1 Convexity

- A function is convex iff a line segment between two points on the function's graph always lies above the function.
- A function $h(\mathbf{u}), \mathbf{u} \in \mathbb{R}^D$ is convex if $\forall$ $\mathbf{u}, \mathbf{v} \in \mathbb{R}^D$, $0 \le \lambda \le 1$: $h(\lambda \mathbf{u} + (1-\lambda)\mathbf{v}) \le \lambda h(\mathbf{u}) + (1-\lambda) h(\mathbf{v})$ (Strictly convex if $\le \Rightarrow <$)
- A strictly convex function has a unique global minimum $\mathbf{w}^*$. For convex functions, every local minimum is a global minimum.
- Sums of cvx funcs $\Rightarrow$ cvx (MSE+Lin model = cvx)

2 Optimization

- Local min $\mathbf{w}^* \to \exists \varepsilon > 0$ s.t. $\mathcal{L}(\mathbf{w}^*) \le \mathcal{L}(\mathbf{w})$ $\forall \mathbf{w}$ w/ $\|\mathbf{w} - \mathbf{w}^*\| < \epsilon$
- Global minimum $\mathbf{w}^*$, $\mathcal{L}(\mathbf{w}^*) \le \mathcal{L}(\mathbf{w})$ $\forall \mathbf{w} \in \mathbb{R}^D$

Gradient Descent (GD): $\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma \nabla \mathcal{L}(\mathbf{w}^{(t)}), \gamma > 0$

Gradient Descent for Linear MSE, $\mathcal{O}(N \times D)$

$\mathcal{L}(\mathbf{w}) := \frac{1}{2N}\sum_{n=1}^N (y_n - \mathbf{x}_n^\top \mathbf{w})^2 = \frac{1}{2N}\mathbf{e}^\top \mathbf{e}$, $\nabla \mathcal{L}(\mathbf{w}) = -\frac{1}{N}\mathbf{X}^\top \mathbf{e}$

$\mathbf{e} := \mathbf{y} - \mathbf{X}\mathbf{w}$

Stochastic Gradient Descent (SGD), $\mathcal{O}(D)$

$\mathcal{L}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^N \mathcal{L}_n(\mathbf{w})$, $\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma \nabla \mathcal{L}_n(\mathbf{w}^{(t)})$

- Cheap and unbiased estimate of the gradient $E[\nabla \mathcal{L}_n(w)] = \frac{1}{N}\sum_{n=1}^N \nabla \mathcal{L}_n(w) = \nabla(\frac{1}{N}\sum \cdots) = \nabla \mathcal{L}(w)$

Mini-batch SGD: $\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma \mathbf{g}$

$\mathbf{g} := \frac{1}{|B|}\sum_{n\in B} \nabla \mathcal{L}_n(\mathbf{w}^{(t)})$, subset $B \subseteq [N]$ of training samples

Non-Smooth Optimization

- Convexity, for differentiable functions: $\mathcal{L}(\mathbf{u}) \ge \mathcal{L}(\mathbf{w}) + \nabla \mathcal{L}(\mathbf{w})^\top(\mathbf{u} - \mathbf{w})$ $\forall \mathbf{u}, \mathbf{w}$

Subgradients

- A vector $\mathbf{g} \in \mathbb{R}^D$ s.t. $\mathcal{L}(\mathbf{u}) \ge \mathcal{L}(\mathbf{w}) + \mathbf{g}^\top(\mathbf{u} - \mathbf{w})$ $\forall \mathbf{u}$ is called a subgradient to the function $\mathcal{L}$ at $\mathbf{w}$.
- This definition makes sense for objectives $\mathcal{L}$ which are not necessarily differentiable (and not even necessarily convex).
- $\mathcal{L}$ convex and differentiable at $\mathbf{w} \to$ only subgradient $\mathbf{g} = \nabla \mathcal{L}(\mathbf{w})$.

Subgradient Descent

$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - \gamma \mathbf{g}$ for $\mathbf{g}$ a subgradient to $\mathcal{L}$

Convex Sets

- Intersubsections of convex sets are convex
- Projections onto convex sets are unique.
- Formal definition: $P_{\mathcal{C}}(\mathbf{w}') := \arg\min_{\mathbf{v}\in\mathcal{C}} \|\mathbf{v} - \mathbf{w}'\|$.

Projected Gradient Descent: SGD + Projection step

- Idea: add a projection onto $\mathcal{C}$ after every step:

$P_{\mathcal{C}}(\mathbf{w}') := \arg\min_{\mathbf{v}\in\mathcal{C}} \|\mathbf{v} - \mathbf{w}'\|$, $\mathbf{w}^{(t+1)} := P_{\mathcal{C}}[\mathbf{w}^{(t)} - \gamma \nabla \mathcal{L}(\mathbf{w}^{(t)})]$

Constrained → Unconstrained Problems

$\min_{\mathbf{w}\in\mathcal{C}} \mathcal{L}(\mathbf{w}) \sim \min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) + P(\mathbf{w})$

Implementation Issues

Optimality: 1st order: if $\mathcal{L}$ is cvx and $\nabla \mathcal{L}(w^*) = 0 \to$ global

2nd order: $\mathcal{L}$ potentially non-convex, if $\nabla \mathcal{L}(w^*) = 0$, $\nabla^2 \mathcal{L}(w^*) \ge 0 \to w$ local minimum ($\nabla^2 \mathcal{L}(\mathbf{w}) := \frac{\partial^2 \mathcal{L}}{\partial \mathbf{w}\partial \mathbf{w}^\top}(\mathbf{w})$ expensive)

3 Least Squares (Linear regression + MSE)

$\mathcal{L}(\mathbf{w}) = \frac{1}{2N}\sum_{n=1}^N (y_n - \mathbf{x}_n^\top \mathbf{w})^2 = \frac{1}{2N}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w})$,

- Proof of convexity:

1) Composition of a linear function with a convex function (the square function).

2) Verify definition, for any $\lambda \in [0,1]$ and $\mathbf{w}, \mathbf{w}'$: $\mathcal{L}(\lambda \mathbf{w} + (1-\lambda)\mathbf{w}') - (\lambda \mathcal{L}(\mathbf{w}) + (1-\lambda)\mathcal{L}(\mathbf{w}')) \le 0$.

LHS $= -\frac{1}{2N}\lambda(1-\lambda)\|\mathbf{X}(\mathbf{w} - \mathbf{w}')\|^2 < 0$

3) Hessian positive semidefinite (all its eigenvalues are non-negative).

$\mathbf{H}(\mathbf{w}) = \frac{1}{2N}\nabla^2((\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w})) = \frac{1}{N}\mathbf{X}^\top \mathbf{X}$

Singular value decomposition (SVD): $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$ U and V are orthogonal matrices, and S is a diagonal matrix with the singular values σ_i on the diagonal.

- Now find its minimum $\nabla \mathcal{L}(\mathbf{w}) = -\frac{1}{N}\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0}$

Closed form (if Gram matrix $\mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{D\times D}$ is invertible)

$\mathbf{w}^* = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$

Invertibility and Uniqueness

- $\mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{D\times D}$ invertible iff $\text{rank}(\mathbf{X}) = D$.
- Proof: assume $\text{rank}(\mathbf{X}) < D \Rightarrow \exists \mathbf{u} \ne \mathbf{0}$ s.t. $\mathbf{X}\mathbf{u} = \mathbf{0} \Rightarrow \mathbf{X}^\top \mathbf{X}\mathbf{u} = \mathbf{0} \Rightarrow \text{rank}(\mathbf{X}^\top \mathbf{X}) < D \Rightarrow \mathbf{X}^\top \mathbf{X}$ is not invertible.

Rank Deficiency and Ill-Conditioning

- In practice, $\mathbf{X}$ is often rank deficient.
- If $D > N$, we always have $\text{rank}(\mathbf{X}) < D$ (row rank = col. rank)
- If $D \le N$, but some of the columns $\mathbf{x}_i$ are (nearly) collinear, then the matrix is illconditioned → numerical issues

4 Maximum Likelihood

Gaussian distribution and independence

$p(y \mid \mu, \sigma^2) = \mathcal{N}(y \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp[-\frac{(y-\mu)^2}{2\sigma^2}]$, $\mathbf{\Sigma}$ psd.

$\mathcal{N}(\mathbf{y} \mid \boldsymbol{\mu}, \mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D \det(\mathbf{\Sigma})}}\exp[-\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu})^\top \mathbf{\Sigma}^{-1}(\mathbf{y} - \boldsymbol{\mu})]$

- Two RVs X, Y independent when $p(x, y) = p(x)p(y)$.

A probabilistic model for least-squares

- $y_n = \mathbf{x}_n^\top \mathbf{w} + \epsilon_n$ where the ϵ_n zero mean Gaussian RV

$p(\mathbf{y} \mid \mathbf{X}, \mathbf{w}) = \prod_{n=1}^N p(y_n \mid \mathbf{x}_n, \mathbf{w}) = \prod_{n=1}^N \mathcal{N}(y_n \mid \mathbf{x}_n^\top \mathbf{w}, \sigma^2)$

Defining cost with log-likelihood

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

Maximum-likelihood estimator (MLE)

$\arg\min_{\mathbf{w}} \mathcal{L}_{\text{MSE}}(\mathbf{w}) = \arg\max_{\mathbf{w}} \mathcal{L}_{\text{LL}}(\mathbf{w})$.

Properties of MLE

- MLE is a sample approximation to the expected log-likelihood: $\mathcal{L}_{\text{LL}}(\mathbf{w}) \approx \mathbb{E}_{p(y,\mathbf{x})}[\log p(y \mid \mathbf{x}, \mathbf{w})]$
- MLE is consistent, i.e., it will give us the correct model assuming that we have a sufficient amount of data. $\mathbf{w}_{\text{MLE}} \longrightarrow^p \mathbf{w}_{\text{true}}$
- The MLE is asymptotically normal, i.e., $(\mathbf{w}_{\text{MLE}} - \mathbf{w}_{\text{true}}) \longrightarrow^d \frac{1}{\sqrt{N}}\mathcal{N}(\mathbf{w}_{\text{MLE}} \mid \mathbf{0}, \mathbf{F}^{-1}(\mathbf{w}_{\text{true}}))$

where $\mathbf{F}(\mathbf{w}) = -\mathbb{E}_{p(\mathbf{y})}[\frac{\partial^2 \mathcal{L}}{\partial \mathbf{w}\partial \mathbf{w}^\top}]$ is the Fisher information.

- MLE is efficient, i.e. it achieves the Cramer-Rao lower bound. Covariance $(\mathbf{w}_{\text{MLE}}) = \mathbf{F}^{-1}(\mathbf{w}_{\text{true}})$

Laplace distribution $p(y_n \mid \mathbf{x}_n, \mathbf{w}) = \frac{1}{2b}e^{-\frac{1}{b}|y_n - \mathbf{x}_n^\top \mathbf{w}|}$

5 Regularization: $\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) + \Omega(\mathbf{w})$

Ridge Regression: L_2-Regularization + $\mathcal{L}_{MSE}$

- Euclidean norm (L_2 norm) $\Omega(\mathbf{w}) = \lambda\|\mathbf{w}\|_2^2$, $\|\mathbf{w}\|_2^2 = \sum_i w_i^2$.

$\min_{\mathbf{w}} \frac{1}{2N}\sum_{n=1}^N [y_n - \mathbf{x}_n^\top \mathbf{w}]^2 + \lambda\|\mathbf{w}\|_2^2 = \min_{\mathbf{w}} \frac{1}{2N}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|_2^2$

Explicit solution for w:

$\nabla = \nabla \mathcal{L} + \nabla \Omega = -\frac{1}{N}X^T(y - Xw) + \lambda 2w = 0$

$\Rightarrow \mathbf{w}^*_{\text{ridge}} = (\mathbf{X}^\top \mathbf{X} + \lambda' \mathbf{I})^{-1}\mathbf{X}^\top \mathbf{y}$, $(\frac{\lambda'}{2N} = \lambda)$

Ridge Regression Fights Ill-Conditioning

- Lifting the eigenvalues: The eigenvalues of $(\mathbf{X}^\top \mathbf{X} + \lambda' \mathbf{I})$ are all at least λ' and so the inverse always exists.
- Proof: Write the Eigenvalue decomposition of $\mathbf{X}^\top \mathbf{X}$ as $\mathbf{U}\mathbf{S}\mathbf{U}^\top$, $\mathbf{X}^\top \mathbf{X} + \lambda' \mathbf{I} = \mathbf{U}\mathbf{S}\mathbf{U}^\top + \lambda' \mathbf{U}\mathbf{I}\mathbf{U}^\top = \mathbf{U}[\mathbf{S} + \lambda' \mathbf{I}]\mathbf{U}^\top$

L_1-Regularization: The Lasso (L_1-norm + $\mathcal{L}_{\text{MSE}}$)

$\min_{\mathbf{w}} \frac{1}{2N}\sum_{n=1}^N [y_n - \mathbf{x}_n^\top \mathbf{w}]^2 + \lambda\|\mathbf{w}\|_1$ where $\|\mathbf{w}\|_1 := \sum_i |w_i|$

6 Model Selection

Generalization Error

- Expected error over all samples drawn from distribution $\mathscr{D}$: $L_{\mathscr{D}}(f) = \mathbb{E}_{(x,y)\sim\mathscr{D}}[\ell(y, f(x))]$

Empirical Error

- approximate the true error by averaging the loss function over the dataset: $L_S(f) = \frac{1}{|S|}\sum_{(x_n,y_n)\in S} \ell(y_n, f(x_n))$, (RV)
- unbiased estimator of the true error
- Law of large number: $L_S(f) \underset{|S|\to\infty}{\to} L_{\mathscr{D}}(f)$ but fluctuations!

