IML Summary 2023 - Jorit Geurts, jgeurts@ethz.ch

1 Regression

Dataset: $\mathcal{D} = \{(\vec{x}_i, y_i)\}_{i=1}^N$, $\vec{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$
$\tilde{X} = [\vec{x}_1, \dots, \vec{x}_N]^\top \in \mathbb{R}^{N\times d}$, $\vec{y} = [y_1, \dots, y_N]^T \in \mathbb{R}^N$
Training Error (TE): $\mathcal{L}(f_D) = \frac{1}{N}\sum_{i=1}^N \ell(f_D(x_i), y_i)$

1.1 Least Squares: $\ell(f(x_i), y_i) = (y_i - f(x_i))^2$

Objective: $\hat{w} = \arg\min_w \frac{1}{N}\sum_{i=1}^N (y_i - w^T\vec{x}_i)^2$
$\hat{w} = \arg\min_w \frac{1}{n}\|y - Xw\|_2^2, X = [\vec{1}, \tilde{X}]^\top, w = [w_0, w]^\top$
CF Solution: $n > d \rightarrow \hat{w} = (X^TX)^{-1}X^Ty$
$n < d \rightarrow \hat{w} = X^T(XX^T)^{-1}y$ (many exact sol.)
non-linear fnct.: X to $\phi(X)$ ($\phi(X) = [1, X, X^2]$)

1.1.1 Ridge Reg.: $\ell = (y_i - f(x_i))^2 + \lambda\|w\|^2$

Objective: $\hat{w} = \arg\min_w \frac{1}{N}\sum_{i=1}^N (y_i - w^T\vec{x}_i)^2 + \lambda\|w\|^2$
CF Solution: $\hat{w} = (X^TX + \lambda I)^{-1}X^Ty$

1.1.2 LASSO Reg.: $\ell = (y_i - f(x_i))^2 + \lambda\|w\|_1$

Objective: $\hat{w} = \arg\min_w \frac{1}{N}\sum_{i=1}^N (y_i - w^T\vec{x}_i)^2 + \lambda\|w\|_1$

LASSO vs Ridge

LASSO limits model complexity (coefficients = 0, sparse matrix), Ridge limits value of coefficients

2 Optimization

2.0.1 Gradient Descent

1. Initialization of random $w_0 \in \mathbb{R}^N$
2. Update: $w_{t+1} = w_t - \eta_t \nabla\mathcal{L}(w_t)$

Stop: $\|\nabla\mathcal{L}(w_t)\| < \epsilon$ or $t > t_{max}$

2.0.2 (Batch) Stochastic Gradient Descent

1. Initialization of random $w_0 \in \mathbb{R}^N$
2. Update: $w_{t+1} = w_t - \eta_t \nabla\mathcal{L}_S(w_t)$

$\mathcal{L}(w_t) = \frac{1}{|S|}\sum_{i\in S}\ell(f_w(x_i), y_i)$, (with replacement)
Stop: $\|\nabla\mathcal{L}(w_t)\| < \epsilon$ or $t > t_{max}$
Memory ↓, p(escape saddle p.)↑, faster than GD

GD for Linear Regression

Gradient: $\nabla\mathcal{L}(w) = (2)X^T(Xw - y)$
$\rightarrow \|w^{t+1} - \hat{w}\| \leq \|I - \eta X^\top X\|_{op}\|w^t - \hat{w}\|$
$\leq \|I - \eta X^\top X\|_{op}^{t+1}\|w^0 - \hat{w}\|$, $\rho = \|I - \eta X^\top X\|_{op}$
$\rho < 1 \rightarrow$ convergence, $\eta \leq 2/\lambda_{max}(X^\top X)$ **sufficient**
Well conditioned: $\lambda_{max}(X^\top X) \approx \lambda_{min}(X^\top X)$
Iterations until within ϵ: $t \geq \frac{\log(c/\epsilon)}{\log(1/\rho)}$
Step size: $\eta_{opt} = 2/(\lambda_{max}(X^\top X) + \lambda_{min}(X^\top X))$
$K = \frac{\lambda_{max}(X^\top X)}{\lambda_{min}(X^\top X)}$ condition number $\rightarrow \rho_{min} = \frac{K-1}{K+1}$
GD less computationally costly than CF for large $N \vee d$

3 Model Selection & Validation

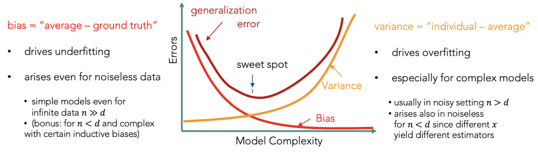
Estimation Error: (EE) tells goodness of $\hat{f}_D$
$\mathbb{E}_X[\ell(f^\star(X), \hat{f}(X))] = \mathbb{E}_{X,Y}[\ell(\hat{f}_D(X), Y)] + K$
Generalization Error: (GE) $\mathbb{E}_{X,Y}[\ell(\hat{f}_D(X), Y)]$
Test Error: (Estim. GE, unbiased) $\frac{1}{|\mathcal{D}_V|}\sum\ell(f(x), y)$
Generally TE < EE, but EE not available
Split data: $\mathcal{D} = \mathcal{D}_T \cup \mathcal{D}_V$, Train on $\mathcal{D}_T$, test on $\mathcal{D}_V$
Training loss not always smaller than test loss!!!

