_n} = \frac{1}{N} \sum_{n=1}^{N} 1_{y_n f(x_n) \le 0}$

Kernel: Ridge: $\min_w \frac{1}{2N} \sum_{n=1}^{N} (y_n - w^T x_n)^2 + \frac{\lambda}{2} \|w\|^2$ → 2 Solutions $w_* = \frac{1}{N} (\frac{1}{N} X^T X + \lambda I_d)^{-1} X^T y$ $w_* = \frac{1}{N} X^T (\frac{1}{N} X X^T + \lambda I_N)^{-1} y = X^T \alpha_*$ | $w^* \in \text{span}(x_1, \dots, x_N)$

Cov $O(d^3 + Nd^2)$ Kernel $O(N^3 + dN^2)$

Representer theorem: $w_* \in \arg\min_w \frac{1}{N} \sum_{n=1}^{N} \ell(x_n^T w, y_n) + \frac{\lambda}{2} \|w\|^2$ For ANY loss and min (better than LS) ⇒ $w_* = X^T \alpha_*$ → solut°n in

For ridge ($\ell = \|y - Xw\|^2$): alt formula $\alpha_* = \arg\min_\alpha \frac{1}{2} \alpha^T (\frac{1}{N} X X^T + \lambda I_N) \alpha - \frac{1}{N} \alpha^T y$ Kernel matrix: $K = XX^T = \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & \\ \vdots & \ddots & \\ x_N^T x_1 & \dots & x_N^T x_N \end{bmatrix} = (x_i^T x_j)_{i,j} \in \mathbb{R}^{N \times N}$

With feature map $\mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$: $K = \Phi\Phi^T = (\phi(x_i)^T \phi(x_j))_{i,j} \in \mathbb{R}^{N \times N}$ Problem: if $d \ll \tilde{d}$, $\phi(x)^T \phi(x)$ cost $O(\tilde{d})$ expensive!

Kernel Trick: K fct: $\kappa(x, x') = \phi(x)^T \phi(x')$ $\kappa(x, x')$ is in the original space! enable computation of classifiers in high-dimensional space without performing computations in this high-dimensional space.

Prediction: $y = \phi(x)^T w_*$ (Linear) is expensive, so use Rep theorem: $\phi(x)^T w_* = \phi(x)^T \phi(X)^T \alpha_* = \sum_{n=1}^{N} \kappa(x, x_n) \alpha_n$ (Non linear in X space) (≠ feature space)

Kernels: Linear: $\kappa(x, x') = x^T x'$, $\phi(x) = x$ | Quad: $\kappa(x, x') = (xx')^2$, $\phi(x) = x^2$ | Poly: $\kappa(x, x') = (x_1 x_1' + x_2 x_2' + x_3 x_3')^2$, $\phi(x) = [x_1^2, x_2^2, x_3^2, \sqrt{2} x_1 x_2, \sqrt{2} x_1 x_3, \sqrt{2} x_2 x_3]$ | RBF: $\kappa(x, x') = e^{-(x - x')^2}$, $\phi(x) = e^{-x^2} (\dots, \frac{2^k x^k}{\sqrt{k!}}, \dots)$ • $K(x, x') = f(\kappa(x, x'))$

Create kernels: ⊕ lin Comb: $\kappa(x, x') = \alpha \kappa_1(x, x') + \beta \kappa_2(x, x')$ for $\alpha, \beta \geq 0$ Products: $\kappa(x, x') = \kappa_1(x, x') \kappa_2(x, x')$ ⇒ • $K(x, x') = e^{\kappa(x, x')}$

Mercers condition: given K, ensures ϕ iff: Kernel fct is symmetric $\kappa(x, x') = \kappa(x', x)$ Kernel matrix psd $K = (\kappa(x_i, x_j))_{i,j=1}^{N} \succeq 0 \quad \forall N \ge 0, \forall (x_i)_{i=1}^N$

SVM: Hard-SVM (LS): Max-margin separating HP: $\max_{w, \|w\|=1} \underbrace{\min_{n \le N} |w^T x_n|}_{\text{Margin}}$ such that $\underbrace{\forall n, y_n x_n^T w \ge 0}_{\text{Correctly classified}}$ Equiv to: $\max_{M \in \mathbb{R}, w, \|w\|=1} M$ such that $\forall n, y_n x_n^T w \ge M$ $M = \frac{1}{\|w\|}$

