IML Summary

Lasse Fierz - lfierz

Version: 9. Februar 2023

Basics

- General p-norm: $||x||_p = (\sum_{i=1}^n |x_i|^p)^{1-p}$
- Taylor: $f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \mathbb{O}(x^3)$
- Power series of exp.: $exp(x) := \sum_{k=0}^{\infty} \frac{x^k}{k!}$
- $\sum_{k=0}^{\infty}(xy)^k = \frac{1}{1-xy}$
- Entropy: $\mathsf{H(X)} = \mathbb{E}_X[-log\mathbb{P}(X=x)]$
- KL-Divergence: $D_{KL}(P||Q) = \sum_{x\in\mathbb{X}} P(x) log\left(\frac{P(x)}{Q(x)}\right) \geq 0$
- $1 - z \leq exp(-z)$
- Cauchy-Schwarz: $|\mathbb{E}[X,Y]|^2 \leq \mathbb{E}(X^2)\mathbb{E}(Y^2)$
- Jensens Inequality: for a convex f(X): $f(\mathbb{E}(X)) \leq \mathbb{E}(f(X))$
- M p.s.d. if $v^T M v \succeq 0$

Probability Theory:

- Gaussian: $\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2})$
- $(N)(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}} exp(-\frac{1}{2}(x-\mu)^T\boldsymbol{\Sigma}^{-1}(x-\mu))$
- $X \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma}), Y = A + BX \Rightarrow Y \sim \mathcal{N}(A + B\boldsymbol{\mu}, B\boldsymbol{\Sigma}\boldsymbol{^{-1}}B^T)$
- Binomial Distr.: $\mathbb{P}(X=k) = \binom{n}{k} p^k (1-p)^{n-k}$
- $\mathbb{V}(X) = \mathbb{E}\left[(X - \mathbb{E}(X))^2\right] = \mathbb{E}(X^2) - [\mathbb{E}(X)]^2$
- $\mathbb{V}[X+Y] = \mathbb{V}[X] + \mathbb{V}[Y] + 2Cov(X,Y)$
- $Cov(X,Y) = \mathbb{E}[(X - \mathbb{E}(X))(Y - \mathbb{E}(Y))]$
- $Cov(aX, bY) = abCov(X,Y)$

Calculus

- $\int uv'dx = uv - \int u'vdx$ • $\frac{\partial}{\partial x}\frac{g}{h} = \frac{g'h}{h^2} - \frac{gh'}{h^2}$
- $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{b^T A x}) = A^T b$ • $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{b^T x}) = \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x^T b}) = b$
- $\frac{\partial}{\partial \boldsymbol{X}}(\boldsymbol{c^T X^T b}) = \boldsymbol{bc}^T$ • $\frac{\partial}{\partial \boldsymbol{X}}(\boldsymbol{c^T X b}) = \boldsymbol{cb^T}$
- $\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x^T A x}) = (\boldsymbol{A^T} + \boldsymbol{A})x \overset{\text{A sym.}}{=} 2\boldsymbol{Ax}$
- $\frac{\partial}{\partial \boldsymbol{X}} Tr(\boldsymbol{X^T A}) = A$ • Tr.trick: $\boldsymbol{x^T A x} \overset{\text{inner prod.}}{=} TR(\boldsymbol{x^T A x}) \overset{\text{cyclic perm.}}{=} Tr(\boldsymbol{x x^T A}) = Tr(\boldsymbol{A x x^T})$
- $\frac{\partial \boldsymbol{A^{-1}}}{\partial x} = -\boldsymbol{A}^{-1}\frac{\partial \boldsymbol{A}}{\partial x}\boldsymbol{A}^{-1}$
- $|X^{-1}| = |X|^{-1}$ • $\frac{\partial}{\partial \boldsymbol{X}} log|x| = x^{-T}$ • $\frac{\partial}{\partial x}|x| = \frac{x}{|x|}$
- $\frac{\partial}{\partial \boldsymbol{x}}||x||_2 = \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{x^T x}) = 2x$
- $\frac{\partial}{\partial \boldsymbol{x}}||\boldsymbol{x} - \boldsymbol{b}||_2 = \frac{\boldsymbol{x}-\boldsymbol{b}}{||\boldsymbol{x}-\boldsymbol{b}||_2}$
- $\frac{\partial}{\partial \boldsymbol{x}}||x||_1 = sgn(x)$
- $\sigma_{\text{sigmoid}}(x) = \frac{1}{1+exp(-x)} \Rightarrow$
- $\nabla\sigma_{\text{sigmoid}}(x) = \sigma(x)(1-\sigma(x)) = \sigma(x)\sigma(-x)$
- Jacobian $= \frac{d}{d\boldsymbol{x}} f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \dots & & \dots \\ \frac{\partial f_n}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$
- Hessian: $\nabla^2 f(x)$ e.g. for reg. $2(X^TX + \lambda\mathbb{I}_d)$
- $tanhx = \frac{2sinhx}{2coshx} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- $\nabla tanhx = 1 - tanh^2 x$
- $sin(a \pm b) = sin(a)cos(b) \pm cos(a)sin(b)$
- $cos(a \pm b) = cos(a)cos(b) \mp sin(a)sin(b)$