3.1 Cross Validation

Find optimal model → split into trainingset D_{use}, validationset $D_k \subset D_{use}$ and testset D_{test} $K = |D_{use}|$: Leave-One-Out CV (LOOCV)
Large K: CV bad estimate of generalization error and comp. expensive, but f_k closer to full model

3.2 Bias-Variance Tradeoff

Squared Model Bias: avg. distance to truth
Model Variance: avg. distance to avg. model $\bar{f}$
Approximated by: $\text{Var}_\mathcal{D} \approx \frac{1}{J}\sum_J(\hat{f}_j(x) - \bar{f}(x))$



Expected Generalization Error:
$\mathbb{E}_\mathcal{D}[(\hat{f}_\mathcal{D}(X) - Y)^2] = \text{Var}_\mathcal{D}(\hat{f}_\mathcal{D}) + \text{Bias}_\mathcal{D}^2(\hat{f}_\mathcal{D}) + \sigma^2$
$\lambda \rightarrow 0$: Var → ∞, Bias → 0 $\lambda \rightarrow \infty$: Var → 0, Bias → ∞
CV to find optimal size of λ

4 Classification

4.1 Binary Classification

Label: $y \in \{-1, 1\}$ with $\hat{y} = \text{sign}(\hat{f}(x))$
Loss functions: 0-1, Hinge, Logistic, Exponential
Logistic is more robust to outliers. ⇒ best loss
other losses upper bound of 0-1 loss.

4.2 Logistic Regression (ℓ_{log} with GD)

Aligns direction that maximizes the min ℓ_2-distance.
$w_{MM} = \arg\max_{\|w\|_2=1}\min_i y_i\langle w, x_i\rangle$

4.3 Support Vector Machines

Hard Margin SVM (data linearly separable)

Solves the max margin optimization problem:
$w_{MM} \parallel \arg\min_{\tilde{w}}\|w\|_2$ s.t. $y_i\tilde{w}^\top x_i \geq 1\, \forall i$

Soft Margin SVM (data not separable)

Solves the max margin problem with slack variables:
$w_{MM} \parallel \arg\min_{\tilde{w}} 0.5\|w\|_2^2 + \lambda\sum_{i=1}^n \xi_i$
s.t. $y_i\tilde{w}^\top x_i \geq 1 - \xi_i$, $\xi_i \geq 0$ $\forall i = 1, \dots, n$
For given w unconstrained SM SVM: (hinge loss)
$\min_w \frac{1}{2}\|w\|_2^2 + \lambda\sum_{i=1}^n \max(0, 1 - y_iw^\top x_i)$
margin $\gamma = \frac{1}{\|w\|_2}$

4.4 Multiclass Classification

Label: $y \in \{1, \dots, K\}$. Often multiple $\hat{f}_i(x)$ needed!
Prediction: $\hat{y} = \arg\max_{k=1,\dots,K}\hat{f}_k(x)$
One-vs-All:

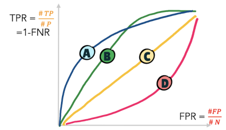
1. Train K binary classifiers $\hat{f}_k(x)$
2. Apply softmax: $\hat{\sigma}_k(x) = \frac{\exp(\hat{f}_k(x))}{\sum_{k'=1}^K \exp(\hat{f}_{k'}(x))}$
3. Predict: $\hat{y} = \arg\max_{k=1,\dots,K}\hat{\sigma}_k(x)$

Decision boundary: $\hat{f}_k(x) = \hat{f}_{k'}(x)$

4.5 Asymmetric Losses

Misclassification of one class worse. Choose as negative more important class → **Null Hypothesis**

Prediction \ Truth	$y = -1$	$y = +1$	
$\hat{y} = -1$	TN	FN	type II error
$\hat{y} = +1$	FP	TP	
	type I error		



Minimize over weighted loss by c_{FP} and c_{FN}:
$\min_{\hat{y}} \frac{c_{FN}}{\#y=1}\sum_{y=1}\mathbb{I}_{(y_i=-1)} + \frac{c_{FP}}{\#y=-1}\sum_{y=-1}\mathbb{I}_{(y_i=1)}$
Equal to: $\min_{\hat{y}} c_{FN}\frac{\#FN}{\#P} + c_{FP}\frac{\#FP}{\#N}$, $\uparrow c_{FP} \rightarrow \uparrow \tau$
mathematically written as: $\hat{y} = \begin{cases} +1 & \text{if } \hat{f}(x) > \tau \\ -1 & \text{if } \hat{f}(x) < \tau \end{cases}$

4.5.1 ROC: varying τ, fixed $\hat{f} \rightarrow$ TPR vs. FPR

$\tau \downarrow \rightarrow$ TPR ↑, FPR ↑, $\tau \uparrow \rightarrow$ TPR ↓, FPR ↓