Generalization gap $|L_{\mathscr{D}}(f) - L_S(f)|$

- Claim: given a model f and a test set $S_{\text{test}} \sim \mathscr{D}$ i.i.d. (not used to learn f) and a loss $\ell(\cdot,\cdot) \in [a, b]$:

$\mathbb{P}[|L_{\mathscr{D}}(f) - L_{S_{\text{test}}}(f)| \ge \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2|S_{\text{test}}|}}] \le \delta$

- Error ↓ as $\mathcal{O}(1/\sqrt{|S_{\text{test}}|})$ with nb. of test points
- High probability bound: δ is only in the ln

The proof relies only on concentration inequalities

- $(x_n, y_n) \in S_{\text{test}}$ are chosen independently, the associated losses $\Theta_n = \ell(y_n, f(x_n)) \in [a, b]$ given a fixed model f, are also i.i.d. random variables
- Empirical loss: $\frac{1}{N}\sum_{n=1}^N \Theta_n = \frac{1}{N}\sum_{n=1}^N \ell(y_n, f(x_n)) = L_{S_{\text{test}}}(f)$
- True loss: $\mathbb{E}[\Theta_n] = \mathbb{E}[\ell(y_n, f(x_n))] = L_{\mathscr{D}}(f)$

Hoeffding inequality: a simple concentration bound

- Claim: Let $\Theta_1, \ldots, \Theta_N$ be a sequence of i.i.d. random variables with mean $\mathbb{E}[\Theta]$ and range $[a, b]$. Then, for any $\varepsilon > 0$

$\mathbb{P}[|\frac{1}{N}\sum_{n=1}^N \Theta_n - \mathbb{E}[\Theta]| \ge \varepsilon] \le 2e^{-2N\varepsilon^2/(b-a)^2}$

- Concentration bound: the empirical mean is concentrated around its mean

A. Use it with $\Theta_n = \ell(y_n, f(x_n))$ B. Equating $\delta = 2e^{-2|S_{\text{test}}|\varepsilon^2/(b-a)^2}$

we get $\varepsilon = \sqrt{\frac{(b-a)^2 \ln(2/\delta)}{2|S_{\text{test}}|}}$

How far is each of the K test errors $L_{S_{\text{test}}}(f_k)$ from the true $L_{\mathscr{D}}(f_k)$?

- Claim: we can bound the maximum deviation for all K candidates, by $\mathbb{P}[\max_k |L_{\mathscr{D}}(f_k) - L_{S_{\text{test}}}(f_k)| \ge \sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2|S_{\text{test}}|}}] \le \delta$
- The error decreases as $\mathcal{O}(1/\sqrt{|S_{\text{test}}|})$ with the number test points
- test K hyper-parameters, the error only goes up by $\sqrt{\ln(K)}$

$\Rightarrow$ can test many different models without incurring a large penalty

- It can be extended to infinitely many models

Proof: A simple union bound

- Special case $K = 1$ $\mathbb{P}[\max_k |L_{\mathscr{D}}(f_k) - L_{S_{\text{test}}}(f_k)| \ge \varepsilon] = \mathbb{P}[\cup_k |L_{\mathscr{D}}(f_k) - L_{S_{\text{test}}}(f_k)| \ge \varepsilon] \le \sum_k \mathbb{P}[|L_{\mathscr{D}}(f_k) - L_{S_{\text{test}}}(f_k)| \ge \varepsilon] \le 2Ke^{-2N\varepsilon^2/(b-a)^2}$
- $\delta = 2Ke^{-2N\varepsilon^2/(b-a)^2} \Rightarrow \varepsilon = \sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2N}}$
- Let $k^* = \text{argmin}_k L_{\mathscr{D}}(f_k)$ (smallest true risk) and $\hat{k} = \text{argmin}_k L_{S_{\text{test}}}(f_k)$ (smallest empirical risk) then:

$\mathbb{P}[L_{\mathscr{D}}(f_{\hat{k}}) \ge L_{\mathscr{D}}(f_{k^*}) + 2\sqrt{\frac{(b-a)^2 \ln(2K/\delta)}{2|S_{\text{test}}|}}] \le \delta$

7 Bias Variance Decomposition

- Data model: output perturbed by some noise $y = f(x) + \varepsilon$, $\varepsilon \sim \mathscr{D}_\varepsilon \approx$ i.i.d. independent of x $\mathbb{E}[\varepsilon] = 0$

Error Decomposition $\mathbb{E}_{(x,y)\sim\mathscr{D}}[(y - f_S(x))^2]$

- Consider the expected error of f_S for a fixed element x_0: $L(f_S) = \mathbb{E}_{\varepsilon\sim\mathscr{D}_\varepsilon}[(f(x_0) + \varepsilon - f_S(x_0))^2]$, (RV from train set S)
- Run experiment many times → average and the variance of the predictions $(f_{S_1}, \cdots, f_{S_k})$ over these multiple runs

A decomposition in three terms

Interested in the expectation of the true risk over training set S

$\mathbb{E}_{S\sim\mathscr{D}}[L(f_S)] = \mathbb{E}_{S\sim\mathscr{D}}[\mathbb{E}_{\varepsilon\sim\mathscr{D}_\varepsilon}[(f(x_0) + \varepsilon - f_S(x_0))^2]]$

$= \mathbb{E}_{S\sim\mathscr{D},\varepsilon\sim\mathscr{D}_\varepsilon}[(f(x_0) + \varepsilon - f_S(x_0))^2] = \mathbb{E}_{\varepsilon\sim\mathscr{D}_\varepsilon}[\varepsilon^2] + 2\mathbb{E}_{S\sim\mathscr{D},\varepsilon\sim\mathscr{D}_\varepsilon}[\varepsilon(f(x_0) - f_S(x_0))] + \mathbb{E}_{S\sim\mathscr{D}}[(f(x_0) - f_S(x_0))^2]$

Using that $\mathbb{E}_{\varepsilon\sim\mathscr{D}}[\varepsilon] = 0$ and $\varepsilon \perp\!\!\!\perp S$:

$\bullet \mathbb{E}_{\varepsilon\sim\mathscr{D}_\varepsilon}[\varepsilon^2] = \text{Var}_{\varepsilon\sim\mathscr{D}_\varepsilon}[\varepsilon]$ $\bullet \mathbb{E}_{S\sim\mathscr{D},\varepsilon\sim\mathscr{D}_\varepsilon}[\varepsilon(f(x_0) - f_S(x_0))] = \mathbb{E}_{\varepsilon\sim\mathscr{D}_\varepsilon}[\varepsilon] \times \mathbb{E}_{S\sim\mathscr{D}}[f(x_0) - f_S(x_0)] = 0$

$\Longrightarrow \mathbb{E}_{S\sim\mathscr{D},\varepsilon\sim\mathscr{D}_\varepsilon}[(f(x_0) + \varepsilon - f_S(x_0))^2] = \text{Var}_{\varepsilon\sim\mathscr{D}_\varepsilon}[\varepsilon] + \mathbb{E}_{S\sim\mathscr{D}}[(f(x_0) - f_S(x_0))^2]$

Trick: we add and subtract the constant term $\mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)]$, where S' is a second training set independent from S

$\mathbb{E}_{S\sim\mathscr{D}}[(f(x_0) - f_S(x_0))^2] = \mathbb{E}_{S\sim\mathscr{D}}[(f(x_0) - \mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)] + \mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)] - f_S(x_0))^2]$

$= \mathbb{E}_{S\sim\mathscr{D}}[(f(x_0) - \mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)])^2] + \mathbb{E}_{S\sim\mathscr{D}}[(\mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)] - f_S(x_0))^2]$
$+ 2(f(x_0) - \mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)])(\mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)] - f_S(x_0))]$

Cross-term:

$\mathbb{E}_{S\sim\mathscr{D}}[(f(x_0) - \mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)]) \cdot (\mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)] - f_S(x_0))]$
$= (f(x_0) - \mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)]) \cdot \mathbb{E}_{S\sim\mathscr{D}}[(\mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)] - f_S(x_0))]$
$= (f(x_0) - \mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)]) \cdot (\mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)] - \mathbb{E}_{S\sim\mathscr{D}}[f_S(x_0)]) = 0$
$\Longrightarrow \mathbb{E}_{S\sim\mathscr{D}}[(f(x_0) - f_S(x_0))^2] = (f(x_0) - \mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)])^2 + \mathbb{E}_{S\sim\mathscr{D}}[(\mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)] - f_S(x_0))^2]$

Bias-Variance Decomposition

- We obtain the following decomposition into three positive terms:

$\mathbb{E}_{S\sim\mathscr{D},\varepsilon\sim\mathscr{D}_\varepsilon}[(f(x_0) + \varepsilon - f_S(x_0))^2] = \text{Var}_{\varepsilon\sim\mathscr{D}_\varepsilon}[\varepsilon] + (f(x_0) - \mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)])^2 + \mathbb{E}_{S\sim\mathscr{D}}[(f_S(x_0) - \mathbb{E}_{S'\sim\mathscr{D}}[f_{S'}(x_0)])^2]$

= Noise variance + Bias + Var.

Bias Variance tradeoff

8 Classification

- Loss function: (0-1 Loss) $\ell(y, y') = 1_{y\ne y'} = \begin{cases} 1 & \text{if } y \ne y' \\ 0 & \text{if } y = y' \end{cases}$
- True risk for the classification: $L_{\mathscr{D}}(f) = \mathbb{E}_{\mathscr{D}}[1_{y\ne f(X)}] = \mathbb{P}_{\mathscr{D}}[Y \ne f(X)]$
- minimize $L_{\mathscr{D}}(f)$

Bayes classifier f, $f_* = \arg\min L_{\mathscr{D}}(f)$

- Claim: $f_*(x) = \arg\max_{y\in\{-1,1\}} \mathbb{P}(Y = y \mid X = x)$

Proof of the Bayes classifier

- Claim 1: $\forall x \in \mathscr{X}$, $f_*(x) \in \arg\min_{y\in\mathscr{Y}} \mathbb{P}(Y \ne y \mid X = x) \Longrightarrow f_* \in \arg\min_{f:\mathscr{X}\to\mathscr{Y}} L_{\mathscr{D}}(f)$
- Claim 2: $f_*(x) = \arg\min_{y\in\mathscr{Y}} \mathbb{P}(Y \ne y \mid X = x)$

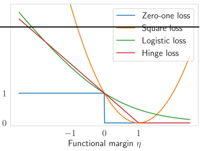
Classification by empirical risk minimization

$\min_{f:\mathscr{X}\to\mathscr{Y}} L_{\text{train}}(f) := \frac{1}{N}\sum_{n=1}^N 1_{f(x_n)\ne y_n} = \frac{1}{N}\sum_{n=1}^N 1_{y_n f(x_n)\le 0}$

- PROBLEM: L_{train} is not convex because:

The set of classifiers is not convex because $\mathscr{Y}$ is discrete

The indicator function 1 is not convex because it is not continuous



9 Logistic Regression

Logistic function: $\sigma(\eta) := \frac{e^\eta}{1+e^\eta} = \frac{1}{1+e^{-\eta}}$, $1 - \sigma(\eta) = (1 + e^\eta)^{-1}$, $\sigma'(\eta) = \sigma(\eta)(1 - \sigma(\eta))$

Logistic Regression

$p(1 \mid x) := \mathbb{P}(Y = 1 \mid X = x) = \sigma(x^\top w + w_0)$

$p(0 \mid x) := \mathbb{P}(Y = 0 \mid X = x) = 1 - \sigma(x^\top w + w_0)$

If $p(1 \mid x) \ge 1/2$, you predict the class 1, else class 0

MLE for logistic regression

- Assumption: The inputs $\mathbf{X}$ do not depend on the parameter w we choose:

$\mathscr{L}(w) = p(\mathbf{y}, \mathbf{X} \mid w) = p(\mathbf{X} \mid w)p(\mathbf{y} \mid \mathbf{X}, w)$

$\underset{\mathbf{X}\perp\!\!\!\perp w}{=} p(\mathbf{X})p(\mathbf{y} \mid \mathbf{X}, w)p(\mathbf{y} \mid \mathbf{X}, w) = \prod_{n=1}^N p(y_n \mid x_n, w) = \prod_{n:y_n=1} p(y_n = 1 \mid x_n, w) \times \prod_{n:y_n=0} p(y_n = 0 \mid x_n, w) = \prod_{n=1}^N \sigma(x_n^\top w)^{y_n}[1 - \sigma(x_n^\top w)]^{1-y_n}$