(Linear) Regression

General Regression: find $\hat{y} = f(x) \leftrightarrow \min_{\hat{y}(x)} ||y - \hat{y}(x)||_2^2$.
$f^*(x)$ that minimizes L is $\mathbb{E}[X|\boldsymbol{X} = \boldsymbol{x}]$ called **Bayes Optimal Predictor**. In practice unattainable.
$f(x) = \omega x$ or nonlinear **base fct**: $f(x) = \omega\phi(x)$
Multidim.: $L = \min_{\omega} ||\boldsymbol{Y} - \boldsymbol{X\omega}||^2$,
$Y \in \mathbb{R}^n, x \in \mathbb{R}^{nxd}, \omega \in \mathbb{R}^d$

Closed Solution

If X^TX is invertible (X^TX has full rank $\Leftrightarrow rank(X) = min(d,n)) \Rightarrow$ closed solution: $\omega = \boldsymbol{(X^TX)^{-1}X^TY}$
∇L is $\mathbb{O}(nd)$, closed solution is $\mathbb{O}(nd^2)$.
Alternatively geom. proj.: $(\boldsymbol{y - X\hat{\omega}})^T X\omega = 0$

Optimization

If not solvable in closed form or expensive to invert $X^TX \rightarrow$
$\omega_{t+1} \leftarrow \omega_t - \eta\nabla L(\boldsymbol{\omega_t})$, η is the learning rate.
Convergence guaranteed for $\eta \geq \frac{2}{\lambda_{max}}$, where λ_{max} is the max EV of X^TX.
X^TX diagonal $\Rightarrow$ contour lines (L const) are ellipses

Nonlinear Regression

Note: When working with NNs both the weights and the nonlinear functions are chosen.
For closed solution same applies rank$\phi(x) \overset{!}{=} min(n,p)$

Regularization

Among all unbiased solutions $(X^TX)^{-1}X^TY$ is the solution that has the smallest variance $\Rightarrow$ minimizes gen. Error
One can set the ω of higher order features manually to zero ($\leftrightarrow$ choose a less complex model) or
Ridge Regression $min||Y - X\omega||^2 + \lambda||\omega||^2$
Always allows for closed solution and lets LS converge faster (EVs of Hessian X^TX change)
Equivalent to performing Bayesianism approach with $p(\omega) = \mathcal{N}(\omega|0, \boldsymbol{\Lambda}^{-1})$ or linearly $p(\omega) = \mathcal{N}(\omega|0,1)$
Lasso Regression $\min_{\omega} ||Y - X\omega||^2 + \lambda|\omega|$
Convex loss but no closed form solution (not differentiable)
Bayes. w/ Laplacian prior: $p(\omega_i) = \frac{\lambda}{4\sigma^2} exp(-|\omega_i|\frac{\lambda}{w\sigma^2})$
$\omega_{\text{high}} \rightarrow 0 \Rightarrow$ sparse, λ_{opt} through CV.

Gradient Descent and Convexity

Gradient Descent

$\omega_{t+1} \leftarrow \omega_t - \eta L(\omega_t)$
Converges to a stationary point. $\nabla L(\omega) = 0 \Rightarrow$ GD stuck.
Complex fcts: $\nabla L(\omega)$ from lin. approx. and use small η
Large EVs $\leftrightarrow$ Dominant feature. Well conditioned if λ_{max} and λ_{min} are in similar range.
GD is sometimes slower and less accurate but there is more control and less comp. complexity
Gradient Methods: Momentum usage, Adaptive Methods, 2nd order methods
Stochastic GD: Use subsample from data for update step. Helps against saddle point conversion.