How many positives are (correctly) labeled positive — False Discovery Rate — How many negatives are (falsely) labeled positive

Precision	$\frac{\#TP}{\#[\hat{y} = +1]}$	$\sim P_n(y = 1 \mid \hat{y} = 1)$	FDR (= 1 - precision)	$\frac{\#FP}{\#[\hat{y} = +1]}$	$\sim P_n(y = -1 \mid \hat{y} = 1)$
Recall (= TPR, ~ Power = 1- FNR)	$\frac{\#TP}{\#[y = +1]}$	$\sim P_n(\hat{y} = 1 \mid y = 1)$	FPR (~ Type I error)	$\frac{\#FP}{\#[y = -1]}$	$\sim P_n(\hat{y} = 1 \mid y = -1)$

Want large! — Want small!

F1 score: $F_1 = \frac{2}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = \frac{2\cdot\text{TP}}{2\text{TP}+\text{FP}+\text{FN}}$

5 Kernel Methods $\langle x, z\rangle = x^\top z$

Non-linear feature map → computational infeasiblity.
Featurize f via $\alpha \in \mathbb{R}^n$ via $f(x) = \alpha^\top\Phi\phi(x)$
$\Phi = \begin{bmatrix}\phi^\top(x_1) \\ \phi^\top(x_n)\end{bmatrix} \rightarrow f(x) = \sum_{i=1}^n \alpha_i\langle\phi(x_i)^\top\phi(x)\rangle$

1. Solve $\hat{\alpha} = \arg\min_{\alpha\in\mathbb{R}^n}\frac{1}{n}\sum_{i=1}^n\ell(y_i, \alpha^\top\Phi\phi(x_i))$
2. Ouput $\hat{f}(x) = \hat{\alpha}^\top\Phi\phi(x)$ - funtion of $\langle\phi(x), \phi(z)\rangle$
3. Replace $\langle\phi(x), \phi(z)\rangle$: $k(x, z) = \phi(x)^\top\phi(z)$

$K = \Phi\Phi^\top = \begin{bmatrix}k(x_1, x_1) & k(x_1, x_n) \\ k(x_n, x_1) & k(x_n, x_n)\end{bmatrix} \in \mathbb{R}^{n\times n}$

Trick: $k(x, z) = f(x, z)$ without needing $\phi(x)$, $\phi(z)$

5.1 Properties & Valid Kernels & Operations

symmetric: $k(x, z) = k(z, x)$, psd: $k(x, z) \succcurlyeq 0$

Inner product: $k(x, z) = h(\langle x, z\rangle)$ psd iff Taylor:
$h(\langle x, z\rangle) = \sum_{j=0}^\infty a_j(\langle x, z\rangle)^j$ has $a_j > 0$
Polynomial: $k(x, z) = (\langle x, z\rangle + c)^m$ psd for $m \in \mathbb{N}^+$
RBF: $k(x, z) = \exp\left(-\frac{\|x-z\|^\alpha}{\tau}\right)$ is psd for $\tau > 0$
τ is bandwidth, α is shape parameter
Gaussian: $\alpha = 2$, **Laplace:** $\alpha = 1$
$k(x, y) = \frac{1}{1-xy} = \sum_{i=0}^\infty x^iy^i$, $x, y \in (-1, 1)$
$k(x, y) = 2^{xy}$, $k(x, y) = \cos(x - y)$
$k(x, y) = \min(x, y)$, $k(x, z) = x^\top Mz$, $M \succcurlyeq 0, M^\top = M^\top$

Sum: $k(x, z) = k_1(x, z) + k_2(x, z)$
Prod.: $k(x, z) = k_1(x, z)\cdot k_2(x, z) = h(x)k(x, y)h(y)$
Map: $k(x, z) = \langle\phi(\nu(x)), \phi(\nu(z))\rangle = k_\phi = \langle\nu(x), \nu(z)\rangle$

5.2 Kernel Linear Regression

$\hat{w} = \arg\min_{w\in\mathbb{R}^d}\|y - \Phi w\|_2^2 = \Phi^\top\hat{\alpha}$
$\hat{\alpha} = \arg\min_{\alpha\in\mathbb{R}^n}\|y - \Phi\Phi^\top\alpha\|_2^2 = \arg\min_\alpha\|y - K\alpha\|_2^2$
$\hat{f}(x) = \hat{w}^\top\phi(x) = \hat{\alpha}^\top\Phi\phi(x) = y^\top K^{-1}\Phi\phi(x)$
Solve Problem: $\hat{\alpha} = K^{-1}y$

5.2.1 Kernel Ridge Regression

$\hat{\alpha} = \arg\min_{\alpha\in\mathbb{R}^n}\|y - K\alpha\|_2^2 + \lambda\alpha^\top K\alpha$
Solution: $\hat{\alpha} = (K + \lambda I)^{-1}y$

5.3 Non-Parametric Methods

Not by minimizing loss (classification and regression)

k-Nearest Neighbors

1. Given training set D
2. Pick k and a distance metric d in $\mathcal{X}$
3. Given x, find k nearest neighbors $x_1, \dots, x_k \in D$
4. Output majority vote $y_{i_1}, \dots, y_{i_k} \Leftrightarrow x_{i_1}, \dots, x_{i_k}$

Caution: sensitive to k, erratic for $d\uparrow$, needs large n

6 Neural Networks

Universal Approximation Theorem: Using any sigmoidal function in a NN, then any function can be approximated by a finite sum.