⇒ $\min_w \frac{1}{2} \|w\|^2$ such that $\forall n, y_n x_n^T w \ge 1$ Soft-SVM: Not linearly separable! → use Slack variable: $y_n x_n^T w \ge 1 - \xi_n$

⇒ $\min_{w, \xi} \frac{\lambda}{2} \|w\|^2 + \frac{1}{N} \sum_{n=1}^{N} \xi_n$ s.t. $\forall n, y_n x_n^T w \ge 1 - \xi_n$ and $\xi_n \ge 0$ equiv to: $\min_w \frac{\lambda}{2} \|w\|^2 + \frac{1}{N} \sum_{n=1}^{N} [1 - y_n x_n^T w]_+$ → ERM for hinge loss with ridge λ

Margin-base losses ($\eta = y x^T w$): $\text{MSE}(\eta) = (1 - \eta)^2$ $\text{Hinge}(\eta) = [1 - \eta]_+$ be: If $y_n x_n^T w \ge 1$, then $\xi_n = 0$ If $y_n x_n^T w < 1$, $\xi_n = 1 - y_n x_n^T w$

$\text{Logistic}(\eta) = \frac{\log(1 + \exp(-\eta))}{\log(2)}$ all upper bound for 0-1 loss

Opt: Convex duality: How to get w for Soft-SVM ⇒ Find $G(w, \alpha)$ so $\min_w L(w) = \min_w \max_\alpha G(w, \alpha)$ (Primal) $\ge \max_\alpha \min_w G(w, \alpha)$ (Dual)

1) $[1 - y_n x_n^T w]_+ = \max_{\alpha_n \in [0,1]} \alpha_n (1 - y_n x_n^T w)$ → $\min_w L(w) = \min_w \max_{\alpha \in [0,1]^N} \underbrace{\frac{1}{N} \sum_{n=1}^{N} \alpha_n (1 - y_n x_n^T w) + \frac{\lambda}{2} \|w\|_2^2}_{G(w, \alpha)}$ → G convex in w, concave in α, so we can invert min max

2) Min: $\nabla_w G(w, \alpha) = -\frac{1}{N} \sum_{n=1}^{N} \alpha_n y_n x_n + \lambda w = 0 \Longrightarrow w(\alpha) = \frac{1}{\lambda N} \sum_{n=1}^{N} \alpha_n y_n x_n = \frac{1}{\lambda N} X^T Y \alpha$ (diag(y)) 3) Max: $\min_w L(w) = \max_{\alpha \in [0,1]^N} \frac{1^T \alpha}{N} - \frac{1}{2\lambda N^2} \alpha^T \underbrace{Y X X^T Y}_{\text{PSD}} \alpha$ (Kernel of data) $\alpha_n > 0$ are SV!

$\alpha_n = 0$: x_n ✓ Not margin, $\alpha_n \in (0,1)$: x_n ✓ margin, $\alpha_n = 1$: x_n ✗

Ethics: We want knowledge, not stereotypes. ML can't distinguish. data might discriminate minorities Three Fairness criterias: Can't satisfy them all.

Independence: R independent of A ⟹ equal acceptance rate

Separation: R independent of A, conditional on Y ⟹ equal error rate

Sufficiency: Y independent of A conditional on R ⟹ calibration by group

K-NN: $\text{nbh}_{S_{\text{train},k}}: \mathcal{X} \to \mathcal{X}^k$ $x \mapsto$ {the k elements of S_{train} closest to x} Slow: $\boxed{O(N)}$

For reg: $f_{S_{\text{train}},k} = \frac{1}{k}\sum_{\in k} y_n$ (AVG of points around) For classif: $f_{S_{\text{train}},k}$ = Majority $\{y_n : x_n \in \text{nbh}\}$

Small K: Low Bias, High Var. Use odd K. When K=N: cst prediction.