Convexity

Always:

- global min/max $\Rightarrow$ local min/max
- local min/max $\Rightarrow$ stationary point
- $L(\omega) < L(v) \forall v \neq \omega \Leftrightarrow \omega$ is a global min

Convexity:

- 0-order condition: $L(sw + (1-s)v) \leq sL(w) + (1-s)L(v)$ aka function is lower or equal to linear connection of two points.
- 1st-order: $L(v) \geq L(\omega) + \nabla L(\omega)^T(v - \omega)$ aka any point v on function is higher than point on linear approximation drawn at position ω
- 2nd-order: $\nabla^2 L(\omega)$ is p.s.d. aka non-neg. curvature throughout function.
- ω stationary $\Rightarrow \omega$ is local minimum
- ω is local minimum $\Rightarrow \omega$ is global minimum

Strong Convexity:

- 0-order: $L(sw + (1-s)v) + \epsilon \leq sL(w) + (1-s)L(v)$ so fct always a bit below linear connection of points
- 1st-order: same as convex
- 2nd-order: strictly positive curvature always
- ω is global minimum $\Rightarrow L(\omega) < L(v) \forall v \neq \omega$
- Only one global minimum

Convexity Operations:

- Linear Comb. of convex functions are convex
- $f(g(x))$ is convex if f convex and g affine or f non-decreasing and g convex.
- Convex + strictly convex fct. = strictly convex fct

Model Selection

In general $y = f(x) + \epsilon$, where ϵ is random noise
We can never know $f(x)$ as we can only observe y.
We use the gen. error $(y - \hat{f}(x))^2 =$
$\underbrace{(f(x) - \hat{f}(x))^2}_{\text{estimation error}} + \underbrace{\epsilon^2}_{\text{irreducible noise}} \underbrace{- 2\epsilon(\hat{f}(x - f(x)))}_{\text{0 on average}}$
Often interested in $\mathbb{E}\left[(y - \hat{f}(x))^2\right] \approx \underbrace{\frac{1}{n}\sum_{i=1}^n (y_i - \hat{f}(x_i))^2}_{\text{empirical error}}$

Bias and Variance

- **Bias** $= \mathbb{E}\left[(f(x) - \hat{f}(x))^2\right]$
- **Variance** $= \mathbb{E}\left[(\hat{f}(x) - \mathbb{E}\hat{f}(x))^2\right]$ fluctuation of $\hat{f}$

If $y = f(x)$ a complex model can still overfit. (Data sample)
Generalization Error = bias^2 + **variance**

Cross Validation

Split training data into k batches

1. For each option of hyperparameter:
2. for each batch:
   - Train model on the whole training data except for the batch
   - Calculate validation error on remaining batch
3. Average validation error over all batches
4. Choose hyperparameter with lowest avg. val. error
5. Train model with that hyperparameter on the whole training set
6. Determine test error

Leave one out CV (LOOCV):

- Batch size 1
- Results in best model approximation
- Validation error bad (only one sample) but avg. ok
- Computationally expensive

For $n < d$ GD finds solution that minimizes $||\omega||_2$

Classification

- Probabilistic generative: p(x,y) (gen)
- Prob. discriminative: p(y—x) (certainty)
- Purely discr. c: $X \rightarrow y$ (easiest)

Lin. seperable data $\Rightarrow$ infinitely many solutions $\Rightarrow$ SVM

Loss Functions

- Cross Entropy: $\mathcal{L}^{CE} = -\left[y' log\hat{f}(x)' + (1-y')log(1-\hat{f}(x)')\right]$
  Where $y' = \frac{1+y}{2}$ and $\hat{f}(x)' = \frac{1+\hat{f}(x)}{2}$

- Zero one loss: $\mathbb{L}^{0/1} = \mathbb{I}\{sign(\hat{f}(x) \neq y)\}$
  Not convex nor continuous $\Rightarrow$ surrogate logistic loss
- $\mathbb{L}^{\text{Hinge}} = \max(0, 1 - y\hat{f}(x))$
- $\mathbb{L}^{\text{percep}} = \max(0, -y\hat{f}(x))$
- $\mathbb{L}^{\text{logistic}} = log(1 + exp(-y\hat{f}(x)))$
- multidim. logistic loss: softmax:
  $\mathbb{L}_i^{\text{softmax}} = \frac{e^{-af_i}}{\sum_{j=1}^{K} e^{-af_j}}$
- $\mathbb{L}^{\text{exp}}(x)_i = exp(-y\hat{f}(x))$