6.1 Activation Functions

Identity: $\phi(z) = z$ $\phi'(z) = 1$	Sigmoid: $\phi(z) = \frac{1}{1+e^{-z}}$ $\phi'(z) = \phi(z)(1 - \phi(z))$
Tanh: $\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$ $\phi'(z) = 1 - \phi^2(z)$	ReLU: $\phi(z) = max(0, z)$ $\phi'(z) = \begin{cases}0 & z \leq 0 \\ 1 & z > 0\end{cases}$

6.2 Forward Propagation (predict)

$f(\boldsymbol{x}; w, \theta) = \sum_{j=1}^p w_j\phi(\theta_j^\top\boldsymbol{x})$
Input Layer: $v_j^{(0)} = x_j$; $v_0^{(0)} = 1$, $\boldsymbol{v}^{(0)} = [\boldsymbol{x}; 1]$
Hidden Layer: $z_j^{(l)} = \sum_{i=0}^{n_{l-1}} w_{j,i}^{(l)}v_i^{(l-1)}$, $v_j^{(l)} = \phi(z_j^{(l)})$
$\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{v}^{(l-1)}$ and $\boldsymbol{v}^{(l)} = [\phi(\boldsymbol{z}^{(l)}); 1]$
Output: $f_j = \sum_{j=1}^{n_{L-1}} w_{j,i}^{(L)}v_j^{(L-1)}$, $\boldsymbol{f} = \boldsymbol{W}^{(L)}\boldsymbol{v}^{(L-1)}$
Regression: $y_i = f_i$, Classification: $y = \arg\max_j f_j$

6.3 Backward Propagation (train weights)

Output Layer:
Error: $\boldsymbol{\delta}^{(L)} = \nabla_{\boldsymbol{f}}\ell(\boldsymbol{f}, y) = [\ell'(f_1, y_1), \dots, \ell'(f_p, y_p)]$
Gradient: $\nabla_{W^{(L)}}\ell = \boldsymbol{\delta}^{(L)}\boldsymbol{v}^{(L-1)\top}$
Hidden Layer: ($\odot$ componentwise multiplication)
Error: $\boldsymbol{\delta}^{(l)} = \boldsymbol{\phi}'(\boldsymbol{z}^{(l)}) \odot (\boldsymbol{W}^{(l+1)\top}\boldsymbol{\delta}^{(l+1)})$
Gradient: $\nabla_{\boldsymbol{W}^{(l)}}\ell = \boldsymbol{\delta}^{(l)}\boldsymbol{v}^{(l-1)\top}$

6.4 Weight Initialization

Issues: vanishing/exploding gradients
Idea: Keep variance of $w_i \approx$ constant across layers:
Tanh: $w_{i,j} \sim \mathcal{N}(0, \frac{1}{n_{in}})$ or $w_{i,j} \sim \mathcal{N}(0, \frac{2}{n_{in}+n_{out}})$
ReLU: $w_{i,j} \sim \mathcal{N}(0, \frac{2}{n_{in}})$
n_{in}: # nodes previous layer, n_{out}: # nodes next layer

6.5 Optimization

Optimize weights: $\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}}\sum\ell(\boldsymbol{W}; \boldsymbol{x}_i, y_i)$
Update Rule: $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta_t\nabla_{\boldsymbol{W}}\ell(\boldsymbol{W}; \boldsymbol{x}, y)$
Minibatch: $\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta_t\frac{1}{|\mathcal{B}|}\sum_{(\boldsymbol{x},y)\in\mathcal{B}}\nabla_{\boldsymbol{W}}\ell(\boldsymbol{W}; \boldsymbol{x}, y)$
Momentum: $\boldsymbol{d} \leftarrow m\cdot\boldsymbol{d} + \eta_t\nabla_{\boldsymbol{W}}\ell(\boldsymbol{W}; \boldsymbol{x}, y)$, $W \leftarrow W - \boldsymbol{d}$
Decaying η_t: check $\frac{\Delta w}{\|w\|}$, too small $\eta\uparrow$, too large $\eta\downarrow$

6.6 Regularization (Avoid overfitting)

Weight Decay: Add penalty e.g L_2 regularization
Early stopping: Stop when validation error converges
Drop out: Randomly ignore hidden units during training with probability p, after training multiply weights by p to compensate
Batch Normalization: Normalize activations to zero mean and unit variance during training
Input: $v_i, \forall i \in S$, Learnable parameters: γ, β
Normalization: $\bar{v} = BN(v; \gamma, \beta)$

1. Mean: $\mu_S = \frac{1}{|S|}\sum_{i\in S}v_i$
2. Variance: $\sigma_S^2 = \frac{1}{|S|}\sum_{i\in S}(v_i - \mu_S)^2$
3. Normalize: $\hat{v}_i = \frac{v_i - \mu_S}{\sqrt{\sigma_S^2 + \epsilon}}$, 4. Scaling: $\bar{v}_i = \gamma\hat{v}_i + \beta$

Output: Replace v_i with $\bar{v}_i$

ResNets

Recurrent Networks have connections between all layers or skip connections to other layers. Allows for efficient training of large nets and introduces memory into the NN. Helps avoid vanishing gradients.