Curse of D: When dimension d gets large (with fixed N), hard to find a neighboring point. $X = Y = [0,1]^d \times \{0,1\}$

Bayes Classifier: min L on all classif $f_*(x) = 1_{\eta(x) \geq 1/2}$

Proba min of missclassif $\eta(x) = \mathbb{P}(Y = 1 \mid X = x)$

Bayes Risk: $L(f_*) = \mathbb{P}(f_*(X) \neq Y) = \mathbb{E}_{X \sim \mathcal{D}_X}[\min\{\eta(X), 1 - \eta(X)\}]$

Bound for 1-NN: $\mathbb{E}_{S_{\text{train}}}[L(f_{S_{\text{train}}})] \leq 2L(f_*) + 4c\sqrt{d}N^{-\frac{1}{d+1}}$ c Lipschitz

If $N \to \infty$, bound is $2L(f_*)$. But we need $N \propto d^{(d+1)/2}$ to have constant error. Curse of D!

CNN, Conv: filter f, size K, stride S. $x_{n,m}^{(1)} = \sum_{k,l=0}^{K-1} f_{k,l} \cdot x^{(0)}_{nS+k, mS+l}$

→ Same filter used → weight sharing. Value depends on close values.

→ 0 and valid padding. Can also use multiple filters. $W_{out} = \frac{W_{in} + 2P - F}{S} + 1$

Conv layer has multiple filters. HP: size, padding, stride

Pooling: Max or Avg → reduce spatial dim. HP: size, type, stride

ReLU after each conv layer to make it non-linear.

Backpropa: 1) Backpropa indep 2) Sum grad of edges with same weights

ResNet: Add X to standard network $F(x)$, → Skip-connection $R(x) + x$

Data augmentation: transform $\tau: \mathbb{R}^d \to \mathbb{R}^d$ (same label) by rotation

Weight decay: $\min \mathcal{L} + \frac{\lambda}{2}\sum_l \|W^{(l)}\|_F^2$ No need for Bias.

Dropout: Subnetwork by keeping with proba $p^{(l)}$ each node. But use whole network when testing!

Adversarial ML: Standard vs Adv risk: $R(f) = \mathbb{E}_{\mathcal{D}}\left[1_{f(X) \neq Y}\right]$ $R_\epsilon(f) = \mathbb{E}_{\mathcal{D}}\left[\max_{\hat{x}, \|\hat{x} - X\| \leq \epsilon} 1_{f(\hat{x}) \neq Y}\right]$

Generating adv example: given input (x, y) and $f: X \to \{-1, 1\}$: find $\hat{x}$ st: a) $\|\hat{x} - x\| \leq \epsilon$ b) the model f makes a mistake ⇒ $\max_{\hat{x}, \|\hat{x}-x\| \leq \epsilon} 1_{f(\hat{x}) \neq y}$

Issues: 1) $1_{(\cdot)}$ not continuous 2) NN pred output $\{-1, 1\}$. Solutions: 1) use smooth ℓ 2) Consider output before classif ($f(x) = \text{sign}(g(x))$). Note: $g(x) = P(y=1 \mid x)$

⇒ White-box: Solve (know g) $\max_{\hat{x}, \|\hat{x}-x\| \leq \epsilon} \ell(y g(\hat{x}))$ ∇_x: $y\ell'(yg(x))\nabla_x g(x)$ $(\propto -y\nabla_x g(x))$

≤ 0 since classification loss are decreasing

L_2: $\hat{x} = x - \epsilon y \cdot \frac{\nabla_x g(x)}{\|\nabla_x g(x)\|_2}$ L_∞: $\hat{x} = x - \epsilon y \cdot \text{sign}(\nabla_x g(x))$ ⇒ PGD: multiple steps!

don't know $g(x)$

Black-box: 1a) Score-based: query conti. model scores $g(x)$ but needs lots of queries

$\nabla_x g(x) \approx \sum_{i=1}^d \frac{g(x + \alpha e_i) - g(x)}{\alpha} e_i$ 1b) Decision-based: query only predicted class $f(x) \in \{-1, 1\}$

2) Transfer attacks: Train $\hat{f} \approx f$ on similar data. ⇒ Model stealing

Train robust models: Train and min on adv examples and risk.