$\mathscr{L}(w) \propto \prod_{n=1}^N \sigma(x_n^\top w)^{y_n}[1 - \sigma(x_n^\top w)]^{1-y_n}$

Minimum of the Negative Log Likelihood (NLL)

$-\log(p(\mathbf{y} \mid \mathbf{X}, w)) = -\log\left(\prod_{n=1}^N \sigma(x_n^\top w)^{y_n}[1 - \sigma(x_n^\top w)]^{1-y_n}\right) = \sum_{n=1}^N -y_n x_n^\top w + \log\left(1 + e^{x_n^\top w}\right)$

$w_* = \arg\min L(w) := \frac{1}{N}\sum_{n=1}^N -y_n x_n^\top w + \log\left(1 + e^{x_n^\top w}\right)$

- NLL is equivalent to ERM for the logistic loss
- $y \in \{0, 1\}$: $\ell(y, g(x)) = -yg(x) + \log(1 + \exp(g(x)))$
- $y \in \{-1, 1\}$: $\ell(y, g(x)) = \log(1 + \exp(-yg(x)))$

Gradient of the negative log likelihood

$\nabla L(w) = \nabla\left[\frac{1}{N}\sum_{n=1}^N \log\left(1 + e^{x_n^\top w}\right) - y_n x_n^\top w\right] = \frac{1}{N}\sum_{n=1}^N \frac{e^{x_n^\top w} x_n}{1+e^{x_n^\top w}} - y_n x_n = \frac{1}{N}\sum_{n=1}^N (\sigma(x_n^\top w) - y_n) x_n$

$\nabla L(w) = \frac{1}{N}\mathbf{X}^\top(\sigma(\mathbf{X}w) - \mathbf{y})$

Convexity of the loss function L

- The Hessian $\nabla^2 L$, $\frac{\partial^2}{\partial w_i \partial w_j} L(w)$, is psd $\Rightarrow$ convex

$\nabla^2 L(w) = \frac{1}{N}\sum_{n=1}^N \sigma(x_n^\top w)(1 - \sigma(x_n^\top w)) x_n x_n^\top$

$\nabla^2 L(w) = \frac{1}{N}\mathbf{X}^\top S\mathbf{X}$ where $S = \text{diag}[\sigma(x_n^\top w)(1 - \sigma(x_n^\top w))] \succcurlyeq 0$

$\Rightarrow$ L is convex since $\nabla^2 L(w) \succcurlyeq 0$

Newton's method uses second order information

- Newton's method minimizes the quadratic approximation:

$L(w) \sim L(w_t) + \nabla L(w_t)^\top(w - w_t) + \frac{1}{2}(w - w_t)^\top \nabla^2 L(w_t)(w - w_t) := \phi_t(w)$

$\tilde{w} = \arg\min \phi_t(w) \Rightarrow \nabla L(w_t) + \nabla^2 L(w_t)(\tilde{w} - w_t) = 0$

- Newton's method: $w_{t+1} = w_t - \gamma_t \nabla^2 L(w_t)^{-1}\nabla L(w_t)$
- Step-size needed to ensure convergence (damped Newton's method)
- Convergence faster than GD but comp. complex. higher.

Problem when the data are linearly separable

$\inf_w L(w) = 0 = \lim_{\alpha\to\infty} L(\alpha \cdot \bar{w}) \Rightarrow$ the weights will go to ∞

- Solution: add a ℓ_2-regularization (Ridge logistic regression)

$\frac{1}{N}\sum_{n=1}^N -y_n x_n^\top w + \log\left(1 + e^{x_n^\top w}\right) + \frac{\lambda}{2}\|w\|_2^2$

- Optimization perspective: stabilize the optimization process
- Statistical perspective: avoid overfitting

10 Support Vector Machines

- Define a hyperplane as $\{x : w^\top x = 0\}$ where $\|w\| = 1$
- Prediction: $f(x) = \text{sign}(x^\top w)$
- Claim: distance x_0–hyperplane defined by w is $|w^\top x_0|$
- Proof: $\min_{u:w^\top u=0} \|x_0 - u\|$. Let $v = x_0 - w^\top x_0 w$ then by the Pythagorean theorem for any u s.t. $w^\top u = 0$, $\|x_0 - u\|^2 = (w^\top x_0)^2 + \|v - u\|^2 \ge (w^\top x_0)^2$

Hard-SVM rule: max-margin separating hyperplane

- Margin of a hyperplane: $\min_{n\le N} |w^\top x_n|$
- Max-margin separating hyperplane: $\max_{w,\|w\|=1} \min_{n\le N} |w^\top x_n|$ s.t. $\forall n, y_n x_n^\top w \ge 0$
- Equivalent to: $\max_{M\in\mathbb{R},w,\|w\|=1} M$ s.t. $\forall n, y_n x_n^\top w \ge M$
- Also equivalent to: $\min_w \frac{1}{2}\|w\|^2$ such that $\forall n, y_n x_n^\top w \ge 1$

Soft SVM: when training set is not linearly separable

- Maximize margin but allow some constraints to be violated
- Introduce positive slack variables $\xi_1, \cdots, \xi_N$ and replace the constraints with $y_n x_n^\top w \ge 1 - \xi_n$

$\min_{w,\xi} \frac{\lambda}{2}\|w\|^2 + \frac{1}{N}\sum_{n=1}^N \xi_n$ s.t. $\forall n, y_n x_n^\top w \ge 1 - \xi_n$ and $\xi_n \ge 0$

- Equivalent: $\min_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{N}\sum_{n=1}^N [1 - y_n x_n^\top w]_+$, (a hinge loss)
- Proof: 1) If $y_n x_n^\top w \ge 1$, then $\xi_n = 0$, 2) If $y_n x_n^\top w < 1$, $\xi_n = 1 - y_n x_n^\top w$, therefore $\xi_n = [1 - y_n x_n^\top w]_+$

ERM for the hinge loss with ridge regularization

$\min_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{N}\sum_{n=1}^N [1 - y_n x_n^\top w]_+$

1) Choose direction of w s.t. $w^\perp$ acts as a separating hyperplane

2) Adjust scale of w to ensure that no point lies with the margin

3) Select the hyperplane with the largest margin

- Optimization: Convex (but non-smooth) objective which can be minimized with: 1) SubGD method, 2) StochSubGD method

Convex duality

- Define $G(w, \alpha)$ s.t. $\min_w L(w) = \min_w \max_\alpha G(w, \alpha)$
- Primal pb.: $\min_w \max_\alpha G(w, \alpha)$, dual pb.: $\max_\alpha \min_w G(w, \alpha)$

$G(w, \alpha)$ for SVM

$[z]_+ = \max(0, z) = \max_{\alpha\in[0,1]} \alpha z$

$[1 - y_n x_n^\top w]_+ = \max_{\alpha_n\in[0,1]} \alpha_n(1 - y_n x_n^\top w)$

$\min_w L(w) = \min_w \max_{\alpha\in[0,1]^n} \frac{1}{N}\sum_{n=1}^N \alpha_n(1 - y_n x_n^\top w) + \frac{\lambda}{2}\|w\|_2^2$

- The function G is convex in w and concave in α

Min max interchangeable

$\max_\alpha \min_w G(w, \alpha) \le \min_w \max_\alpha G(w, \alpha)$

- Equality if G is convex in w, concave in α and the domains of w and α are convex and compact
- Proof: $\min_w G(\alpha, w) \le G(\alpha, w')$ $\forall w'$

$\max_\alpha \min_w G(\alpha, w) \le \max_\alpha G(\alpha, w')$ $\forall w'$

$\max_\alpha \min_w G(\alpha, w) \le \min_{w'} \max_\alpha G(\alpha, w')$

Application to SVM: the condition is met

$\min_w L(w) = \max_{\alpha\in[0,1]^n} \min_w \frac{1}{N}\sum_{n=1}^N \alpha_n(1 - y_n x_n^\top w) + \frac{\lambda}{2}\|w\|_2^2$

- Minimizer computation: $\mathbf{Y} = \text{diag}(\mathbf{y})$

$\nabla_w G(w, \alpha) = -\frac{1}{N}\sum_{n=1}^N \alpha_n y_n x_n + \lambda w = 0 \Longrightarrow w(\alpha) = \frac{1}{\lambda N}\sum_{n=1}^N \alpha_n y_n x_n = \frac{1}{\lambda N}\mathbf{X}^\top \mathbf{Y}\alpha$

- Dual optimization problem: $\min_w L(w) = \max_{\alpha\in[0,1]^n} \frac{1}{N}\sum_{n=1}^N \alpha_n\left(1 - \frac{1}{\lambda N}y_n x_n^\top \mathbf{X}^\top \mathbf{Y}\alpha\right) + \frac{1}{2\lambda N^2}\|\mathbf{X}^\top \mathbf{Y}\alpha\|_2^2 = \max_{\alpha\in[0,1]^n} \frac{1^\top \alpha}{N} - \frac{1}{2\lambda N^2}\alpha^\top \underbrace{\mathbf{Y}\mathbf{X}\mathbf{X}^\top \mathbf{Y}}_{\text{PSD matrix}}\alpha$

1. Differentiable Concave Problem
2. Kernel Matrix Dependency $K = \mathbf{X}\mathbf{X}^\top \in \mathbb{R}^{N\times N}$: no d
3. Dual Formulation Insight: α is typically sparse and non-zero exclusively for the training examples that are crucial in determining the decision boundary

Interpretation of the dual formulation

$\forall (x_n, y_n) \exists \alpha_n$ given by: $\max_{\alpha_n\in[0,1]} \alpha_n(1 - y_n x_n^\top w)$

- x_n correct side, outside margin, $1 - y_n x_n^\top w < 0 \Rightarrow \alpha_n = 0$
- x_n correct side, on the margin, $1 - y_n x_n^\top w = 0 \Rightarrow \alpha_n \in [0, 1]$
- x_n strictly inside the margin or incorrect side, $1 - y_n x_n^\top w > 0 \Rightarrow \alpha_n = 1$

- points where $\alpha_n > 0$:= support vectors

$w = \frac{1}{\lambda N}\sum_{n=1}^N \alpha_n y_n x_n$

11 KNN Curse of dimensionality

- Claim 1: As the dimensionality grows, fixed-size training sets cover a diminishing fraction of the input space
- Assume the data $x \sim \mathcal{U}([0, 1]^d)$
- Box ⬚ around the center x_0 of size r, $\mathbb{P}(x \in ⬚) = r^d := \alpha$
- Claim 2: In high-dimension, data-points are far from each other.
- Consider N i.i.d. points uniform in the $[0, 1]^d$

$\mathbb{P}(\exists x_i \in ⬚) \ge 1/2 \Longrightarrow r \ge \left(1 - \frac{1}{2^{1/N}}\right)^{1/d}$

Generalization bound for 1-NN

- Setup: $(X, Y) \sim \mathscr{D}$ over $\mathscr{X} \times \mathscr{Y} = [0, 1]^d \times \{0, 1\}$
- Goal: Bound the clf error: $L(f) = \mathbb{P}_{(X,Y)\sim D}(Y \ne f(X))$
- Baseline: ($N \to \infty$, $1 -$ NN is competitive with Bayes classifier)

- Bayes classifier: minimizes L over all classifiers $f_*(x) = 1_{\eta(x)\ge 1/2}$ where $\eta(x) = \mathbb{P}(Y = 1 \mid X = x)$
- Bayes risk: represents the minimum probability of misclassification $L(f_*) = \mathbb{P}(f_*(X) \ne Y) = \mathbb{E}_{X\sim D_X}[\min\{\eta(X), 1 - \eta(X)\}]$

- Assumption: $\exists c \ge 0, \forall x, x' \in \mathscr{X}$: $|\eta(x) - \eta(x')| \le c\|x - x'\|_2$

$\Rightarrow$ Nearby points are likely to share the same label

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

- For constant d and $N \to \infty$: $\mathbb{E}_{S_{\text{train}}}[L(f_{S_{\text{train}}})] \le 2L(f_*)$
- constant error, $N \propto d^{(d+1)/2}$ - curse of dimensionality
- Interpolation method can generalize well

- Claim: $\mathbb{P}(Y' \ne Y) \le 2\min\{\eta(x), 1 - \eta(x)\} + c\|x - x'\|$

Bound on the geometric term (I)

- Consider a fresh sample $X \sim \mathscr{D}$ and denote $p_k = \mathbb{P}(X \in \text{Box}_k)$
- Consider the box which contains X. Two options:

- The box contains an element of S_{train} X has a neighbor in S_{train} at distance at most $\sqrt{d}\varepsilon$. It happens with probability $1 - (1 - p_k)^N$
- There is no element of S_{train} . The nearest neighbor of X can be at worst at a distance $\sqrt{d}$ It happens with probability $(1 - p_k)^N$

Bound on the geometric term (II)