6.7 Convolutional Neural Networks (CNN)

NN robust against translation and less parameters
Next layer only depends on close inputs of prev. layer

$$\boldsymbol{z} = \boldsymbol{W} * \boldsymbol{x} = \sum_{j=\max(1,i-d+1)}^{\min(i,k)} w_j x_{i-j+1}, \quad \boldsymbol{v}^{l+1} = \phi(\boldsymbol{W} * \boldsymbol{v}^l)$$

Often used for image recognition:
Applying m $f \times f$ filter to a $n \times n$ image with p padding and s stride results in a $l \times l \times m$ with $l = \frac{n+2p-f}{s} + 1$ output. **Pooling:** Reduce size of feature maps by pooling units into one (mean, max, min)

7 Clustering (unsupervised classification)

7.1 k-means

Pick centers to minimize sum of squared distances:
$\min \hat{R}(\mu) := \sum_{i=1}^n \min_{\mu_j \in \mu} ||x_i - \mu_j||^2$
Non-convex, NP-hard (=not solvable). Lloyd:

1. initialize cluster centers $\mu^{(0)} = [\mu_1^{(0)}, \dots, \mu_k^{(0)}]$
2. repeat until convergence:
   (a) $z_i \leftarrow \arg\min_{j \in \{1,\dots,k\}} ||x_i - \mu_j^{(t)}||^2$
   (b) $\mu_j^{(t+1)} \leftarrow \frac{1}{|C_j^{(t)}|} \sum_{i \in C_j^{(t)}} x_i$

Allways convergence to local optimum. Cost per iter $\mathcal{O}(nkd)$. Dependent on initialization, finding good k is hard (Heuristic approaches, regularization, information theoretic basis, no CV), linear → kernelizing helps.

k-means++ (for data initialization)

1. choose first center uniformly from data points
2. for $j = 2, \dots, k$:
   (a) compute $D(x) = \min_{\mu \in \mu^{(j-1)}} ||x - \mu||^2$
   (b) choose next center with probability $\propto D(x)$
3. $\hat{R}(\mu_{k-means++}) \leq \mathcal{O}(\log(k)) \min_\mu \hat{R}(\mu)$

8 Dimensionality Reduction

Data: $X \in \mathbb{R}^{n \times d} \to Z \in \mathbb{R}^{n \times k}$ with $k \ll d$

8.1 Principal Component Analysis (PCA)

Centerd data: $\mu = \frac{1}{n}\sum_{i=1}^n x_i = 0, \quad x \in \mathbb{R}^d$
Empirical Cov: $\Sigma = \frac{1}{n}\sum_{i=1}^n x_i x_i^T = \frac{1}{n} X^T X \in \mathbb{R}^{d \times d}$

$$C^\star = \underset{W^TW=I_k}{\arg\min} \sum_{i=1}^n ||Wz_i - x_i||_2^2 \to C^\star = \sum_{i=k+1}^d \lambda_i$$

Optimal solution: Principal Eigenvectors of Σ.
$W^\star = (v_1|\dots|v_k)$, $1 \leq k \leq d$ and $z_i = W^T x_i$.
$W^\star$ orthogonal v_i to $\lambda_1 \geq \lambda_2 \geq \dots \lambda_d$
Solution minimizes reconstruction error (ℓ_2 distance)
Select k when x% of variance is explained. Represent any $X \in \mathbb{R}^{n \times d}$ as
$X = USV^T; U \in \mathbb{R}^{n \times n}, S \in \mathbb{R}^{n \times d}, V \in \mathbb{R}^{d \times d}$
$X^T X = V S^2 V^T$, V eigenvectors of $X^T X$

8.1.1 Kernel PCA

$\boldsymbol{w} = \sum_{j=1}^n \alpha_j \phi(\boldsymbol{x}_j)$, $\|\boldsymbol{w}\|_2^2 = \boldsymbol{\alpha}^T \boldsymbol{K} \boldsymbol{\alpha}$.
$\hat{\alpha} = \underset{\boldsymbol{\alpha}^T\boldsymbol{K}\boldsymbol{\alpha}=1}{\arg\max}\ \boldsymbol{\alpha}^T \boldsymbol{K}^T \boldsymbol{K} \boldsymbol{\alpha}$; $\boldsymbol{K}$: kernel matrix
Solution: $\boldsymbol{\alpha}^{(i)} = \frac{1}{\sqrt{\lambda_i}} \boldsymbol{v}_i$, ($\boldsymbol{v_i}$ eigenvectors of $\boldsymbol{K}$)
$\boldsymbol{z}_i = \sum_{j=1}^n \alpha_j^{(i)} k(\boldsymbol{x_j}, \boldsymbol{x}) \in \mathbb{R}^k$

8.2 Autoencoders

Idea: Learn representation of data by training ANN.
$f(x;\theta) = f_{dec}(f_{enc}(x;\theta_{enc});\theta_{dec})$
$f_{enc} : \mathbb{R}^d \to \mathbb{R}^k$, $f_{dec} : \mathbb{R}^k \to \mathbb{R}^d$
Optimize weights that output agrees with input:
$\boldsymbol{W}^\star = \min_{\boldsymbol{W}} \sum_{i=1}^n \|f(x_i; \boldsymbol{W}) - x_i\|_2^2$
If activation function is the identity, solution is PCA.