⇒ $\min_\theta \frac{1}{N}\sum_{n=1}^N \max_{\hat{x}_n, \|x_n - \hat{x}_n\| \leq \epsilon} \ell(y_n g_\theta(\hat{x}_n))$ 1) approx $\hat{x}_n^* \approx \arg\max_{\|\hat{x}_n - x_n\| \leq \epsilon} \ell(y_n g_\theta(\hat{x}_n))$ with PGD

2) GD w.r.t θ using $\frac{1}{N}\sum_{n=1}^N \nabla_\theta \ell(y_n g_\theta(\hat{x}_n^*))$

Note: robustness comes at the cost of accuracy

Matrix facto: $X (D \times N) \approx WZ^T$, $W \in (D \times K)$ item (row), $Z \in (N \times K)$ user (row), $K \ll D, N$

⇒ $\min_{W,Z} \mathcal{L}(W, Z) = \frac{1}{2}\sum_{(d,n) \in \Omega}[x_{dn} - (WZ^T)_{dn}]^2$ ⇒ Opt. not jointly convex and not unique

K: # latent feat. large K → overfit Reg: $\mathcal{L}(W, Z) + \frac{\lambda_W}{2}\|W\|_F^2 + \frac{\lambda_Z}{2}\|Z\|_F^2$

$\|A\|_F = \sqrt{\sum_i^M \sum_j^N a_{ij}^2}$ SGD: Derive ∇ for (d', k) (W), (n', k) (Z) for fixed (d, n).

$\frac{\partial}{\partial w_{d',k}} f_{d,n}(W, Z) = \begin{cases} -[x_{dn} - (WZ^T)_{dn}] z_{n,k} & \text{if } d' = d \\ 0 & \text{otherwise} \end{cases}$ $\frac{\partial}{\partial z_{n',k}} f_{d,n}(W, Z) = \begin{cases} -[x_{dn} - (WZ^T)_{dn}] w_{d,k} & \text{if } n' = n \\ 0 & \text{otherwise} \end{cases}$

ALS: Assume X is full, we use coord descent: Cost $\boxed{O(K)}$ indep of N, D!

$\min \frac{1}{2}\sum^D \sum^N [x_{dn} - (WZ^T)_{dn}]^2 = \frac{1}{2}\|X - WZ^T\|_F^2$ → Fix W min Z, vice-versa

⇒ $Z^T := (W^TW + \lambda_z I_K)^{-1}W^TX$ $W^T := (Z^TZ + \lambda_w I_K)^{-1}Z^TX^T$ → optimal Cost $\boxed{O(DKN + K^3)}$?

⇒ But X not full: $\min \sum_{(d,n) \in \Omega} [x_{dn} - (WZ^T)_{dn}]^2$ → 1) $\nabla_W \mathcal{L}(W, Z) = 0$ 2) $\nabla_Z \mathcal{L} = 0$ ...

Text represent: find $w_i \to W_i \in \mathbb{R}^K$ Words → Features Co-occur matrix: n_{ij} = # contexts where w_i occurs with w_j. Very sparse D×N. Need context (size) and voc V

⇒ Matrix facto: log of counts: $x_{dn} = \log(n_{dn})$ → $X \approx WZ^T$ where W (word), Z (context word)

$\min_{W,Z} \mathcal{L}(W, Z) := \frac{1}{2}\sum_{(d,n) \in \Omega} f_{dn}[x_{dn} - (WZ^T)_{dn}]^2$ Glove: $f_{dn} := \min\{1, (n_{dn}/n_{max})^\alpha\}$, $\alpha \in [0, 1]$

Training w SGD or ALS: $\boxed{O(K)}$ per step Skip-gram (Word2Vec): use log-reg to separate real word pairs (w_d, w_n) from fake ones. (real: appearing in context window 5)

Fast text: Supervised. $S_n = (w_1, \ldots, w_m)$ and $x_n \in \mathbb{R}^{|V|}$ a BoW of S_n and $y_n \in \{-1, 1\}$ label:

$\min_{W,Z} \mathcal{L}(W, Z) = \sum_{S_n \text{ a sentence}} f(y_n WZ^T x_n)$, $W \in \mathbb{R}^{1 \times K}$, $Z \in \mathbb{R}^{|V| \times K}$, f a classifier

Supervised: use CNN or mat fact unsupervised: adding or avg word vectors and train, then direct unsupervised train using sentences instead of words.

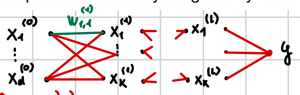
LLMs: 1) LM: distribution over text: $p(\text{I saw a cat on a mat}) = p(x_1, \ldots, x_l)$ Next-token pred: $P(X_t \mid X_1, \ldots, X_{t-1})$. $P(X_{t-1} \mid X_1, \ldots, X_{t-2}) \ldots P(X_1)$ → $p(\text{mat} \mid \text{I saw a cat on a}) = 0.002$

Tokenizer: indep from model AR inference: 1) Tokenize 2) Forward 3) Proba for next token 4) Sample 5) Detoken 6) repeat 2) Data: Pretraining: A lot of low quality to learn.