GD on logistic loss: $\rightarrow$ SVM sol.
Worst group error(related to group fairness): Highest error among all clusters of a class (e.g. if one blob is 100% false)
Robust generalization w.r.t. perturbations
Data augm., models that allow for invariance (e.g. CNNs)
Distribution shifts train $\neq$ test data $\Rightarrow$ try for sim. samples

SVM

margin $= \min_i y_i\langle\omega, x_i\rangle$, distance DB-SV $= \frac{y_i\langle\omega,x_i\rangle}{||\omega||}$
$\underset{\omega}{\text{argmax}}$ margin(ω)s.t. $||\omega|| = 1$ or $||\omega|| = \frac{1}{||margin||}$
Latter case: smaller subspace to look for ω
Objective: $\mathcal{L}$(soft margin) $=$

$$\min_{\omega,\xi} \frac{1}{2}||\omega||^2 + C\sum_i \xi_i$$
$$\text{s.t. } y_i\omega^T x_i \geq 1 - \xi_i \text{ and } \xi_i \geq 0 \ \forall i = 1, .., n$$

Solve using lagrangian:

$$\mathcal{L} = \frac{1}{2}||\omega||^2 + C\sum_{i=1}^{n} \xi_i + \sum_{i=1}^{n} \alpha_i(1 - \xi_i - y_i\omega^T x_i)$$

Kernels

At least one nonlinear $\phi(x)$ then $\hat{f}(x)$ can be nonlinear
Complexity of constructing $\phi(x)$ is $\mathcal{O}(nd^m) \Rightarrow$ huge

Kernel Trick

Feature maps only enter $\hat{f}(x)$ by their inner product.
Can write one of the possible global minimizers $\hat{\omega} = \phi^T a$, $a \in \mathbb{R}^n \Rightarrow$Can write objective as:
$L(\omega) = \frac{1}{n}\sum_{i=+}^{n} l$

Other Nonlinear Models

K Nearest Neighbours

- For each datapoint assign majority class of knns
- Heavily dependent on k $\Rightarrow$ Cross Validation
- Error prone in high dim. because of large distances
- Needs a lot of data but $\mathcal{O}(nd)$ can be reduced to $\wr(n^p), p < 1$ if we allow for some error probability

Decision Trees

- At each node split w.r.t. to one feature and threshold
- Each node returns class of the subset by majority
- Greedy Method: best step for current situation as opposed to generally best step $\Rightarrow$ errors propagate.
- Prone to overfitting as partitions can get very detailed
- $\Rightarrow$ random forest (averaged over random splits)

Kernel Operations

- $\alpha x^T x'$
- Polynomial: $\alpha(x^T x' + \beta\mathbb{I})^p$
- RBF(Gaussian): $exp(-\frac{||x-x'||_2^2}{h^2})$
- Sigmoid: $tanh(\kappa x^T x' - b)$

For k1, k2 valid:

- $ak_1(x, x')$ for $a \in \mathbb{R}$
- $k_1(x, x') + k_2(x, x')$
- $k_1(x, x')k_2(x, x')$
- $f(x)k_1(x, x')f(x')$
- $k(\phi(x), \phi(x'))$
- $g(k_1)$ with g: exp. or polyn. with all pos. coeff.

Neural Networks

NNs allow us to choose the feature maps in the model itself

$$z_1^{(1)} = \sum_{j=1}^{d} \omega_{1,j}^{(1)} x_j + \omega_{1,0}^{(1)}$$
$$z_k^{(l)} = \sum_{j=1}^{n_{l-1}} \omega_{k,j}^{(l)} v_j^{(l-1)} + \omega_{k,j}^{(l)} \quad v_l = \varphi(z_l)$$

Vector notation:

$$\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{v}^{(l-1)} + \boldsymbol{W}_0^{(l)}$$
$$f(\boldsymbol{x}) = \boldsymbol{W}^{(L)}\boldsymbol{v}^{(L-1)}$$
where $\boldsymbol{v}^{(l)} = \left[\varphi(\boldsymbol{z}^{(l)}; 1)\right]$ φ applied comp. wise

$\rightarrow$ optimize weights w.r.t. loss fct (squared loss, CE etc.)

$$l(\boldsymbol{W}; \boldsymbol{x}, y) = \sum_{i=1}^{n} l_i(\boldsymbol{W}; \boldsymbol{x}_i, y_i)$$

Universal approx. theorem: Any cont. fct can be approximated by a finite layered NN with sigmoidal act. function.
Weight decay reduces complexity

Activation Functions

A neural network with one hidden layer and nonlinear activation functions can approximate every continuous function.