$\mathbb{E}[\|X - \text{nbh}(X)\|] \le \sum_k p_k\left[(1 - p_k)^N\sqrt{d} + (1 - (1 - p_k)^N)\sqrt{d}\varepsilon\right]$

- Claim: The bound is derived by optimizing over p_k and ε

- If p_k is large: it is likely that we pick that box but it is also likely that we find a training point in that box
- If p_k is small, this scenario occurs infrequently

Nearest Neighbors is a local averaging method

- Local averaging methods aim to approximate the Bayes predictor directly - without the need for optimization
- In the case of nearest neighbors: $\hat{p}(y \mid x) = \sum_{n=1}^N \hat{w}_n(x)1_{y=y_n}$, where $\hat{w}(x) = 1/k$ for the k nearest neighbors (0 otherwise)

12 Kernel Regression

$\mathcal{W}_{*\text{ridge}} = \frac{1}{N}(\underbrace{\frac{1}{N}\mathbf{X}^\top \mathbf{X} + \lambda I_d}_{d\times d})^{-1}\mathbf{X}^\top \mathbf{y} = \frac{1}{N}\mathbf{X}^\top(\underbrace{\frac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda I_N}_{N\times N})^{-1}\mathbf{y}$

- Proof: Let $P \in \mathbb{R}^{m\times n}$ and $Q \in \mathbb{R}^{n\times m}$, $P(QP + I_n) = PQP + P = (PQ + I_m)P$, $(PQ + I_m)^{-1}P = P(QP + I_n)^{-1}$, $P = \mathbf{X}^\top$, $Q = \frac{1}{\lambda N}\mathbf{X}$

Usefulness of the alternative form

1) Original $\to \mathcal{O}(d^3 + Nd^2)$, Alternative $\to \mathcal{O}(N^3 + dN^2)$

2) Structural difference: $w_* = \mathbf{X}^\top \alpha_*$ where $\alpha_* = \frac{1}{N}\left(\frac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda I_N\right)^{-1}\mathbf{y} \Rightarrow w_* \in \text{span}\{x_1, \cdots, x_N\}$

Representer Theorem

- Claim: For any loss function ℓ, there exists $\alpha_* \in \mathbb{R}^N$ s.t. $w_* := \mathbf{X}^\top \alpha_* \in \arg\min_w \frac{1}{N}\sum_{n=1}^N \ell(x_n^\top w, y_n) + \frac{\lambda}{2}\|w\|^2$
- Meaning: $\exists w_* \in \text{span}\{x_1, \cdots, x_N\}$
- more general than LS, Kernel SVM, Kernel LS, and Kernel PCA

Proof of the representer theorem

- Let $w_* = \min_w \frac{1}{N}\sum_{n=1}^N \ell(x_n^\top w, y_n) + \frac{\lambda}{2}\|w\|^2 = \sum_{n=1}^N \alpha_n x_n + u$ where $u^\top x_n = 0$ for all n
- Let's define $w = w_* - u$, $\|w_*\|^2 = \|w\|^2 + \|u\|^2 \Rightarrow \|w\|^2 \le \|w_*\|^2$

$\forall n, w^\top x_n = (w_* - u)^\top x_n = w_*^\top x_n \Rightarrow \ell(x_n^\top w, y_n) = \ell(x_n^\top w_*, y_n)$

Therefore $\frac{1}{N}\sum_{n=1}^N \ell(x_n^\top w, y_n) + \frac{\lambda}{2}\|w\|^2 \le \frac{1}{N}\sum_{n=1}^N \ell(x_n^\top w_*, y_n) + \frac{\lambda}{2}\|w_*\|^2$, and w is an optimal solution for this problem.

Kernelized ridge regression

$\alpha_* = \arg\min_\alpha \frac{1}{2}\alpha^\top\left(\frac{1}{N}\mathbf{X}\mathbf{X}^\top + \lambda I_N\right)\alpha - \frac{1}{N}\alpha^\top \mathbf{y}$

Kernel matrix K with feature spaces

- When a feature map $\phi : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ is used, $(x_n)_{n=1}^N \hookrightarrow (\phi(x_n))_{n=1}^N$, $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^\top \in \mathbb{R}^{N\times N}$
- Problem: when $d \lll \tilde{d}$ computing $\phi(x)^\top \phi(x')$ costs $\mathcal{D}(\tilde{d})$

Kernel trick: Kernel function $\kappa(x, x') = \phi(x)^\top \phi(x')$

Same: Compute $\kappa(x, x')$ or map features to $\phi(x)$ and compute $\phi(x)^\top \phi(x') \Rightarrow$ compute linear classifiers in high-dimensional space without performing computations in it

Predicting with kernels: only using the kernel function

$\phi(x)^\top w_* = \phi(x)^\top \phi(\mathbf{X})^\top \alpha_* = \sum_{n=1}^N \kappa(x, x_n)\alpha_{*_n}$

$y = \phi(x)^\top w_{*\text{Linear pred in feature space}} = f_{W_*}(x)_{\text{Non linear pred in } \mathcal{X} \text{ space}}$

1. Linear Kernel: $\kappa(x, x') = x^\top x'$, $\phi(x) = x$

2. Quadratic kernel: $\kappa(x, x') = (xx')^2$, $x, x' \in \mathbb{R}$, $\phi(x) = x^2$

3. Polynomial kernel

Let $x, x' \in \mathbb{R}^3$, $\kappa(x, x') = (x_1 x_1' + x_2 x_2' + x_3 x_3')^2$

$\phi(x) = \left[x_1^2, x_2^2, x_3^2, \sqrt{2}x_1 x_2, \sqrt{2}x_1 x_3, \sqrt{2}x_2 x_3\right] \in \mathbb{R}^6$

4. Radial basis function (RBF) kernel

- Let $x, x' \in \mathbb{R}^d$ $\kappa(x, x') = e^{-(x-x')^\top(x-x')}$
- e.g. $\phi(x) = e^{-x^2}\left(\cdots, \frac{2^{k/2}x^k}{\sqrt{k!}}\cdots\right)$

Building new kernels from existing kernels

Claim 1: Positive linear combinations of kernel are kernels: $\kappa(x, x') = \alpha\kappa_1(x, x') + \beta\kappa_2(x, x')$ for $\alpha, \beta \ge 0$

Proof 1: $\phi(x) = \begin{pmatrix} \sqrt{\alpha}\phi_1(x) \\ \sqrt{\beta}\phi_2(x) \end{pmatrix} \in \mathbb{R}^{d_1+d_2}$

Claim 2: Products of kernels are kernels:

Proof 2: Let $\phi(x)^\top = (\phi_1(x)_1\phi_2(x)_1, \cdots, \phi_1(x)_1\phi_2(x)_{d_2}, \cdots, \phi_1(x)_{d_1}\phi_2(x)_1, \cdots, \phi_1(x)_{d_1}\phi_2(x)_{d_2}) \in \mathbb{R}^{d_1 d_2}$ then

$\phi(x)^\top \phi(x') = \sum_{i,j} (\phi_1(x))_i(\phi_2(x))_j(\phi_1(x'))_i(\phi_2(x'))_j = \sum_i (\phi_1(x))_i(\phi_1(x'))_i \sum_j (\phi_2(x))_j(\phi_2(x'))_j = \phi_1(x)^\top \phi_1(x')\phi_2(x)^\top \phi_2(x') = \kappa(x, x')$

Mercer's condition

- Given κ, $\exists\phi$ s.t. $\kappa(x, x') = \phi(x)^\top \phi(x')$ iff:

1) The kernel function is symmetric: $\forall x, x'$, $\kappa(x, x') = \kappa(x', x)$

2) The kernel matrix is psd for all possible input sets: $\forall N \ge 0, \forall (x_n)_{n=1}^N$, $\mathbf{K} = (\kappa(x_i, x_j))_{i,j=1}^N \succcurlyeq 0$

Proof of Mercer theorem

- If κ represents an inner product then it is symmetric and the kernel matrix is psd: $v^\top \mathrm{K}v = \sum_{i,j} v_i v_j \phi(x_i)^\top \phi(x_j) = \|\sum_i v_i \phi(x_i)\|^2$
- Define $\phi(x) = \kappa(\cdot, x)$. Define a vector space of functions by considering all linear combinations $\{\sum_i \alpha_i \kappa(\cdot, x_i)\}$. Define an inner product on this vector space by

$\left\langle \sum_i \alpha_i \kappa(\cdot, x_i), \sum_j \beta_j \kappa\left(\cdot, x_j'\right)\right\rangle = \sum_{i,j} \alpha_i \beta_j \kappa\left(x_i, x_j'\right)$

This is a valid inner product (symmetric, bilinear and positive definite, with equality holding only if $\phi(x)$ is the zero function)

Consequently $\langle \phi(x), \phi(x')\rangle = \langle \kappa(\cdot, x), \kappa(\cdot, x')\rangle = \kappa(x, x')$

13 Deep Learning: Multi-layer Neural Networks

- Learnable Parameters: biases $\left\{b^{(l)}\right\}_{l\in[L]}$ and weights $\left\{\mathbf{W}^{(l)}\right\}_{l\in[L]}$

so we learn $\mathcal{O}(dK + K^2 L) \approx \mathcal{O}(K^2 L)$ parameters

L_2 Approximation: Barron's result

- Let $f : \mathbb{R}^d \to \mathbb{R}$ and define $\hat{f}(\omega) = \int_{\mathbb{R}^d} f(x)e^{-i\omega^\top x}dx$, its Fourier transform, $\int_{\mathbb{R}^d} |\omega||\hat{f}(\omega)|d\omega \le C$ (smoothness assumption)
- Claim: $\forall n \ge 1$, $r > 0$, $\exists$ f_n: $f_n(x) = \sum_{j=1}^n c_j \phi(x^\top w_j + b_j) + c_0$, s.t. $\int_{|x|\le r} (f(x) - f_n(x))^2 dx \le \frac{(2Cr)^2}{n} \Rightarrow$ All sufficiently smooth function can be approximated by a one-hidden-layer NN

L_1 Approximation

$\int_{|x|\le r} |f(x) - f_n(x)| dx \le$ Something small

- $f : \mathbb{R} \to \mathbb{R}$ on a bounded domain, Riemann integrable - its integral can be approximated to any desired accuracy using "lower" and "upper" sums of the area of rectangles
- Approximation of the function by a sum of rectangle functions

$\int_{|x|\le r} |f(x) - f_n(x)| dx = \int_{|x|\le r} (f(x) - f_n(x)) dx = \int_{|x|\le r} f(x)dx - \int_{|x|\le r} f_n(x)dx = \int_{|x|\le r} f(x) - \sum_i \text{Area(Rectangle}_i)$

Approximation of the rectangle

- A rectangle function is equal to the sum of two step functions
- Approximate a step function with a sigmoid $\tilde{\phi}(x) = \phi(w(x - b))$,

$\Rightarrow$ NN with one hidden layer containing $2k$ nodes for a Riemann sum with k rectangles

Larger dimension: $d = 2$, same idea

- Two sigmoids can approximate an infinite rectangle function
- Four sigmoids approximate a cross

Point-wise approximations

- Def: piecewise linear (PWL) function:

$q(x) = \sum_{i=1}^m (a_i x + b_i)1_{r_{i-1}\le x<r_i}$ with $a_i r_i + b_i = a_{i+1}r_i + b_{i+1}$

- ℓ_∞-approximation result: Let f be a continuous function on $[c, d]$. $\forall$ $\varepsilon > 0$ $\exists$ q s.t. $\sup_{x\in[c,d]} |f(x) - q(x)| \le \varepsilon$

PWL functions can be written as combination of RELU

- Claim 1: Any PWL q can be rewritten as $q(x) = \tilde{a}_1 x + \tilde{b}_1 + \sum_{i=2}^m \tilde{a}_i\left(x - \tilde{b}_i\right)_+$ where $\tilde{a}_1 = a_1, \tilde{b}_1 = b_1, a_i = \sum_{j=1}^i \tilde{a}_i$ and $\tilde{b}_i = r_{i-1}$
- Claim 2: q can be implemented as a one-hidden-layer NN with RELU activation. Each term corresponds to one node:

14 Neural Network Training

$\mathbf{W}^{(1)} \in \mathbb{R}^{d\times K}$, $\mathbf{W}^{(l)} \in \mathbb{R}^{K\times K}$ for $2 \le l \le L$, $\mathbf{W}^{(L+1)} \in \mathbb{R}^K$, $b^{(l)} \in \mathbb{R}^K$ for $1 \le l \le L$, $b^{(L+1)} \in \mathbb{R}$

Compact description of output

- $x^{(l)} = f^{(l)}\left(x^{(l-1)}\right) := \phi\left(\left(\mathbf{W}^{(l)}\right)^\top x^{(l-1)} + b^{(l)}\right)$
- $y = f^{(L+1)}\left(x^{(L)}\right) := \left(\mathbf{W}^{(L+1)}\right)^\top x^{(L)} + b^{(L+1)}$

$f = f^{(L+1)} \circ f^{(L)} \circ \cdots \circ f^{(l)} \circ \cdots \circ f^{(2)} \circ f^{(1)}$