Post training: Small of high quality to make model usable. Distributed learning: LLM need multiple GPUs to run in parallel 3) Posttraining: ZS vs FS (in-context): if we give examples to GPT, it will perform better. Also if we give examples of reasoning (CoT). Supervised finetuning: Have labelers show desired output

NN: Linear: $f_{Lin}(x) = \phi(x)^T w$ (fixed) NN: $f_{NN}(x) = f(x)^T w$ (learned) | L hidden layers d inputs K neurons $W^{(l)}$ matrices $b^{(l)}$ vectors

Outputs of hidden layer l given by vector: $x^{(l)} = f^{(l)}(x^{(l-1)}) := \phi((W^{(l)})^T x^{(l-1)} + b^{(l)})$ $x_j^{(l)} = \phi\left(\sum_{i=1}^K x_i^{(l-1)} w_{ij}^{(l)} + b_j^{(l)}\right)$



Non-linearity is important

Params: $w_{i,j}^{(l)}$, $b_j^{(l)}$ $\boxed{O(dK + K^2L)}$ Barron's theorem: $\int_{|x| \leq r} (f(x) - f_n(x))^2 dx \leq \frac{(2Cr)^2}{n}$ Also use comb of ReLU as PWL

Representation power: We can use sigmoids to approximate functions. Cross: $(x_1, x_2) \mapsto \phi(w(x_1 - a_1)) - \phi(w(x_1 - b_1)) + \phi(w(x_2 - a_2)) - \phi(w(x_2 - b_2))$

MSE: $\min_{w_{i,j}^{(l)}, b_j^{(l)}} \mathcal{L}(f) = \frac{1}{2N}\sum_{n=1}^N (y_n - f(x_n))^2$ where f is the fct of NN. ⇒ Non-convex! | Note: We could add a Reg term.

SGD: $(w_{i,j}^{(l)})_{t+1} = (w_{i,j}^{(l)})_t - \gamma\frac{\partial \mathcal{L}_n}{\partial w_{i,j}^{(l)}}$ $(b_j^{(l)})_{t+1} = (b_j^{(l)})_t - \gamma\frac{\partial \mathcal{L}_n}{\partial b_j^{(l)}}$ But doing all chain rules too expensive! Solution: Backpropagation.

NN params: $W^{(1)}: d \times K$, $W^{(l)}: K \times K$ for $2 \leq l \leq L$, $W^{(L+1)}: K \times 1$, $b^{(l)}: K \times 1$ for $1 \leq l \leq L$, $b^{(L+1)}: 1 \times 1$ | $x^{(l)} = f^{(l)}(x^{(l-1)}) := \phi((W^{(l)})^T x^{(l-1)} + b^{(l)}) = \phi(z^{(l)})$

⇒ Final layer $y = f^{(L+1)}(x^{(L)}) := (W^{(L+1)})^T x^{(L)} + b^{(L+1)}$ and overall function is a compo: $y = f(x^{(0)}) = f^{(L+1)} \circ f^{(L)} \circ \cdots \circ f^{(2)} \circ f^{(1)}$

Goal: compute for all (i, j, l): $\frac{\partial \mathcal{L}_n}{\partial w_{i,j}^{(l)}}$ and $\frac{\partial \mathcal{L}_n}{\partial b_j^{(l)}}$. FW pass: Set $x^{(0)} = x_n$, then compute $z^{(l)}$ and $x^{(l)}$ $O(K^2L)$

BW pass: Define $\delta_j^{(l)} = \frac{\partial \mathcal{L}_n}{\partial z_j^{(l)}} = (W^{(l+1)}\delta^{(l+1)}) \odot \phi'(z^{(l)})$ → Initialize by $\delta^{(L+1)} = z^{(L+1)} - y_n$ (MSE!) $O(K^2L)$

⇒ $\frac{\partial \mathcal{L}_n}{\partial b_j^{(l)}} = \delta_j^{(l)}$ $\frac{\partial \mathcal{L}_n}{\partial w_{i,j}^{(l)}} = \delta_j^{(l)} \cdot x_i^{(l-1)}$. Issues with GD: Grad vanishing or exploding. Solutions are:

1) Param initialization: Control layerwise Var of the neurons (He)! $z^{(l)} \sim \mathcal{N}(0, I_K)$ $w_i^{(l+1)} \sim \mathcal{N}(0, \sigma^2 I_K)$ $z_i^{(l+1)} = ReLU(z^{(l)})^T w_i^{(l+1)}$ → $\sigma^2 = 2/K$ for $\text{Var}(z_i^{(l+1)}) = 1$

2) Batchnorm (B ∈ M): $\hat{z}_n^{(l)} = (z_n^{(l)} - \mu_B^{(l)}) // \sqrt{(\sigma_B^{(l)})^2 + \varepsilon}$ then: $\hat{z}_n = \gamma^{(l)} \odot \hat{z}_n^{(l)} + \beta^{(l)}$ Layer norm (on K): $\hat{z}_n^{(l)} = (z_n^{(l)} - \mu_n^{(l)} 1_K) // \sqrt{(\sigma_n^{(l)})^2 + \varepsilon}$ then same

Transformers: f: seq → seq input (T) token → T tokens of dim D $\xrightarrow{L \text{ transf}}$ T×D → output Transformer: LN, SA (between), MLP (within), SC

Input: Tokenized → Embedding: map each token ID into vector $W_i \in \mathbb{R}$ → $X = [W_0\ W_{23} \ldots W_{60}]^T \in \mathbb{R}^{T \times D}$

$W_{emb} = [W_1 \ldots W_{N_{voc}}] \in \mathbb{R}^{N_{voc} \times D}$ learnt via backpropa. $X = [e_{i_1}, \ldots, e_{i_T}]^T W_{emb}$. Position: $X = [e_{i_1}, \ldots, e_{i_T}]^T W_{emb} + [e_1, \ldots, e_T]^T W_{pos}$

⇒ SA doesn't account for position

SA: A: tok → tok using learned indep. Weighted avg. ⇒ $Z = \text{softmax}\left(\frac{XW_QW_K^TX^T}{\sqrt{D_K}}\right)XW_V$ $\boxed{O(T^2)}$ W_K, W_Q $(D \times D_K)$, W_V $(D \times D_V)$

⇒ Can also do it for H SA: Z_h

MLP: mix info within each token: $MLP(X) = \varphi(XW_1)W_2$ → applied to each token. Output: Single or multiple (simple)

Transformers are used for Encoders (classification), Decoders (chat GPT), Encoder-Decoder (translation)

Unsupervised learning: Model-Based clustering: K-Means: Fix K clusters and find cluster centroids μ_k and assignment $z_{nk} = \{0, 1\}$

Objective Function: $\min_{z,\mu} \mathcal{L}(z, \mu) = \sum_{n=1}^N \sum_{k=1}^K z_{nk}\|x_n - \mu_k\|_2^2$ s.t. $\mu_k \in \mathbb{R}^D, z_{nk} \in \{0, 1\}, \sum_{k=1}^K z_{nk} = 1$ (WCSS) W: within

Algo: 1) input data D, # clusters K 2) Initialize random μ_K 3) Assign each point to new center $z_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|x_n - \mu_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$

4) Update centroids (means): $\mu_k = \frac{\sum_{n=1}^N z_{nk}x_n}{\sum_{n=1}^N z_{nk}}$ ⇒ repeat 3) 4) until convergence Iteration cost $O(NKd)$ Every iteration ↓ the WCSS

Convergence: Coordinate descent: $z^{(t+1)} = \arg\min_z \mathcal{L}(z, \mu^{(t)})$ $\mu^{(t+1)} = \arg\min_\mu \mathcal{L}(z^{(t+1)}, \mu)$ → optimum not guaranteed

$\min_{Z,M} \mathcal{L}(Z, M) = \|X^T - MZ^T\|_{Frob}^2$ Challenges: 1) Depends on initialization ⇒ K-means++: spread initial centroids $O(\log K)$ optimal solution

2) Which K optimal? ⇒ Elbow method of K on WCSS. 3) Can't model other shapes than spheres. 4) Sensitive to outliers

5) Hard assignments in overlapping clusters.