9 Probebalistic Modeling

9.1 Discriminative vs. Generative Models

Discriminative: y is a consequence of x: $p(y|x;\theta)$
Generative: x is a consequence of y (classification)
Model: $p(x,y;\theta) = p(x|y;\gamma) \cdot p(y;\pi)$, $\theta = (\gamma, \pi)$

9.2 Probebalistic Inference

9.2.1 Maximum Likelihood Estimation

Find $\mathbb{P}_X$ from data → MLE on data (unsupervised):
$\hat{\theta}_{ML} = \arg\max_\theta p(D|\theta) = \arg\min_\theta - \sum \log(p(x_i;\theta))$
Find $\mathbb{P}_{XY}$ using $\mathbb{P}_X$ and $\mathbb{P}_{Y|X}$ (supervised):
1. $\mathbb{P}_X$ MLE, 2. $\mathbb{P}_{Y|X}$ MLE, 3. $\mathbb{P}_{XY} = \mathbb{P}_X \mathbb{P}_{Y|X}$
Classification limit y: ($\hat{y} = \arg\max_{y \in \{1,-1\}} \hat{p}(y|x)$.)

9.2.2 Maximum A Posterior Estimation

Instead of $p(x;\theta)$ now $p(x|\theta)$ (θ is now a distribution)
Take knowledge of θ into account (prior)
Posterior distribution: $p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}$

$$\hat{\theta}_{MAP} = \arg\max_\theta p(\theta|D) = \arg\max_\theta p(D|\theta)p(\theta) = \arg\min_\theta \sum -\log(p(x_i;\theta)) - \log(p(\theta))$$

Unsupervised Learning

MAP: use $p(x|\theta)$ and $p(\theta)$ for MAP to find θ
Avrg.: $\hat{p}(x|D) = \mathbb{E}_{\theta|D}[p(x|\theta)] = \int p(x|\theta)p(\theta|D)d\theta$

Supervised Learning ($\mathbb{P}_{XY}$ using $\mathbb{P}_X$ and $\mathbb{P}_{Y|X}$)

MLE for Gaussian results in (weighted) LS problem.
MLE with logistic model results in logistic regression.
MAP results in regularized MLE problem.
Prior: $y_i = w^\top x_i + \epsilon_i, \epsilon_i \sim \mathcal{N}(0,1), w \sim \mathcal{N}(0, \sigma_w^2)$
Result: $\hat{w}_{MAP} = \arg\min_w \frac{1}{2}\|y - Xw\|_2^2 + \frac{1}{\sigma_w^2}\|w\|_2^2$
Laplace prior: $p(w) = \prod_{i=1}^d \frac{\lambda}{2} e^{-\lambda|w_i|}$
Result: $\hat{w}_{MAP} = \arg\min_w \frac{1}{2}\|y - Xw\|_2^2 + \lambda\|w\|_1$

9.3 Decision Theory and Decision Rules

Decision theory: decision rules $a : X \to A$, A is an action set. Want to find an $a^\star$ that minimizes:
$a^\star = \arg\min_a \mathbb{E}[L(a(X), Y)]$
asymetric 0-1 loss with abstention (r):
$\ell(\tilde{y}(x), y) = I_{\tilde{y}(x) \neq y} I_{\tilde{y}(x \neq r)} + c I_{\tilde{y}(x) = r}$
Solution: $\hat{y}(x) = r$, for $c < \hat{p}(y = -1|x) < 1 - c$

9.4 Gaussian Bayes Classifier (GBC)

Prior: $Y \sim \text{Cat}(\pi)$, $p(Y = y; \pi) = \pi_y$
Feature Distribution: $p(x|y) \sim \mathcal{N}(x; \mu_y, \Sigma_y)$
MLE (Gaussian case): $\hat{\pi}_j = \hat{p}_j = \frac{\#Y=j}{n}$
$\hat{\mu}_y = \frac{1}{\#Y=y} \sum_{i:Y_i=y} x_i$
$\hat{\Sigma}_y = \frac{1}{\#Y=y} \sum_{i:Y_i=y} (x_i - \hat{\mu}_y)(x_i - \hat{\mu}_y)^T$
Gaussian Naive Bayes: $\Sigma_Y = \text{diag}(\sigma_{y,1}^2, \dots, \sigma_{y,k}^2)$
MLE: $\hat{\sigma}_{y,k}^2 = \frac{1}{\#Y=y} \sum_{i:Y_i=y} (x_{i,k} - \hat{\mu}_{y,k})^2$
Linear Discriminant Analysis (LDA): $\Sigma_Y = \Sigma$
Fishers LDA: $c = 2, p_1, p_2 = 0.5, \Sigma_1 = \Sigma_2$
Quadratic Discriminant Analysis: π_y is uniform