$\mathscr{L} = \frac{1}{2N}\sum_{n=1}^N \left(y_n - f^{(L+1)} \circ \cdots \circ f^{(2)} \circ f^{(1)}(x_n)\right)^2$

Chain rule

$\mathscr{L}_n = \frac{1}{2}\left(y_n - f^{(L+1)} \circ \cdots \circ f^{(l+1)} \circ \phi\left(z^{(l)}\right)\right)^2$

$\frac{\partial \mathscr{L}_n}{\partial w_{i,j}^{(l)}} = \sum_{k=1}^K \frac{\partial \mathscr{L}_n}{\partial z_k^{(l)}}\frac{\partial z_k^{(l)}}{\partial w_{i,j}^{(l)}} = \frac{\partial \mathscr{L}_n}{\partial z_j^{(l)}}\frac{\partial z_j^{(l)}}{\partial w_{i,j}^{(l)}} = \frac{\partial \mathscr{L}_n}{\partial z_j^{(l)}} \cdot x_i^{(l-1)}$ since $\frac{\partial z_k^{(l)}}{\partial w_{i,j}^{(l)}} = 0$ for $k \ne j$, since $z_j^{(l)} = \sum_{k=1}^K w_{k,j}^{(l)} x_k^{(l-1)} + b_j^{(l)}$

Derivatives computation

- Using $z_m^{(l)} = \sum_{k=1}^K w_{k,m}^{(l)} x_k^{(l-1)} + b_m^{(l)}$:

$\frac{\partial \mathscr{L}_n}{\partial b_j^{(l)}} = \sum_{k=1}^K \frac{\partial \mathscr{L}_n}{\partial z_k^{(l)}}\frac{\partial z_k^{(l)}}{\partial b_j^{(l)}} = \frac{\partial \mathscr{L}_n}{\partial z_j^{(l)}}\frac{\partial z_j^{(l)}}{\partial b_j^{(l)}} = \delta_j^{(l)}$

$\frac{\partial \mathscr{L}_n}{\partial w_{i,j}^{(l)}} = \sum_{k=1}^K \frac{\partial \mathscr{L}_n}{\partial z_k^{(l)}}\frac{\partial z_k^{(l)}}{\partial w_{i,j}^{(l)}} = \frac{\partial \mathscr{L}_n}{\partial z_j^{(l)}}\frac{\partial z_j^{(l)}}{\partial w_{i,j}^{(l)}} = \delta_j^{(l)} \cdot x_i^{(l-1)}$

Backpropagation algorithm

Forward pass $\mathcal{O}(K^2 L)$:

$x^{(0)} = x_n \in \mathbb{R}^d$, $z^{(l)} = \left(\mathbf{W}^{(l)}\right)^\top x^{(l-1)} + b^{(l)}$, $x^{(l)} = \phi\left(z^{(l)}\right)$

Backward pass $\mathcal{O}(K^2 L)$:

$\delta^{(L+1)} = z^{(L+1)} - y_n$, $\delta^{(l)} = \left(\mathbf{W}^{(l+1)}\delta^{(l+1)}\right) \odot \phi'\left(z^{(l)}\right)$

Compute the derivatives: $\frac{\partial \mathscr{L}_n}{\partial w_{i,j}^{(l)}} = \delta_j^{(l)} x_i^{(l-1)}$, $\frac{\partial \mathscr{L}_n}{\partial b_j^{(l)}} = \delta_j^{(l)}$

Batch Normalization

$\bar{z}_n^{(l)} = \frac{z^{(l)}-\mu_B^{(l)}}{\sqrt{(\sigma_B^{(l)})^2+\varepsilon}}$ (Component-wise) where $\mu_B^{(l)} = \frac{1}{M}\sum_{n=1}^{M} z_n^{(l)}$ and $\left(\sigma_B^{(l)}\right)^2 = \frac{1}{M}\sum_{n=1}^{M}\left(z_n^{(l)}-\mu_B^{(l)}\right)^2$, and $\varepsilon \in \mathbb{R}_{\geq 0}$ is a small value added for numerical stability

- Introduce learnable parameters $\gamma^{(l)}, \beta^{(l)} \in \mathbb{R}^K$ to reverse the normalization: $\hat{z}_n^{(l)} = \gamma^{(l)} \odot \bar{z}_n^{(l)} + \beta^{(l)}$
- Scale-invariance: $\mathrm{BN}\left(a \odot z_n^{(l)} + b\right) = \mathrm{BN}\left(z_n^{(l)}\right)$
- Inference: Estimate $\hat{\mu}^{(l)} = \mathbf{E}\left[\mu_B^{(l)}\right]$ and $\hat{\sigma}^{(l)} = \mathbf{E}\left[\sigma_B^{(l)}\right]$ during training, use these for inference

Layer Normalization

$\bar{z}_n^{(l)} = \frac{z^{(l)}-\mu_n^{(l)}\cdot 1_K}{\sqrt{(\sigma_n^{(l)})^2+\varepsilon}}$ where $\mu_n^{(l)} = \frac{1}{K}\sum_{k=1}^{K} z_n^{(l)}(k)$ and $\left(\sigma_n^{(l)}\right)^2 = \frac{1}{K}\sum_{k=1}^{K}\left(z_n^{(l)}(k)-\mu_n^{(l)}\right)^2$, and $\varepsilon \in \mathbb{R}_{\geq 0}$

- Learnable parameters $\gamma^{(l)}, \beta^{(l)} \in \mathbb{R}^K$: $\hat{z}_n^{(l)} = \gamma^{(l)} \odot \bar{z}_n^{(l)} + \beta^{(l)}$

- Normalize across features, independently for each observation
- common alternative, widely used for transformers and text data
- No batch dependency, use the same for training and inference

15 Convolutional Networks

$x_{n,m}^{(1)} = \sum_{k,l} f_{k,l} \cdot x_{n-k,m-l}^{(0)}$, ($f$ is the learnable filter)

- Same filter at every position - weight sharing
- Translation equivariance :shifted input results in shifted output

Training:

1. Run backpropagation as if the weights were not shared
2. Sum the gradients of all edges that share the same weight

Weight Decay: l_2-regularization

- Regularize weights without bias: $\min \mathscr{L} + \frac{\lambda}{2}\sum_l \left\|\mathbf{W}^{(l)}\right\|_F^2$
- Favors small w which can aid in generalization and opti

$\left(w_{i,j}^{(l)}\right)_{t+1} = \left(w_{i,j}^{(l)}\right)_t - \eta\nabla\mathscr{L} - \eta\lambda\left(w_{i,j}^{(l)}\right)_t = \underbrace{(1-\eta\lambda)}_{\text{weight decay}}\left(w_{i,j}^{(l)}\right)_t - \eta\nabla\mathscr{L}$

- Interaction with BatchNorm:
- $\mathrm{BN}(\mathbf{WX}) = \mathrm{BN}(\alpha\mathbf{WX})$ for $\alpha \in \mathbb{R}_{>0}$ (assuming $\varepsilon \approx 0$)

16 Transformers

- Self-Attention (SA): mixes information between tokens
- Multi-Layer Percep. (MLP): mixes information within each token
- Skip connections are widely used
- Layer normalization (LN) placed at the start of a residual branch

Text Token Embeddings

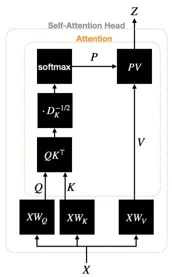
- Tokenization: split text into sequence of tokens (predefined)
- Convert each token ID $i \in \{1, \ldots, N_{vocab}\}$ into $\mathbf{w}_i \in \mathbb{R}^D$
- Matrix multiplication $\mathbf{W} \cdot \mathbf{e}_i = \mathbf{W}_{:,i} = \mathbf{w}_i$ (with $\mathbf{W} \in \mathbb{R}^{D \times N_{vocab}}$)
- $\mathbf{W}$ learned via backpropagation
- Input sequence of T tokens leads to an input matrix $X \in \mathbb{R}^{T\times D}$

Attention: learning input-dependent weighted average

- Input tokens : $V \in \mathbb{R}^{T_{in}\times D}$, Output tokens : $Z \in \mathbb{R}^{T_{out}\times D}$

$z_i = \sum_{j=1}^{T_i} p_{ij} v_j$ i.e. $Z = PV$, Weighting coefficients $P \in [0,1]^{T_{out}\times T_{in}}$ valid probability distributions over input $\sum_{j=1}^{T_{in}} p_{ij} = 1$

- Query tokens : $Q \in \mathbb{R}^{T_{out}\times D_K}$, Key tokens : $K \in \mathbb{R}^{T_{in}\times D_K}$
- Determine weight $p_{i,j}$ based on how simmilar q_i and k_j are.
- Use inner product to obtain raw similarity scores.
- Normalize with softmax (scaled temperature by $\sqrt{D_K}$) to obtain a probability distribution. (applied on each row independently)
- Scaling → uniformity at initialization and faster convergence



Self-Attention

- V, K, Q all derived from same input token sequence $X \in \mathbb{R}^{T\times D}$
- Values : $V = XW_V \in \mathbb{R}^{T\times D}$, $W_V \in \mathbb{R}^{D\times D}$
- Keys : $K = XW_K \in \mathbb{R}^{T\times D_K}$, $W_K \in \mathbb{R}^{D\times D_K}$
- Queries : $Q = XW_Q \in \mathbb{R}^{T\times D_K}$, $W_Q \in \mathbb{R}^{D\times D_K}$
- W_Q, W_V, W_K are learned params.
- $Z = \mathrm{softmax}\left(\frac{XW_QW_K^\top X^\top}{\sqrt{D_K}}\right)XW_V$

Multi-Head Self-Attention

- Run H Self-Attention "heads" in parallel $Z_h = \mathrm{softmax}\left(\frac{XW_{Q,h}W_{K,h}^\top X^\top}{\sqrt{D_K}}\right)XW_{V,h} \in \mathbb{R}^{T\times D_V}$, $W_{V,h} \in \mathbb{R}^{D\times D_V}$, $W_{K,h} \in \mathbb{R}^{D\times D_K}$, $W_{Q,h} \in \mathbb{R}^{D\times D_K}$, $Z = [Z_1,\ldots,Z_H]W_o$ where $W_O \in \mathbb{R}^{HD_V\times D}$ is learned via backprop

Positional Information

- Attention by itself does not account for the order of input
- positional encoding in the network $pos : \{1,\ldots,T\} \to \mathbb{R}^D$
- e.g. W_{pos} corresponding to each token's position t to the input embedding. $W_{pos} \in \mathbb{R}^{D\times T}$ is learned via backprop

MLP: Mixing Information within Tokens

- Apply the same transformation to each token independently: $MLP(X) = \varphi(XW_1)W_2$, $W_1, W_2 \in \mathbb{R}^{D\times D}$ learned via backprop

Output Transformations

- Single output: transfo. to a special taskspecific input token or to the average tks. Multi outputs: transfo to each token indep.

Vision Transformer Architecture

- The receptive field is the whole image after one SA layer
- ViTs require more data than CNNs, reduced inductive bias in extracting local features
- Model attends to image regions semantically relevant for clf

Encoders & Decoders

- Encoders: fixed output size and process all inputs simultaneously
- Decoders: Auto-regressively sample the next token as $x_{t+1} \sim sofimax(f(x_1,\ldots,x_t))$

17 Adversarial ML

- Standard risk: average zero-one loss over X: $R(f) = \mathbb{E}_{\mathcal{D}}\left[1_{f(X)\neq Y}\right] = \mathbb{P}_{\mathcal{D}}[f(X)\neq Y]$ i.e. minimise proba of wrong pred.
- Adversarial risk: average zero-one loss over small, worst-case perturbations of X: $R_\varepsilon(f) = \mathbb{E}_{\mathcal{D}}\left[\max_{\hat{x},\|\hat{x}-X\|\leq\varepsilon} 1_{f(\hat{x})\neq Y}\right]$

Generating adversarial examples

- Task: given an input (x, y) and a model $f : \mathcal{X} \to \{-1,1\}$ find an input $\hat{x}$ s.t.: a) $\|\hat{x} - x\| \leq \varepsilon$ b) the model f makes a mistake
- Trivial case: x already misclassified → no action required
- General case: find $\hat{x}$ such that at$f(\hat{x}) \neq y$ and $\|\hat{x}-x\| \leq \varepsilon$ i.e. $\hat{x} \in B_\varepsilon(x) \cap \{x' f(x') = -y\}$
- Optimization problem with respect to the inputs
- Problem: optimizing the indicator function is difficult: 1) The indicator function is not continuous 2) The NN prediction f outputs discrete class values $\{-1,1\}$
- Replace the difficult problem involving the indicator with a smooth problem $\max_{\hat{x},\|\hat{x}-X\|\leq\varepsilon} 1_{f(\hat{x})\neq Y} \to \max_{\hat{x},\|\hat{x}-X\|\leq\varepsilon} \ell(yg(\hat{x}))$
- decreasing, margin-based (i.e., dependent on $y * g(x)$) clf loss