GMM: Calculate distrib of clusters. K Gaussians. Z (unobserved) multinomial → $P(z = k) = \pi_k$ → $\sum_k \pi_k = 1$

MLE: $\max_\theta \sum_{n=1}^N \log\left(\sum_{k=1}^K \pi_k \mathcal{N}(x_n \mid \mu_k, \Sigma_k)\right)$ but non-convex, no unique opt, unbounded. ⇒ Need EM algorithm

Hard EM: $\theta^{(0)} = \{\pi_k, \mu_k, \Sigma_k\}_{k \in [K]}$. E: Predict most likely class for each data point: $z_n^{(t)} = \arg\max_z \pi_z \mathcal{N}(x_n \mid \mu_z, \Sigma_z)$ → We have labeled data!

M: MLE of θ using $D_c^{(t)} = \{(x_n, z_n^{(t)})\}_{n \in [N]}$: $\theta^{(t)} = \arg\max_\theta p(\mathcal{D}_c^{(t)} \mid \theta) = \arg\max_\theta \sum_{n=1}^N \log(\pi_{z_n^{(t)}}\mathcal{N}(x_n \mid \mu_{z_n^{(t)}}, \Sigma_{z_n^{(t)}}))$. K-Means: Hard EM, $\pi_k = \frac{1}{K}$, $\Sigma_k = \sigma^2 I$

Soft EM: Use posteriors $\theta^{(0)} = \{\pi_k, \mu_k, \Sigma_k\}_{k \in [K]}$ E: Compute posteriors: $q_k^{(t)}(x_n) = p(z_n = k \mid x_n, \theta^{(t-1)}) = \frac{\pi_k \mathcal{N}(x_n; \mu_k^{(t-1)}, \Sigma_k^{(t-1)})}{\sum_{k'=1}^K \pi_{k'}\mathcal{N}(x_n; \mu_{k'}^{(t-1)}, \Sigma_{k'}^{(t-1)})}$

M: MLE of θ using $q_k^{(t)}(x_n)$: $\pi_k^{(t)} = \frac{1}{N}\sum_n q_k^{(t)}(x_n)$, $\mu_k^{(t)} = \frac{\sum_{n=1}^N q_k^{(t)}(x_n)x_n}{\sum_{n=1}^N q_k^{(t)}(x_n)}$, $\Sigma_k^{(t)} = \frac{\sum_{n=1}^N q_k^{(t)}(x_n)(x_n - \mu_k^{(t)})(x_n - \mu_k^{(t)})^T}{\sum_{n=1}^N q_k^{(t)}(x_n)}$ Iteration cost $O(NKD^3)$

General EM: Not only GMM: $\mathcal{L}(\theta) = \mathcal{L}(q, \theta) + KL(q \| P)$ where q is distribution $q(z)$ s.t. $\sum_z q(z) = 1$. Coordinate ascent algorithm on $\mathcal{L}(q, \theta)$

Goal: Max incomplete $\mathcal{L}(\theta) = \log(P(X \mid \theta))$ by max lower bound: $\mathcal{L}(q, \theta) = \log\left(\frac{P(X, Z \mid \theta)}{q(z)}\right)$

E: Compute $P(Z \mid X, \theta^{(t)})$: $q^{(t+1)} = \arg\max_q \mathcal{L}(q, \theta^{(t)})$. Write $\log p(X, Z \mid \theta)$ and calculate: $\mathbb{E}_Z[\log p(X, Z \mid \theta) \mid X, \theta^{(t)}]$

M: Max $\theta^{(t+1)} = \arg\max_\theta \mathbb{E}_Z[\log p(X, Z \mid \theta) \mid X, \theta^{(t)}]$ ⇒ $\theta^{(t+1)} = \arg\max_\theta \mathcal{L}(q^{(t+1)}, \theta)$ Note: $\mathcal{L}(\theta^{(t+1)}) \geq \mathcal{L}(q^{(t+1)}, \theta^{(t+1)}) \geq \mathcal{L}(q^{(t+1)}, \theta^{(t)}) \geq \mathcal{L}(\theta^{(t)})$

alternately computing a lower bound on the log likelihood for the current parameter values (E) and then maximizing this bound to obtain the new parameter values (M)

Initialization: Weights: uniform Means: Rand or K-Means++ Var: empirical

Self-supervised: uses input data to create label or tasks ⇒ $g: X \to (x_{in}, x_{out})$ then learn $f: x_{in} \to x_{out}$ (e.g. MLM)

MLM: Learn to predict hidden words. BERT: finds masked words, learn order of the sentence and classifies.