- Sigmoid: $\varphi(z) = \frac{1}{1+exp(-z)} = \frac{exp(z)}{1+exp(z)}$, $\varphi'(z) = \varphi(z)(1 - \varphi(z))$
- Relu: $\varphi(z) = max(0, z)$ (vanishing grad. for z < 0)
- Tanh: $tanh(z) = \frac{exp(z)-exp(-z)}{exp(z)+exp(-z)}$
- ELU$_\alpha$ (exp. relu): $\begin{cases} \alpha(exp(z) - 1), & \text{if } z < 0 \\ z, & \text{if } z \geq 0 \end{cases}$
- Softmax: $\varphi(z_i) = \frac{exp(z_i)}{\sum_j exp(z_j)}$

Backpropagation

Can reuse computations from forward propagation and from layer l_{i+1} ... to compute $W^{(i)}$

$$(\nabla_{W^{(i)}} l)^T = \underbrace{\frac{\partial l}{\partial f}}_{\delta(L)} \frac{\partial f}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial z^{(L-2)}} \cdots \frac{\partial z^{(i+1)}}{\partial z^{(i)}} \underbrace{\frac{\partial z^{(i)}}{\partial W^{(i)}}}_{v^{(i-1)}}$$
$$\nabla_{W^{(l)}} l = \boldsymbol{\delta}^{(l)}\boldsymbol{v}^{(l-1)},\ \boldsymbol{\delta}^{(l)} = diag(\varphi'(z^{(l)}))W^{(l+1)})\boldsymbol{\delta}^{(l)}$$
$$\omega_{L-l}^{(t+1)} \leftarrow \omega^{(t)} - \eta\frac{\partial l}{\omega_{L-l}}$$

Note usually minibatches are used for cum. weight updates.
Runtime grows linearly with num of params in feed forward
Modifications:

Momentum: initialize $d = 0$
$$\boldsymbol{d} \leftarrow m * \boldsymbol{d} + \eta_t \nabla_W l(\boldsymbol{W}; \boldsymbol{x}, y)$$
$$W \leftarrow W - \boldsymbol{d}$$

Vanishing / Exploding Gradient

Potential reasons:

- $||\delta^{(i)}|| \rightarrow 0|\infty$ or $||v^{(i)}|| \rightarrow 0|\infty$
- Relu (no saturation) can help avoid $\delta \rightarrow 0$
- Note δ only depends on φ' while v depends on φ
- Use standardized input and / or batch normalization
- Weight init. matters as weight opt. is gen. non-convex

Regularization in NNs

Reg. term in loss fct., early stopping, dropout or data augmentation
Batch norm. Normalize unit activations for a laye:

- $\mu_s = \frac{1}{|S|}\sum_{i \in S} v_i$
- $\sigma_S^2 = \frac{1}{|S|}\sum i \in S(v_i - \mu_S)^2$
- $\hat{v}_i = \frac{v_i - \mu_S}{\sqrt{\sigma_S^2 + \epsilon}}$
- Scale and shift: $\bar{v}_i = \gamma\hat{v}_i + \beta$, γ, β given params.

Speeds up and stabilizes training. Solves covariate shift, reduces importance of weight init. Introduces exploding gradients. Mild reg. effect because of "random" batch size.

Convolutional Neural Networks

Less params than FC for images / audio

- Invariant regarding shifts, scale and rotation
- Updates still through backpropagation
- Still need nonlinear act. fct. for nonlinear functions

Dimension of image after CNN layer image: n x n, m kernels of size k*k, stride s, padding p:

$$l = \frac{n+2p-k}{s} + 1$$

(Max)Pooling: Strongly reduces number of parameters

Residual NNs

- Helps avoid vanishing gradients
- Allows for very deep NNs (1000+ layers)

Clustering

K-Means

$$\text{Minimize } \hat{R}(\mu) = \sum_{i=1}^{n} \min_{j=1,..,k} ||x_i - \mu_j||^2$$

K-means algorithm / Lloyds heuristics:
Initialize centers $\mu \rightarrow$ until convergence:
assign points to centers $\rightarrow$ relocate centers

- Risk function monotonically decreasing
- Converges to local minimum
- Cost per iteration: $\mathcal{O}(n * k * d)$, (k clusters)
- Can use kernels for random shapes
- Depends on init, Local convergence, Choice of k?