10 Gaussian Mixture Model

10.1 Gaussian Mixtures

Multiple Gaussians: $p(x|y) = \sum_{k=1}^K w_k \mathcal{N}(x|\mu_k, \Sigma_k)$,
w_k mixing coefficient, $\sum_{k=1}^K w_k = 1$ and $w_k \geq 0$.
$(\mu^\star, \Sigma^\star, w^\star) = \arg\min_{\mu,\Sigma,w} - \sum_{i=1}^N \log(p(x|y))$
Solving with SGD hard → use EM.

10.2 Expectation Maximization (EM)

10.2.1 Hard-EM: assign fixed lables

E-Step: predict most likely class for each data point.
$z_i^{(t)} = \arg\max_z P(z|x_i, \theta^{(t-1)})$
$= \arg\max_z P(z|\theta^{t-1}) P(x_i|z, \theta^{(t-1)})$
M-Step: MLE with new $D^{(t)} = \{(x_i, z_i^{(t)})\}_{i=1}^N$
Use GBC: $\theta^{(t)} = \arg\max_\theta P(D^{(t)}|\theta)$
Problem if multiple clusters are overlapping.

10.2.2 Soft-E: assign probabilities

E-Step: calculate cluster membership weights:

$$\gamma_j^{(t)}(x_i) = p(z = j|x_i, \Sigma^{(t-1)}, \mu^{(t-1)}, w^{(t-1)}) = \frac{w_j^{(t-1)} \mathcal{N}(x_i|\mu_j^{(t-1)}, \Sigma_j^{(t-1)})}{\sum_{k=1}^K w_k^{(t-1)} \mathcal{N}(x_i|\mu_k^{(t-1)}, \Sigma_k^{(t-1)})}$$

M-Step: fit clusters to weighted data points:

$$w_j^{(t)} = \frac{1}{N} \sum_{i=1}^N \gamma_j^{(t)}(x_i), \quad \mu_j^{(t)} = \frac{\sum_{i=1}^N \gamma_j^{(t)}(x_i) x_i}{\sum_{i=1}^N \gamma_j^{(t)}(x_i)}$$

$$\Sigma_j^{(t)} = \frac{\sum_{i=1}^N \gamma_j^{(t)}(x_i)(x_i - \mu_j^{(t)})(x_i - \mu_j^{(t)})^T}{\sum_{i=1}^N \gamma_j^{(t)}(x_i)} (+\nu^2 I)$$

k-finding: Same as k-clusters, CV works fairly well.
Spherical: $\Sigma_i = \sigma_i^2 I$, k parameters
Diagonal: $\Sigma_i = \text{diag}(\sigma_{i,1}^2, \dots, \sigma_{i,d}^2)$, kd parameters
Tied: $\Sigma_i = \Sigma$, $d + d(d+1)/2$ parameters
Full: Σ_i arbitrary $d + d(d+1)/2$ parameters

Degeneracy & Facts

"optimal" GMM chooses $k = n$ at each x_i ($\sigma_i \to 0$). Loss converges to $-\infty$ → **Overfitting**. Omitted by adding diagonal term $\nu^2 I$ to $\Sigma_j^{(t)}$. (ν by CV).
Always converges, always increases likelihood

10.3 Gaussian Mixture Bayes Classifier

Each class is also a mix if Gaussians.
Given labeled dataset: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$
Estimate class prior: $P(y) = p_y$
Estimate conditional distr. for each class (GMM):
$P(x|y) = \sum_{k=1}^{K_y} w_{y,k} \mathcal{N}(x|\mu_{y,k}, \Sigma_{y,k})$
For classification use Bayes rule:
$P(y|x) = \frac{1}{z} P(y) P(x|y)$ with: $z = \sum_{y'} P(y') P(x|y')$

10.4 GMM for density estimation

Used for data imputation and anomaly detection.

10.4.1 Anomaly detection: $p(x) < \epsilon$

Choose ϵ to trade-off false positives and false negatives. Use ROC curve (F1-score) and CV to optimize ϵ.