Joint embed. methods: rather than $f: x_{in} \to x_{out}$, learn $f: X \to \mathbb{R}^d$ s.t. $f(x_1) \approx f(x_2)$ Problem: constant f. Solution:

BYOL: $f_1(x_1) \approx f_2(x_2)$ and Contrastive learning: Similarity of ⊕ example pair better than w ⊖: $s(f(x), f(x^+)) > s(f(x), f(x^-))$

SimCLR: 1. Classifies $L(x) = -\mathbb{E}\left[\log\frac{\exp(s(f(x), f(x^+)))}{\sum_{i=1}^{N} \exp(s(f(x), f(x_i)))}\right]$ 2. Map encoder output to similarity space: $f(x) = f_2 \circ f_1(x)$ f_1 encoder, f_2 projector

3. Cosine similarity: $S(e_1, e_2) = \frac{\langle e_1, e_2 \rangle}{\|e_1\|_2\|e_2\|_2} / \tau$ 4. Data augmentation CLIP: Max similarity of captions and images

Gen Models: instead of model $P(y \mid X)$, model $P(X)$. Explicit (Diffusion): model prob distrib of X. Implicit (GAN): generate samples according to it.

GAN: Generator $G(\theta)$, Discriminator $D(\varphi)$. D judges what G produces. $\theta^* \in \arg\min_{\theta \in \Theta} \mathcal{L}^\theta(\theta, \varphi^*)$ $\varphi^* \in \arg\min_{\varphi \in \Phi} \mathcal{L}^\varphi(\theta^*, \varphi)$

1) D distinguishes real vs. fake sample. 2) G wants to fool D. 3. Obj: $\min_G \max_D \mathbb{E}_{x \sim p_d}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$ Losses:

$\mathscr{L}_D(G, D) = \max_D \mathbb{E}_{x \sim p_d}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$ $\mathscr{L}_G(G, D) = \min_G \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$ 4. Opt solution: $P_g = P_d = -\log 4$ | CGAN: Condition

Diff models: add noise and train model to recover the data. Consider a MC with $X_0 \sim P_0$ and forward transition such that $X_{t+1} \sim P(\cdot \mid X_t)$. Forward decompo: $p(x_{0:T}) = p_0(x_0)\prod_{t=0}^{T-1} p(x_{t+1} \mid x_t)$ where at each step, the marginal satisfies

$p(x_t) = \int p(x_t \mid x_{t-1})p(x_{t-1})\,dx_{t-1}$. Using bayes $(p(x_t \mid x_{t+1}) = p(x_{t+1} \mid x_t)p(x_t)/p(x_{t+1}))$ ⇒ Backward! $p(x_{0:T}) = p_T(x_T)\prod_{t=0}^{T-1} p(x_t \mid x_{t+1})$

We use ancestral sampling: Sample $X_T \sim p_T(\cdot)$ then: for $k = T-1, \ldots, 0$: Sample $X_t \sim p(\cdot \mid X_{t+1})$

$P_0 = P_{data} \in \mathcal{P}(\mathbb{R}^d)$ | Forward transition: $P(X_{t+1} \mid X_t) = \mathcal{N}(X_{t+1}; \alpha X_t, (1 - \alpha^2)I_d)$ | $P(X_t \mid X_0) = \mathcal{N}(X_t; \alpha^t X_0, (1 - \alpha^t)I_d)$ | MC converges to $P_{ref} = \mathcal{N}(x; 0, I_d)$ for $T \to \infty$ ⇒ For T large enough: $P_T(x) \approx P_{ref}(x)$. Problem: BW transition $P(X_t \mid X_{t+1})$ needs to be approx

⇒ $p(x_t \mid x_{t+1}) = p(x_{t+1} \mid x_t)\exp[\log p(x_t) - \log p(x_{t+1})] \approx \mathcal{N}(x_t; (2 - \alpha)x_{t+1} + (1 - \alpha^2)\underbrace{\nabla \log p(x_{t+1})}_{\text{Score}}, (1 - \alpha^2)I_d)$

Approximate score with a NN: $\theta^* = \arg\min_\theta \frac{1}{2}\sum_{t=1}^T \mathbb{E}_{X_0}\mathbb{E}_{X_t \mid X_0}[\|s_\theta(t, X_t) - \nabla \log p(X_t \mid X_0)\|^2]$