K++ seeding: Set one center $\mu_1 \rightarrow$, centers 2-k:

$$\mu_j^{(0)} := x_i \text{ with prob. } \min_{l=1,..,j-1} ||x_i - \mu_l^{(0)}||^2$$

Expected loss is $(O)(logk)$ times that of opt. k-means sol.
Note: finding optimal k is difficult. Can use heuristics
Generalization is very good(unsupervised learning)

Dimensionality Reduction

PCA

Finding linear transform to lower dimension that maximizes variance of data. Typically assume centered data.
$\pi(x) = \mu^T x \Rightarrow$ maximize $\mu^T \frac{1}{n}\sum_i (\bar{x} - x_i)^2\mu, ||\mu|| = 1$
Note: we set $||\mu|| = 1$ to resolve uniqueness issue

$$\Rightarrow \mathcal{L} = \mu^T \frac{1}{n}\sum_i (\bar{x} - x_i)^2\mu + \lambda(1 - ||\mu||)$$
$$\frac{\partial\mathcal{L}}{\partial\mu} \Rightarrow \underbrace{Var(x)}_{S}\mu = \lambda\mu \Rightarrow \lambda \text{ is highest EV of S}$$
μ is the respective Eigenvector
Note S becomes $\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$ when centered

Multidim: Closed form solution also for $k > 1$

First PC with $X_1 = x$ as for $d = 1$, then
$X_2 = X_1 - proj_{\mu_1} X_1$
Then PCA with X_2, $X_3 = X_2 - proj_{\mu_2} X_2$,...
$\Rightarrow \pi(x) = (x^T\mu_1, .., x^T\mu_d)$

Alternatively: $\min_{W,||W||^2=1} \sum_{i=1}^{n} ||\boldsymbol{x}_i - z_i\boldsymbol{\omega}||^2 \rightarrow$ Regr.
$z_i^* = \boldsymbol{\omega}^T\boldsymbol{x}_i \rightarrow \boldsymbol{\omega}^* = arg\min_{||\omega||_2=1} \sum_{i=1}^{n} ||\boldsymbol{\omega}\boldsymbol{\omega}^T\boldsymbol{x}_i - \boldsymbol{x}_i||_2^2$
$\rightarrow arg\min_{||\omega||_2=1} \boldsymbol{\omega}^T\boldsymbol{\Sigma}\boldsymbol{\omega}$, $\boldsymbol{\Sigma} = \sum_{i=1}^{n} \lambda_i v_i v_i^T$
PCA - K-means: Pretty much the same problem statement but for PCA $||\omega|| = 1$ and $z_i \in \mathbb{R}^k$ while for k-means z_i are unit vectors
Nonlinear PCA: same with $\phi(x)$

Statistical Perspective

Frequentism

Frequentist approach: $P(Y|X, \theta) \rightarrow MLE$
$\theta^* = \underset{\theta}{argmin} - log\sum_{i=1}^{n} p(y_i|x_i, \theta)$

- Consistency (Convergence to true params.)
- Asymptotic efficiency (smallest var $\forall$ well behaved estimators **for large n**)

- asymptotically normally distributed

Problem: might need a lot of data ⇒ can overfit

Bayesianism

Prior $p(\theta)$ about data, likelihood $p(y|x)$ ⇒ posterior $p(\theta|y,x) = \frac{p(\theta)p(y|x,\theta)}{p(y|x)}$, where p(x) are cancelled out.
Maximum Aposteriori Estimate (MAP):

$$\arg\max_{\theta} p(\theta)\prod_{i=1}^{n} p(y_i|x_i,\theta)$$

for $\theta \sim \mathcal{N}(0,\sigma^2\mathcal{I})$ MAP yields ridge regr.

laplacian: $p(x;\mu,b) = \frac{1}{2b}exp(-\frac{|x-\mu|}{b})$

- Choose prior & likelihood fct. according to problem
- Optimize MAP parameters
- Choose hyperparameters through CV
- Make predictions using Bayesian decision

Bayesian Decision

$$C(y,a) = \begin{cases} [y \neq a], & \text{if } a \in [+1,-1] \\ c, & \text{if } a = D \end{cases}$$