White-Box attacks

- Solve $\max_{\hat{x},\|\hat{x}-X\|\leq\varepsilon} \ell(yg(\hat{x}))$ knowing g
- $\nabla_x\ell(yg(x)) = y\ell'(yg(x))\nabla_x g(x)$, with $y\ell'(yg(x)) \leq 0$ since classification losses are decreasing.
- Move in direction of $\propto -y\nabla_x g(x)$
- Interpretation $f(x) = \mathrm{sign}(g(x))$: If $y = 1$ we want to decrease $g(x)$ and follow $-\nabla_x g(x)$. If $y = -1$ we want to decrease $g(x)$ and follow $\nabla_x g(x)$
- Use ℓ not $yg(\hat{x})$: extend to multi-class clf and robust training.
- linearize the loss $\tilde{\ell}(x) := \ell(yg(x))$

$\max_{\|\hat{x}-x\|\leq\varepsilon}\tilde{\ell}(x) \approx \max_{\|\hat{x}-x\|\leq\varepsilon}\tilde{\ell}(x) + \nabla_x\tilde{\ell}(x)^T(\hat{x}-x) = \tilde{\ell}(x) + \max_{\|\hat{x}-x\|\leq\varepsilon}\nabla_x\tilde{\ell}(x)^T(\hat{x}-x) = \tilde{\ell}(x) + \max_{\|\delta\|\leq\varepsilon}\nabla_x\tilde{\ell}(x)^T\delta$

- Max inner product under a norm constraint,
- Simple problem for which we can get a closed-form solution depending on the norm used to measure the perturb size $\|\delta\|$

One-step attack

$\delta_2^\star = \varepsilon \cdot \frac{\nabla_x\tilde{\ell}(x)}{\|\nabla_x\tilde{\ell}(x)\|_2} = -\varepsilon y * \frac{\nabla_x g(x)}{\|\nabla_x g(x)\|_2} \Rightarrow \hat{x} = x - \varepsilon y \cdot \frac{\nabla_x g(x)}{\|\nabla_x g(x)\|_2}$

- Solution for the ℓ_∞ norm called **Fast Gradient Sign Method**: $\delta_\infty^\star = \varepsilon \cdot \mathrm{sign}(\nabla_x\tilde{\ell}(x)) = -\varepsilon y \cdot \mathrm{sign}(\nabla_x g(x)) \Rightarrow \hat{x} = x - \varepsilon y \cdot \mathrm{sign}(\nabla_x g(x))$

Multi-step attack

- These updates can be done iteratively and combined with a projection Π on the feasible set (i.e., balls ℓ_2 / ℓ_∞ here)
- Projected Gradient Descent (PGD attack)
- ℓ_2 norm: $\delta^{t+1} = \Pi_{B_2(\varepsilon)}\left[\delta^t + \alpha \cdot \frac{\nabla\tilde{\ell}(x+\delta^t)}{\|\nabla\tilde{\ell}(x+\delta^t)\|_2}\right]$, $\Pi_{B_2(\varepsilon)}(\delta) = \begin{cases} \varepsilon\cdot\delta/\|\delta\|_2, & \text{if } \|\delta\|_2 \geq \varepsilon \\ \delta \text{ otherwise} \end{cases}$
- ℓ_∞ norm: $\delta^{t+1} = \Pi_{B_\infty(\varepsilon)}\left[\delta^t + \alpha\cdot\mathrm{sign}(\nabla\tilde{\ell}(x+\delta^t))\right]$, $\Pi_{B_\infty(\varepsilon)}(\delta)_i = \begin{cases} \varepsilon\cdot\mathrm{sign}(\delta_i), & \text{if } |\delta_i| \geq \varepsilon \\ \delta_i \text{ otherwise} \end{cases}$
- the gradients with backprop w.r.t. inputs, not parameters!

Black-box attacks $g(x)$ unknown

- Obtaining a surrogate model, costly + no guarantee of success
- Query-based methods often require a lot of queries (10k-100k),

Query-based gradient estimation

- Score-based: we can query the continuous model scores $g(x) \in \mathbb{R}$. $\nabla_x g(x) \approx \sum_{i=1}^d \frac{g(x+\alpha e_i)-g(x)}{\alpha}e_i$
- Decision-based: we can query only the predicted class $f(x) \in \{-1,1\}$, similar techniques can be adapted for the decision-based case.

Transfer Attacks

- Train a similar surrogate model $\hat{f} \approx f$ on similar data
- Model stealing (query f given some unlabeled inputs $\{x_n, f(x_n)\}_{n=1}^N$) can facilitate transfer attacks.

Adversarial training

- Adversarial training: the goal is to minimize the adversarial risk. $\min_\theta R_\varepsilon(f_\theta) = \mathbb{E}_{\mathcal{D}}\left[\max_{\hat{x},\|\hat{x}-X\|\leq\varepsilon} 1_{f(\hat{x})\neq Y}\right]$
- $\mathcal{D}$ unknown → approximate it with a sample average + classification loss is non-continuous → use a smooth loss ⇒ $\min_\theta \frac{1}{N}\sum_{n=1}^N \max_{\hat{x}_n,\|x_n-\hat{x}_n\|\leq\varepsilon}\ell(y_n g_\theta(\hat{x}_n))$

1) $\forall x_n$, $\hat{x}_n^\star \approx \arg\max_{\|x_n-\hat{x}_n\|\leq\varepsilon}\ell(y_n g_\theta(\hat{x}_n))$

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

Advantages

- state-of-the-art approach for robust classification
- more interpretable gradients
- fully compatible with SGD

Disadvantages

- Increased comp time: prop to the number of PGD steps
- Robustness-accuracy tradeoff: too large ε = worse accuracy

18 Fairness criteria in classification

Use an algorithm to produce a score function $R = r(X)$ (given)

Sensitive attributes

- No fairness through unawareness: removing/ignoring sensitive attributes is not solving the problem
- Many features slightly correlated with the sensitive attribute can be used to recover the attribute

Three fundamental fairness criteria

Independence: $A \perp R$, Separation: $A \perp R \mid Y$, Sufficiency: $A \perp Y \mid R$ any of these three criteria are mutually exclusive

Independence: equal acceptance rate

$\mathbb{P}(D=1 \mid A=a) = \mathbb{P}(D=1 \mid A=b)$ not unfair practice.

Separation: equal error rate

$\mathbb{P}(D=1 \mid Y=0, A=a) = \mathbb{P}(D=1 \mid Y=0, A=b)$ (equal FP)

$\mathbb{P}(D=0 \mid Y=1, A=a) = \mathbb{P}(D=0 \mid Y=1, A=b)$ (equal FN)

Sufficiency:

$\mathbb{P}(Y=1 \mid R=r, A=a) = \mathbb{P}(Y=1 \mid R=r, A=b)$

Calibration and sufficiency

- A score R is calibrated if $\mathbb{P}(Y=1 \mid R=r) = r$
- Calibration by group: $\mathbb{P}(Y=1 \mid R=r, A=a) = r$, sufficiency

How to achieve fairness criteria

- Post-processing: adjust your learned classifier so that it becomes uncorrelated with the sensitive attribute A
- At training time: add regularization
- Pre-processing: adjust your features so that they become uncorrelated with the sensitive attribute A

19 Clustering

- Clusters are groups of points whose inter-point distances are small compared to the distances outside the cluster.
- Find "prototype" points $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$ and cluster assignments $z_n \in \{1,2,\ldots,K\}$ for all $n = 1,2,\ldots,N$ data vectors $\mathbf{x}_n \in \mathbb{R}^D$.

K-means clustering

Assume K is known. $\min_{\mathbf{z},\boldsymbol{\mu}} \mathcal{L}(\mathbf{z},\boldsymbol{\mu}) = \sum_{n=1}^N\sum_{k=1}^K z_{nk}\|\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2$ s.t. $\boldsymbol{\mu}_k \in \mathbb{R}^D$, $z_{nk} \in \{0,1\}$, $\sum_{k=1}^K z_{nk} = 1$, where $\mathbf{z}_n = [z_{n1}, z_{n2}, \ldots, z_{nK}]^\top$, $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N]^\top$, $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K]^\top$

K-means Algorithm

Initialize $\boldsymbol{\mu}_k\forall k$, then iterate: 1. For all n, compute $\mathbf{z}_n$ given $\boldsymbol{\mu}$.

$z_{nk} = \begin{cases} 1 \text{ if } k = \arg\min_{j=1,2,\ldots,K}\|\mathbf{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 \text{ otherwise} \end{cases} \to O(NKD)$

2. For all k, compute $\boldsymbol{\mu}_k$ given $\mathbf{z}$.Take derivative w.r.t. $\boldsymbol{\mu}_k$ to get: $\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N z_{nk}\mathbf{x}_n}{\sum_{n=1}^N z_{nk}} \to O(NKD)$

- Each step ↓ cost ⇒ Convergence to local optimum

Coordinate descent

$\mathbf{z}^{(t+1)} := \arg\min_{\mathbf{z}}\mathcal{L}\left(\mathbf{z}, \boldsymbol{\mu}^{(t)}\right)$, $\boldsymbol{\mu}^{(t+1)} := \arg\min_{\boldsymbol{\mu}}\mathcal{L}\left(\mathbf{z}^{(t+1)}, \boldsymbol{\mu}\right)$

Probabilistic model for K-means

$\log\prod_{n=1}^N p(x_n \mid \mu, z) = \log\prod_{n=1}^N \mathcal{N}(x_n \mid \mu_{k^*}, I_0) = \log\prod_{n=1}^N\prod_{k=1}^K \mathcal{N}(x_n \mid \mu_k, I_0)^{z_{nk}} = \log\prod_{n=1}^N\prod_{k=1}^K c\, e^{-\frac{1}{2}\|x_n-\mu_k\|^2\cdot z_{nk}} = -\sum_{n=1}^N\sum_{k=1}^K\frac{1}{2}\|x_n-\mu_k\|^2 z_{nk} + c' = -\mathcal{L}(\mu, z)$

K-means as a Matrix Factorization

$\min_{\mathbf{z},\boldsymbol{\mu}}\mathcal{L}(\mathbf{z},\boldsymbol{\mu}) = \sum_{n=1}^N\sum_{k=1}^K z_{nk}\|\mathbf{x}_n - \boldsymbol{\mu}_k\|_2^2 = \|\mathbf{X}^\top - \mathbf{M}\mathbf{Z}^\top\|_{\text{Frob}}^2$ s.t. $\boldsymbol{\mu}_k \in \mathbb{R}^D$, $z_{nk} \in \{0,1\}$, $\sum_{k=1}^K z_{nk} = 1$

Issues with K-means

1. Clusters are forced to be spherical.
2. "hard" cluster assignments.
3. Computation heavy for large N, D and K.

20 Gaussian Mixture Models

(1) resolved by using full covariance matrices $\boldsymbol{\Sigma}_k$ instead of isotropic covariances.

$p(\mathbf{X} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{z}) = \prod_{n=1}^N\prod_{k=1}^K[\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]^{z_{nk}}$ $\mu \in \mathbb{R}^{D\cdot K}$ $\Sigma \in \mathbb{R}^{D^2\cdot K}$ $\mu \in \mathbb{R}^K$

Soft-clustering

(2) resolved by defining z_n to be a random variable, $z_n \in \{1,2,\ldots,K\}$ that follows a multinomial distribution.

$z_{nk} = \begin{cases} 1 \text{ if assigned to k} \\ 0 \end{cases}$, $p(z_n = k) = \pi_k$ where $\pi_k > 0, \forall k$

and $\sum_{k=1}^K \pi_k = 1$

GMM definition

$p(\mathbf{X}, \mathbf{z} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \prod_{n=1}^N p(\mathbf{x}_n \mid z_n, \boldsymbol{\mu}, \boldsymbol{\Sigma})p(z_n \mid \boldsymbol{\pi}) = \prod_{n=1}^N\prod_{k=1}^K[\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]^{z_{nk}}\prod_{k=1}^K[\pi_k]^{z_{nk}}$

$\mathbf{x}_n$:= observed data vectors, z_n := latent unobserved variables, unknown parameters $\boldsymbol{\theta} := \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K, \boldsymbol{\pi}\}$

Marginalize z_n out

$p(\mathbf{x}_n \mid \boldsymbol{\theta}) = \sum_{k=1}^K \pi_k\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

- Deriving cost functions this way is good for statistical efficiency. Without a latent variable model, the number of parameters grows at rate $\mathcal{O}(N)$. After marginalization, the growth is reduced to $\mathcal{O}(D^2K)$ (assuming $D, K \ll N$).