10.4.2 Data imputation:

Semi-supervised learning (unlabeled & labeled data).
First learn clusters from unlabeled data.
Then use labeled data to match clusters to labels.
Use EM algorithm for labeled data:
During E-step: $\gamma_j^{(t)}(x_i)[j = y_i] = \begin{cases} 1 & \text{if } j = y_i \\ 0 & \text{else} \end{cases}$

11 Generative Pretrained Transformer

Embedding and word position: $\boldsymbol{Z}_0 = \boldsymbol{X} W_e + W_p$
W_e learnable word matrix, W_p fixed position matrix
Transformer: $\boldsymbol{Z}_l = \text{transformer\_block}(\boldsymbol{Z}_{l-1}) \, \forall i$
Prediction: $\boldsymbol{P} = \text{softmax}(\boldsymbol{Z}_n W_e^T)$

11.1 Transformer Block

Main part is *Masked Multi Self Attention*:
Attention: learns to predict weighted directed graph.
$\boldsymbol{z}_i^{l+1} = \sum_{j=1}^n \alpha_{ij} \boldsymbol{z}_j^l$ with $\alpha_{ij} = $ score
Can be viewed as learnable lookup table

12 Appendix

12.1 Linear Algebra

Positive Semi-Definite: M is p.s.d $\Leftrightarrow \forall x \in \mathbb{R}^n$: $x^T M x \geq 0 \Leftrightarrow \lambda_i \geq 0$, Gramm Matrix: $X^T X$ p.s.d.
Invertable Matrix M invertable $\Leftrightarrow$ M full rank $\Leftrightarrow \det(M) \neq 0 \Leftrightarrow \lambda_i \neq 0$
Projection Matrix: P is projection matrix $\Leftrightarrow P^2 = P$
Transp.: $(ABC)^T = C^T B^T A^T$, $(A+B)^T = A^T + B^T$
Inv.: $(ABC)^{-1} = C^{-1} B^{-1} A^{-1}$, $(A^T)^{-1} = (A^{-1})^T$
Trace: $\text{tr}(A) = \sum_i a_{ii}$, $\text{tr}(ABC) = \text{tr}(BCA)$
Scalar Product: $\langle \boldsymbol{v}, \boldsymbol{v} \rangle = \boldsymbol{v}^T \boldsymbol{v} = \sum_i v_i^2 = \|v\|_2^2$

Matrix Differentiation

$\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} = (\boldsymbol{A} + \boldsymbol{A}^T) \boldsymbol{x} \overset{A^T = A}{=} 2\boldsymbol{A}\boldsymbol{x} = 2\boldsymbol{x}^\top \boldsymbol{A}$
$\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}^T \boldsymbol{A} = \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{A} \boldsymbol{x} = \boldsymbol{A}$
$\nabla_w \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2 + \lambda\|\boldsymbol{w}\|^2 = 2(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^\top \boldsymbol{X} + 2\lambda \boldsymbol{w}^\top$
Hessian: $H = \nabla_x^2 A(x) \to H_{i,j} = \frac{\partial^2 A(x)}{\partial x_i \partial x_j}$

12.2 Analysis

$\frac{d}{dx} f(x)g(x) = f'(x)g(x) + f(x)g'(x)$
$\frac{d}{dx} f(g(x)) = f'(g(x))g'(x)$
$h(x) = f(x)/g(x) \to h'(x) = \frac{f'(x)g(x) - f(x)g'(x)}{g(x)^2}$
Gradient: $\nabla_x f(x) = \left[\frac{\partial}{\partial x_1} f(x) \quad , \dots , \quad \frac{\partial}{\partial x_n} f(x) \right]^T$

Convexity

$\mathcal{L}$ is convex if $\forall w_1, w_2 \in \mathbb{R}^d, \forall \lambda \in [0,1]$:
$\mathcal{L}(\lambda w_1 + (1-\lambda) w_2) \leq \lambda \mathcal{L}(w_1) + (1-\lambda)\mathcal{L}(w_2)$
1. order: $\mathcal{L}(w_2) \geq \mathcal{L}(w_1) + \nabla \mathcal{L}(w_1)^\top (w_2 - w_1)$
2. order: $\nabla^2 \mathcal{L}(w) \succeq 0$ (positive semi-definite)
$\alpha f + \beta g$ if $\alpha, \beta \geq 0$ and f, g convex

12.3 Probability Theory

$F(x) = P(X \leq x) = \int_{-\infty}^x f(t)dt \to p(x) = \frac{dF(x)}{dx}$
Bayes: $p(x|y) = \frac{p(y|x) \cdot p(x)}{p(y)} = \frac{p(y|x) \cdot p(x)}{\sum_x p(y|x) p(x)}$
Conditional: $p(x,y) = p(x|y)p(y) = p(y|x)p(x)$
Gaussian: $p(x) = \frac{1}{\sqrt{2\pi \det(\Sigma)}} \exp\left(-\frac{(x-\mu)^T \Sigma^{-1} (x-\mu)}{2\sigma^2}\right)$
$Ax + By + c \to \mathcal{N}(A\mu_x + B\mu_y + c, A\Sigma_x A^T + B\Sigma_y B^T)$
Expected Value: $\mathbb{E}[X] = \sum_X x \cdot p(x)$
$\mathbb{E}_{xy}[a + bx + cy] = a + b\mathbb{E}_x[x] + c\mathbb{E}_y[y]$
Variance: $\mathbb{V}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2$
$\mathbb{V}_{xy}[a + bx + cy] = b^2 \mathbb{V}_x[x] + c^2 \mathbb{V}_y[y] + 2bc\text{Cov}_{xy}[x,y]$
Conjugate Prior: Prior $p(\theta)$ and Posterior $p(\theta|x)$ are in the same family of distributions (Likelihood $p(x|\theta)$) → exact inference