With abstention:

$$a^* = \begin{cases} y, & \text{if } P(y|x) \geq 1-c \\ D, & \text{otherwise} \end{cases}$$

E.g. logistic loss in binary classification: Pick $y=+1$ if

$$p(y=-1|x) * C_{FP} < p(y=+1|x) * C_{FN} \Rightarrow$$
$$p(y=+1|x) \geq C_{FP}(C_{FP}+C_{FN})$$

Decision rule for bin. class.: (f is called Discriminative fct.)
$f(x) = log(\frac{p(y=1|x)}{p(y=-1|x)}) \Rightarrow f(x) > 0 \Rightarrow$ choose y=1
Active Learning: get labels in algorithm. Uncertainty sampling: Query labels of samples which the generative model is least sure about. i.i.d assumption is violated though

Conjugate Priors

Posterior $p(\theta|x,y) = \frac{p(\theta)p(\theta|y,x)}{\int p(\hat{\theta})p(y|x,\hat{\theta})d\theta}$ Can be infeasible ⇒
Conjugate priors (Hyperparameters with CV):

Prior/Posterior	Likelihood Fct
Beta	Bernoulli/Binom.
Dirichlet	Cathegorical/Multinom.
Gaussian, fixed cov.	Gaussian
Gaussian-inverse Wishart	Gaussian
Gaussian process	Gaussian

Generative Modelling

- Estimate prior on labels $p(y)$
- Estimate cond. distr. for each class $p(x|y)$ (e.g. how do the feature distributions look like)
- Joint distribution $p(y,x)$ through Bayes law

Alternative approach: $p(x)$, $p(x|y) \rightarrow p(x,y)$

Gaussian Bayes Classifier

- Consider classes y with $p_y = p(Y=y), p_y = \frac{\#Y=y}{n}$
- $P(\boldsymbol{x}|y) \sim \mathcal{N}(\boldsymbol{x};\mu_y,\sigma_y^2)$
  $\boldsymbol{\mu}_y = \frac{1}{\#y}\sum_{y_i=y}\boldsymbol{x}_i$, $\boldsymbol{\Sigma}_y = \frac{1}{\#y}\sum_{y_i=y}(\boldsymbol{x}_i-\boldsymbol{\mu}_y)^2$
  Aka $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ class wise
- $y = \underset{y'}{\text{argmax}} P(y'|x) = \underset{y'}{\text{argmax}} P(y')\prod_{i=1}^{d} P(x_i|y')$
  With i the ith coordinate of a feature x
- To predict: $P(y|x) = \frac{P(y)P(x|y)}{\sum_y P(y)P(x|y)} \Rightarrow$ bayes. decis.
- Fisher's LDA: $p=0.5$, $\Sigma_+ = \Sigma_-$. $c=2$
  LDA if $c \geq 2, p$ random, QDA = LDA with $\Sigma_i \neq \Sigma_j$
- Naive Bayes: $\Sigma_y = diag(\sigma_{y,1},..,\sigma_{y,d})$, $c \geq 2$
- GMMBC GMM instead of a single gaussian

GNB: Naive because it assumes independent samples (e.g. duplicate features would lead to overconfidence)
$f(x) = log\frac{p(Y=1|x)}{p(Y=-1|x)} \Rightarrow p(Y=1|x) = \frac{1}{1+exp(-f(x))}$
$= \sigma(\boldsymbol{\omega}^T x + \omega_0) \Rightarrow$ same as log. reg. with
$\omega_0 = log\frac{p_+}{1-p_+} + \sum_{i=1}^{d}\frac{\mu_{-,i}^2-\mu_{+,i}^2}{2\sigma_i^2}$, $\omega_i = \frac{\mu_{+,i}-\mu_{-,i}}{\sigma_i^2}$
Note: can introduce bias through choice of likelihood (e.g. GNB) to avoid overfitting
Also note that more clusters can fit the data better or as good (sup. no overfitting) while too few clusters is an issue.