Maximum likelihood

$\max_{\boldsymbol{\theta}}\sum_{n=1}^N\log\sum_{k=1}^K\pi_k\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

- Non-convex cost. Non-unique optima (permutations/renaming of the clusters). Unbounded: $\Sigma_k = \sigma_k \mathbf{I}$, $\sigma_k \in \mathbb{R}$

21 Expectation-Maximization Algorithm

Start with $\boldsymbol{\theta}^{(1)}$ and iterate: 1. Expectation step: Compute a lower bound to the cost such that it is tight at the previous $\boldsymbol{\theta}^{(t)}$:

$\mathcal{L}(\boldsymbol{\theta}) \geq \underline{\mathcal{L}}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}\right)$ and $\mathcal{L}\left(\boldsymbol{\theta}^{(t)}\right) = \underline{\mathcal{L}}\left(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^{(t)}\right)$.

2. Maximization step: Update $\boldsymbol{\theta}$: $\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}}\underline{\mathcal{L}}\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}\right)$.

Jensen's Inequality

- Log is convex → Concavity of log
- Given non-negative weights q s.t. $\sum_k q_k = 1$, the following holds for any $r_k > 0$: $\log\left(\sum_{k=1}^K q_k r_k\right) \geq \sum_{k=1}^K q_k\log r_k$

The expectation step

$\log\sum_{k=1}^K\pi_k\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \geq \sum_{k=1}^K q_{kn}\log\frac{\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{q_{kn}}$ with equality when, $q_{kn} = \frac{\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{k=1}^K\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}$

The maximization step

- Maximize the lower bound w.r.t. $\boldsymbol{\theta}$.

$\max_{\boldsymbol{\theta}}\sum_{n=1}^N\sum_{k=1}^K q_{kn}^{(t)}[\log\pi_k + \log\mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$

- Differentiating w.r.t. $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k^{-1}$:

$\boldsymbol{\mu}_k^{(t+1)} := \frac{\sum_n q_{kn}^{(t)}\mathbf{x}_n}{\sum_n q_{kn}^{(t)}}$, $\boldsymbol{\Sigma}_k^{(t+1)} := \frac{\sum_n q_{kn}^{(t)}(\mathbf{x}_n-\boldsymbol{\mu}_k^{(t+1)})(\mathbf{x}_n-\boldsymbol{\mu}_k^{(t+1)})^\top}{\sum_n q_{kn}^{(t)}}$

For π_k, we use the fact that they sum to 1. Therefore, we add a Lagrangian term, differentiate w.r.t. π_k and set to 0 , to get the following update:

$\pi_k^{(t+1)} := \frac{1}{N}\sum_{n=1}^N q_{kn}^{(t)}$

EM for GMM

Initialize $\boldsymbol{\mu}^{(1)}, \boldsymbol{\Sigma}^{(1)}, \boldsymbol{\pi}^{(1)}$ and iterate between the E and M step, until $\mathcal{L}(\boldsymbol{\theta})$ stabilizes.

1. E-step: Compute assignments $q_{kn}^{(t)} := \frac{\pi_k^{(t)}\mathcal{N}\left(\mathbf{x}_n|\boldsymbol{\mu}_k^{(t)},\boldsymbol{\Sigma}_k^{(t)}\right)}{\sum_{k=1}^K\pi_k^{(t)}\mathcal{N}\left(\mathbf{x}_n|\boldsymbol{\mu}_k^{(t)},\boldsymbol{\Sigma}_k^{(t)}\right)}$

2. Compute the marginal likelihood (cost).

$\mathcal{L}\left(\boldsymbol{\theta}^{(t)}\right) = \sum_{n=1}^N\log\sum_{k=1}^K\pi_k^{(t)}\mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k^{(t)}, \boldsymbol{\Sigma}_k^{(t)}\right)$

3. M-step: $\boldsymbol{\mu}_k^{(t+1)} := \frac{\sum_n q_{kn}^{(t)}\mathbf{x}_n}{\sum_n q_{kn}^{(t)}}$, $\pi_k^{(t+1)} := \frac{1}{N}\sum_n q_{kn}^{(t)}$, $\boldsymbol{\Sigma}_k^{(t+1)} := \frac{\sum_n q_{kn}^{(t)}\left(\mathbf{x}_n-\boldsymbol{\mu}_k^{(t+1)}\right)\left(\mathbf{x}_n-\boldsymbol{\mu}_k^{(t+1)}\right)^\top}{\sum_n q_{kn}^{(t)}}$,

- If we let the covariance be diagonal i.e. $\boldsymbol{\Sigma}_k := \sigma^2\mathbf{I}$, then EM algorithm is same as K-means as $\sigma^2 \to 0$.

Posterior distribution

$\underbrace{p(\mathbf{x}_n, z_n \mid \boldsymbol{\theta})}_{\text{joint}} = \underbrace{p(\mathbf{x}_n \mid z_n, \boldsymbol{\theta})}_{\text{likelihood}}\underbrace{p(z_n \mid \boldsymbol{\theta})}_{\text{prior}} = \underbrace{p(z_n \mid \mathbf{x}_n, \boldsymbol{\theta})}_{\text{posterior}}\underbrace{p(\mathbf{x}_n \mid \boldsymbol{\theta})}_{\text{marginal}}$

$\underbrace{p(A,B)}_{\text{joint}}$ $\underbrace{p(A|B)}_{\text{likelihood}}$ $\underbrace{p(B)}_{\text{prior}}$ $\underbrace{p(B|A)}_{\text{posterior}}$ $\underbrace{p(A)}_{\text{marginal}}$

EM in general

- Given joint distribution $p(\mathbf{x}_n, z_n \mid \boldsymbol{\theta})$, the marginal likelihood can be lower bounded similarly. Compact EM: $\boldsymbol{\theta}^{(t+1)} := \arg\max_{\boldsymbol{\theta}}\sum_{n=1}^N\mathbb{E}_{p(z_n|\mathbf{x}_n,\boldsymbol{\theta}^{(t)})}[\log p(\mathbf{x}_n, z_n \mid \boldsymbol{\theta})]$
- Another interpretation: part of the data is missing, i.e. $(\mathbf{x}_n, z_n)$ is "complete" data and z_n is missing. Averages over the "unobserved" part of the data.

22 Matrix Factorization

Given items (movies) $d = 1,2,\ldots,D$ and users $n = 1,2,\ldots,N$, X is $D \times N$ (sparse)

Algorithm

$X \approx WZ^T$, $W \in \mathbb{R}^{D\times K}$, $Z \in \mathbb{R}^{N\times K}$ tall matrices $K << N, D$

$\min_{W,Z}\ \mathcal{L}(W,Z) := \frac{1}{2}\sum_{(d,n)\in\Omega}[x_{dn} - (WZ^T)_{dn}]^2$

- This cost is not jointly convex w.r.t. W and Z, nor identifiable as $(w^*, z^*) \Leftrightarrow (\beta w^*, \beta^{-1}z^*)$

Choosing K

- $\uparrow K \Rightarrow$ overfitting ($\Leftrightarrow \downarrow K \Rightarrow$ underfitting). For $K >> N, D \Rightarrow$ $(W^*, Z^{*^T}) = (X, I) = (I, X)$

Regularization

$\frac{1}{2}\sum_{(d,n)\in\Omega}[x_{dn} - (WZ^T)_{dn}]^2 + \frac{\lambda_w}{2}||W||_{\text{Frob}}^2 + \frac{\lambda_z}{2}||Z||_{\text{Frob}}^2$ $\lambda_w, \lambda_z \in \mathbb{R} > 0$

Stochastic Gradient Descent ($\mathcal{O}(K)$)

$\mathcal{L} = \frac{1}{|\Omega|}\sum_{(d,n)\in\Omega}\frac{1}{2}[x_{dn} - (\mathbf{W}\mathbf{Z}^\top)_{dn}]^2 = \frac{1}{|\Omega|}\sum_{(d,n)\in\Omega}f_{d,n}$

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

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

Alternating Least Squares

$\frac{1}{2}\sum_{d=1}^D\sum_{n=1}^N\left[x_{dn} - (W\mathbf{Z}^\top)_{dn}\right]^2 = \frac{1}{2}\|\mathbf{X} - \mathbf{W}\mathbf{Z}^\top\|_{Frob}^2$

$Z^\top := (\mathsf{W}^\top\mathsf{W} + \lambda_z\mathsf{I}_K)^{-1}\mathsf{W}^\top\mathsf{X}$, $\mathbf{W}^\top := (\mathbf{Z}^\top\mathbf{Z} + \lambda_w\mathbf{I}_K)^{-1}\mathbf{Z}^\top\mathbf{X}^\top$

- Cost: need to invert a $K \times K$ matrix

23 Text Representation

-For each word, find mapping (embedding) $w_i \mapsto \mathbf{w}_i \in \mathbb{R}^K$

Co-Occurence Matrix: sparse

-big corpus un-labeled, co-occurrence counts. n_{ij} := #contexts where word w_i occurs together with word w_j.

Learning Word-Representations: Matrix Factorisation

- transformation: $x_{dn} := \log(n_{dn})$. $\mathbf{W}, \mathbf{Z}$ s.t. $\mathbf{X} \approx \mathbf{W}\mathbf{Z}^\top$

$\min_{\mathbf{W},\mathbf{Z}}\mathcal{L}(\mathbf{W},\mathbf{Z}) := \frac{1}{2}\sum_{(d,n)\in\Omega}f_{dn}\left[x_{dn} - \left(\mathbf{W}\mathbf{Z}^\top\right)dn\right]^2$

where $\mathbf{W} \in \mathbb{R}^{D\times K}$, $\mathbf{Z} \in \mathbb{R}^{N\times K}$, $K \ll D, N$, $\Omega \subseteq [D]\times[N]$ indices of non-zeros of the count matrix $\mathbf{X}$, f_{dn} are weights to each entry.

GloVe: word2vec variant

$f_{dn} := \min\{1, (n_{dn}/n_{\max})^\alpha\}$, $\alpha \in [0;1]$ (e.g. $\alpha = \frac{3}{4}$)

Training: SGD or Alternating Least-Squares (ALS)

Skip-Gram Model

- Binary classification: real or fake word pairs (w_d, w_n).
- Given w_d, a context word w_n is:

real = appearing together in a context window of size 5

fake = any word $w_{n'}$ sampled randomly: Negative sampling (also: Noise Contrastive Estimation)

Fast Text: Matrix factor. for sentence representations

-sentence $s_n = (w_1, w_2, \ldots, w_m)$, let $\mathbf{x}_n \in \mathbb{R}^{|\mathcal{V}|}$ bag-of-words representation of the sentence.

$\min_{\mathbf{W},\mathbf{Z}}\mathcal{L}(\mathbf{W},\mathbf{Z}) := \sum_{s_n \text{ a sentence}} f\left(y_n\mathbf{W}\mathbf{Z}^\top\mathbf{x}_n\right)$, where $\mathbf{W} \in \mathbb{R}^{1\times K}$, $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}|\times K}$, $\mathbf{x}_n \in \mathbb{R}^{|\mathcal{V}|}$ represents our n-th training sentence.

- f is a linear classifier loss function, $y_n \in \{\pm 1\}$ is the classification label for sentence $\mathbf{x}_n$.

24 Self-supervised Learning

BERT: (Bidirectional) Encoder Repr. from Transf.

- encoder-only trained on masked language modelling.
- Encoder: whole sequence at once, every token can generally attend to every other token (both previous and later ones, hence bidirectional). Generates a fixed sized output, typically one token per input.

BERT Training Objective:

Predict original token. Softmax cross-entropy loss over all tks. [CLS] token: order (binary classification).

GPT: Generative Pre-trained Transformers

- for Next Token Prediction: decoder-only transformer architecture
- Training Procedure: teacher forcing,
- In-Context Learning: prompting ChatGPT (prompt engineering)

Joint Embedding Methods (Images): enc. invariant transfo.

- BYOL: use two encoders f_1, f_2, f_2 is exp moving average of f_1
- VicReg: force non-0 var. to avoid collapse but min the cov.

Contrastive learning

- Given a positive pair $\mathbf{x}, \mathbf{x}^+$from the datapoint $\mathbf{x}$, and a negative view $\mathbf{x}^-$from a different datapoint $\mathbf{x}' \neq \mathbf{x}$: $\mathrm{s}\left(f(\mathbf{x}), f\left(\mathbf{x}^+\right)\right) > \mathrm{s}\left(f(\mathbf{x}), f\left(\mathbf{x}^-\right)\right)$, s func quantifies the similarity of two embeddings.

SimCLR: contrastive learning

1. clf-like obj, N neg samples $L(\mathbf{x}) = -\mathbb{E}\left[\log\frac{\exp\left(\mathrm{s}\left(f(\mathbf{x}), f(\mathbf{x}^+)\right)\right)}{\sum_{i=1}^N\exp\left(\mathrm{s}\left(f(\mathbf{x}), f(\mathbf{x}_i^-)\right)\right)}\right]$

max score between $\mathbf{x}$ and $\mathbf{x}^+$, min score between $\mathbf{x}$ and all $\mathbf{x}_i^-$.

2. use projector to map the encoder output to the similarity space: $f(\mathbf{x}) = \underbrace{f_2}_{\text{projector}} \circ \underbrace{f_1}_{\text{encoder}}(\mathbf{x})$ use only f_1 for downstream tasks.

3. cosine similarity : $\mathrm{S}(\mathbf{e}_1, \mathbf{e}_2) = \frac{\langle\mathbf{e}_1,\mathbf{e}_2\rangle}{\|\mathbf{e}_1\|_2\|\mathbf{e}_2\|_2}/\tau$

CLIP: captioned images to learn a joint multimodal embedding space

- Max cos sim. of caption and image embeds, min unrelated pairs.
- Few-shot learning: learn to clf new classes with only a few labelled examples. (CLIP able to zero shot)

25 Generative Models

- Discriminative model predict $y \to$ conditional distribution $p(y \mid x)$. Generative models $\to$ distribution $p(x)$ defined over the datapoints x. Generative models categories: explicit (explicitly model the probability distribution) or implicit (generate samples according to p).
- Implicit density models (GANs), high-quality, realistic samples, challenging to train (mode collapse model fails to capture the diversity of the data distribution)
- Tractable density models, (Autoregressive models and Normalizing Flows), exact likelihood computation, (beneficial for tasks that require likelihood evaluation and interpretability). Autoregressive models generate data sequentially, slow in sampling. Normalizing Flows, more efficient sampling and inference (using invertible transformations), computationally intensive to train.
- Approximate models, (VAEs and Energy-based models). VAEs optimize a lower bound on the likelihood $arrow$ easier and more stable training than GANs but less sharp samples.
- Diffusion models $arrow$ hybrid approach, iteratively convert noise into samples via a reverse Markov process, highquality images and audio, slow at sampling time.
- trade-off between sample quality, sampling speed, and diversity.

Generative Adversarial Networks GANs

- generator G, params:$\boldsymbol{\theta}$ and discriminator D params:$\boldsymbol{\varphi}$.

$\boldsymbol{\theta}^\star \in \arg\min_{\boldsymbol{\theta}\in\Theta}\mathcal{L}^{\boldsymbol{\theta}}(\boldsymbol{\theta}, \boldsymbol{\varphi}^\star)$ $\quad \boldsymbol{\varphi}^\star \in \arg\min_{\boldsymbol{\varphi}\in\Phi}\mathcal{L}^{\varphi}(\boldsymbol{\theta}^\star, \boldsymbol{\varphi})$

- $\mathcal{L}^{\boldsymbol{\theta}} = -\mathcal{L}^{\varphi}$ zero-sum game and is a minmax problem.

1. The discriminator "distinguishes" real vs. fake samples: p_z "noise" distr., e.g. $\mathcal{N}(0,1)$, p_d real data distr., $D : x \mapsto y \in [0,1]$, where y is an estimated probability that $x \sim p_d$
2. The generator aims at fooling the discriminator that its samples are real: $G : z \mapsto x$, such that if $z \sim p_z$, then hopefully $x \sim p_d$ p_g the "fake" data distribution
3. Objective $\min_G\max_D \mathop{\mathbb{E}}_{x\sim p_d}[\log D(x)] + \mathop{\mathbb{E}}_{z\sim p_z}[\log(1 - D(G(z)))]$

$\mathscr{L}_D(G,D) = \max_D \mathop{\mathbb{E}}_{x\sim p_d}[\log D(x)] + \mathop{\mathbb{E}}_{z\sim p_z}[\log(1 - D(G(z)))]$

$\mathscr{L}_G(G,D) = \min_G \mathop{\mathbb{E}}_{z\sim p_z}[\log(1 - D(G(z)))]$

4. Theoretical Solution: The optimum is reached when $p_g = p_d$ and the optimal value is $-\log 4$

Alternating-GAN algorithm

- G : deep neural network $G(\mathbf{z}; \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$
- D : deep neural network $D(\mathbf{x}; \boldsymbol{\varphi})$ with parameters φ
- G and D have different losses $\mathcal{L}^\theta$ and $\mathcal{L}^{\varphi a}$, resp.
- CGANs G, D are conditioned during training by some additional information (one-hot)

Algorithm 1 *alternati*

Input: dataset $\mathcal{D}$, known c
learning rate η, generator los
inator loss $\mathcal{L}^\varphi$
Initialize: $D(\cdot;\boldsymbol{\varphi})$, $G(\cdot;\boldsymbol{\theta})$
for $t = 1$ **to** T **do**
  $\mathbf{x} \sim p_x$ {sam
  $\mathbf{z} \sim p_z$ {sample
  $\boldsymbol{\varphi} = \boldsymbol{\varphi} - \eta\nabla_{\boldsymbol{\varphi}}\mathcal{L}^{\varphi}(\boldsymbol{\theta}, \boldsymbol{\varphi}, \mathbf{x}, \mathbf{z}$
  $\mathbf{z} \sim p_z$ {sample
  $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}}\mathcal{L}^{\theta}(\boldsymbol{\theta}, \boldsymbol{\varphi}, \mathbf{z})$
end for
Output: $\boldsymbol{\theta}, \boldsymbol{\varphi}$

Diffusion models

- Consider a Markov chain with $X_0 \sim p_0$ and a forward transition s.t. $X_{t+1} \sim p(\cdot \mid X_t)$ then we call **forward decomposition**: $p(x_{0:T}) = p_0(x_0)\prod_{t=0}^{T-1}p(x_{t+1} \mid x_t)$ where at each step, the marginal distribution of X_t satisfies: $p(x_t) = \int p(x_t \mid x_{t-1})p(x_{t-1})\mathrm{d}x_{t-1}$
- backward transition can be obtained with Bayes' rule $p(x_t \mid x_{t+1}) = p(x_{t+1} \mid x_t)p(x_t)/p(x_{t+1})$, and we call **backward decomposition**: $p(x_{0:T}) = p_T(x_T)\prod_{t=0}^{T-1}p(x_t \mid x_{t+1})$
- A generative model can be constructed sampling from $p(x_{0:T})$ using **ancestral sampling**:

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

- We consider $p_0 = p_{\text{data}} \in \mathcal{P}(\mathbb{R}^d)$
- We chose the forward transition to be Gaussian $p(x_{t+1} \mid x_t) = \mathcal{N}(x_{t+1}; \alpha x_t, (1 - ..\alpha^2)\mathbb{I}_d)$
- The conditional $p(x_t \mid x_0)$ is also Gaussian $\mathcal{N}(x_t; \alpha^t x_0, (1 - \alpha^{2t})\mathbb{I}_d)$
- The Markov chain converges to $p_{\text{ref}} = \mathcal{N}(x; 0, \mathbb{I}_d)$ for $Tarrow\infty$.
- Therefore for T large enough $p_T(x) \approx p_{\text{ref}}(x)$
- To generate samples we use ancestral sampling replacing p_T with p_{ref}

- Key problem: the backward transition $p(x_t \mid x_{t+1})$ can't be computed, we need an approximation!
- Using a Taylor expansion we can approximate the backward transition: $p(x_t \mid x_{t+1}) = p(x_{t+1} \mid x_t)\exp[\log p(x_t) - \log p(x_{t+1})] \approx \mathcal{N}(x_t; (2-\alpha)x_{t+1} + (1-\alpha^2)\underbrace{\nabla\log p(x_{t+1})}_{\text{Score}}, (1-\alpha^2)\mathbb{I}_d)$

Denoising score matching

- The Stein Score is analytically intractable, but we can approximate it using a NN $s_\theta(x_t)$.
- Surprisingly, we can learn the score solving a simple regression problem via the Denoising Score Matching (DSM) loss: $\theta^\star = \arg\min_\theta\mathcal{L}(\theta) := \frac{1}{2}\mathbb{E}_{X_t}[\|\nabla\log p(X_t) - s_\theta(X_t)\|^2]$
- Problem still intractable: we do not have access to the marginal $p(x_t)$ at each step but we can manipulate it:

$\mathcal{L}(\theta) = \mathrm{C}_1 + \frac{1}{2}\mathbb{E}_{X_t}[\|s_\theta(X_t)\|^2] - \mathbb{E}_{X_t}[\nabla\log p(X_t)^\top s_\theta(X_t)] = \mathrm{C}_1 + \frac{1}{2}\mathbb{E}_{X_t}[\|s_\theta(X_t)\|^2] - \int\nabla\log p(X_t)^\top s_\theta(X_t)p(X_t)\mathrm{d}x_t$

- Using that $p(x_t) = \int p_0(x_0)p(x_t \mid x_0)\mathrm{d}x_0$ we get: $\nabla\log p(X_t) = \mathbb{E}_{X_0|X_t}[\nabla\log p(X_t \mid X_0)]$

Substituding in the DSM-loss:

$\mathcal{L}(\theta) = \mathrm{C}_1 + \frac{1}{2}\mathbb{E}_{X_t}[\|s_\theta(X_t)\|^2] - \mathbb{E}_{X_0}\mathbb{E}_{X_t|X_0}[\nabla\log p(X_t \mid X_0)^\top s_\theta(X_t)] = \mathrm{C}_1 + \frac{1}{2}\mathbb{E}_{X_0}\mathbb{E}_{X_t|X_0}[\|s_\theta(X_t) - \nabla\log p(X_t \mid X_0)\|^2] - \underbrace{\frac{1}{2}\mathbb{E}_{X_0}\mathbb{E}_{X_t|X_0}[\|\nabla\log p(X_t \mid X_0)\|^2]}_{\text{const.}} = \mathrm{C}_2 + \frac{1}{2}\mathbb{E}_{X_0}\mathbb{E}_{X_t|X_0}[\|s_\theta(X_t) - \nabla\log p(X_t \mid X_0)\|^2]$

- In practice we estimate all the scores simultaneously i.e. $s_{\theta^\star}(t, X_t)$ such that: $\theta^* = \arg\min_\theta\frac{1}{2}\sum_{t=1}^T\mathbb{E}_{X_0}\mathbb{E}_{X_t|X_0}[\|s_\theta(t, X_t) - \nabla\log p(X_t \mid X_0)\|^2]$
- Recalling now our choice of forward transition for which $p(x_t \mid x_0) = \mathcal{N}(x_t; \alpha^t x_0, (1 - \ldots\alpha^{2t})\mathbb{I}_d)$:

- $\nabla\log p(x_t \mid x_0) = -\frac{(x_t - \alpha^t x_0)}{(1-\alpha^{2t})}$
- $X_t = \alpha^t X_0 + \sqrt{1-\alpha^{2t}}\xi_t\xi_t \sim \mathcal{N}(0, \mathbb{I}_d)$
- $\nabla\log p(X_t \mid X_0) = -\frac{\xi_t}{\sqrt{1-\alpha^{2t}}}$

- parameterize the score function as $s_\theta(t, X_t) = -\frac{\hat{\xi}_\theta(t,X_t)}{\sqrt{1-\alpha^{2t}}}$ then the problem is equivalent to: $\theta^\star = \arg\min_\theta\frac{1}{2}\sum_{k=1}^T\mathbb{E}_{X_0}\mathbb{E}_{X_t|X_0}[\|\hat{\xi}_\theta(t, X_t) - \xi_t\|^2]$

which has the illuminating interpretation of learning to predict the added noise ξ_t from the noised data X_t.

- Finally, we can generate new samples from the backward process by sampling $X_T \sim p_{\text{ref}}$ and using the approximation of the backward transition and the learned score to perform ancestral sampling:

$X_t = (2-\alpha)X_{t+1} + (1-\alpha^2)s_{\theta^\star}(t+1, X_{t+1}) + \sqrt{1-\alpha^2}\xi_{t+1}$ $\quad \xi_{t+1} \sim \mathcal{N}(0, \mathbb{I}_d)$

Training and Sampling

Algorithm 2 Training

Input: Data distribution p_0, α
Output: Learned parameters θ^*
Initialize: Parameters θ
repeat
  Sample initial data $X_0 \sim p_0$
  Sample time step $t \sim \mathrm{Uniform}(\{1,\ldots,T\})$
  Sample noise $\xi_t \sim \mathcal{N}(0, \mathbb{I}_d)$
  Update θ using gradient descent step on:
  $\nabla_\theta\frac{1}{2}\mathbb{E}_{X_0}\mathbb{E}_{X_t|X_0}\left[\left\|s_\theta(t, \alpha^t X_0 + \sqrt{1-\alpha^{2t}}\xi_t) - \xi_t\right\|^2\right]$
until convergence

Algorithm 3 Sampling

Input: Learned parameters θ^*, p_{ref}, α
Output: Generated sample X_0
Sample $X_T \sim p_{\text{ref}}$
for $t = T-1$ **to** 0 **do**
  Sample noise $\xi_{t+1} \sim \mathcal{N}(0, \mathbb{I}_d)$:
  $X_t = (2-\alpha)X_{t+1} + (1-\alpha^2)s_{\theta^*}(t+1, X_{t+1}) + \sqrt{1-\alpha^2}\xi_{t+1}$
end for
Return x_0

Conditional training

$\theta^\star = \arg\min_\theta\frac{1}{2}\sum_{k=1}^T\mathbb{E}_{X_0,Y}\mathbb{E}_{X_t|X_0}[\|s_\theta(t, X_t^Y; Y) - \nabla\log p(X_t^Y \mid X_0^Y)\|^2]$