Outlier Rejection

Can get $p(x) = \sum_{y_i} p(x|y_i)p(y_i) \Rightarrow$
mark x_i as outlier if $p(x_i) < \tau$

GANs

Discriminator working against Generator:

$$\min_{\omega_G}\max_{\omega_D} \underbrace{\mathbb{E}_{x\sim Data} log D(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z}\sim G} log(1-D(G(\boldsymbol{z})))}_{M(\omega_G,\omega_D)}$$

With $D(x) = D(x,\omega_D), G(z) = G(z,\omega_G)$

- Finds saddle point
- For unlimited data $D_G^* = \frac{P_{data}(x)}{P_G(x)+P_{data}(x)}$ Only conceptual as we can't know P_G and aren't given P_{data}

Common training approach: simultaneous GD:

$$\boldsymbol{\omega}_G^{(t+1)} = \boldsymbol{\omega}_G^{(t)} - \eta_t \nabla_{\boldsymbol{\omega}_G} M(\boldsymbol{\omega}_G, \boldsymbol{\omega}_D^{(t)})$$
$$\boldsymbol{\omega}_D^{(t+1)} = \boldsymbol{\omega}_D^{t} - \eta_t \nabla_{\boldsymbol{\omega}_D} M(\boldsymbol{\omega}_G^{(t)}, \boldsymbol{\omega}_D)$$

Usually using minibatches of data
Possible problems: Oscillation, Mode collapsedata memo. leading to degeneracy, how to evaluate a GAN

Performance metric: Duality gap:

$$DG(\omega_G,\omega_D) := \max_{\omega_D'} M(\omega_G,\omega_D') - \min_{\omega_G'} M(\omega_G',\omega_D)$$

$DG \geq 0, DG = 0$ if $\omega_G \& \omega_D$ perform perfect equilibrium.

Gaussian Mixture Model

$$P(\boldsymbol{x}|\theta) = P(\boldsymbol{x}|\mu,\Sigma,\pi) = \sum_{i=1}^{k}\pi_i\mathcal{N}(\boldsymbol{x};\mu_i,\Sigma_i)$$

GMMs are the same as GBC with unknown labels
MLE: $(\pi_{1:k}^*,\mu_{1:k}^*,\Sigma_{1:k}^*) = \underbrace{\text{argmin} -\sum_{i=1}^{n} log \sum_{j=1}^{k}\pi_j\mathcal{N}(\boldsymbol{x};\mu_j,\Sigma_j)}_{\text{Intractable}}$

Expectation Maximization (EM)

Hard EM:

- Initialize parameters θ
- E-step: Get most likely class for a datapoint
  $z_i^{(t)} = \underset{z}{\text{argmax}} P(z|\boldsymbol{x}_i,\theta^{(t-1)})$
  After E-step we have complete datasets ⇒ MLE
- M-step: MLE
  $\theta^{(t)} = \underset{\theta}{\text{argmax}} P(D^{(t)}|\theta)$
- ∞ clusters possible, many with $\sigma = 0$ or overlaps, doesn't consider certainties
- k-means is like special case with uniform weights and spherical covariance

(soft) EM algorithm:

- Param initialization
- Estimate affiliation prob. of each datapoint and each cluster
- MLE
- Repeat
- Applications include: Clustering, Density estimation, Classification and Outlier detection
- Monotonically increases likelihood
- Depends on init. (often multiple random init. runs)

GMMs can overfit (∞ clusters) can be fixed with prior or simpler model (Naive Bayes e.g.)

Additionals

Jensen's Inequality: $\mathbb{E}_D\left[\min_{f\in F}\hat{R}_D(f)\right] \leq \min_{f\in F}\mathbb{E}_D\left[\hat{R}_D(f)\right]$

Standardization

$x_{new} = \frac{x-\mu}{\sigma}$ - if one feature is bigger than others. Allows higher learning rates. Important for distance based methods: **knn,SVM,PCA,NNs,GD** Stdz always **after** train-test split.

Classification Metrics

Hypothesis test: Set hypothesis, reject it if $\hat{p}(x) > \tau$ and accept it if $\hat{p}(x) < \tau$
Reject hypothesis ⇒ positive — higher $\tau \Rightarrow$ more negatives

- acc.= $\frac{TP+TN}{n}$
- prec.= $\frac{TP}{TP+FP}$
- FPR= $\frac{FP}{FP+TN}$
- Recall / TPR= $\frac{TP}{TP+FN}$
- balanced acc.= $\frac{1}{n}\sum_i TPR_i$
- FDR= $\frac{FP}{TP+FP}$
- F1-score= $\underbrace{\frac{2TP}{2TP+FP+FN}}_{\frac{2}{\frac{1}{Recall}+\frac{1}{prec.}}}$
- ROC= $\frac{\text{FPR}}{\text{TPR}}$

Individual Additions