good luck =)

KNN

- erreur asymptotique est au + le double de Bayes
- Condensed NN
  - choisir 1 point par classe à mettre dans S
  - ∀ x de training ∉ S, essayer de le classifier avec un 1-NN sur S
  - si la prédiction est incorrecte, ajouter x à S
  - répéter tant qu'il y a eu au moins une insertion dans S lors du dernier parcours
- flanchement par des données à haute dimension car elles vivent dans un low dimensional manifold
- performance issues

KMEANS

- unsupervised learning
- K can be found using the elbow method
  - y : average within cluster sum of squares
  - x : the tested k
- the algorithm will converge (not necessarily to an optimal sol)

PCA

On cherche les eigenvectors de la covariance matrix de nos features.

$\max_{\vec{v}} \vec{v}^T C \vec{v}$ • $C = \frac{1}{n}\sum_i x_i x_i^T$

$Cv = -\lambda v$

① centrer les données ② $C = \frac{1}{n} X_{cent}^T X_{cent}$ ③ On trouve les vecteurs propres.

Les valeurs propres donnent la prop de la variance totale de chaque

LDA

KKT: $\nabla_x L(x,\lambda,\varphi) = 0$, $\lambda \geq 0$, $\lambda g(x) = 0$

matrice de dispersion intra-classe: $S_W = \sum_{c=1}^{C}\sum_{i \in c}(x_i - \mu_c)(x_i - \mu_c)^T$ | $\min_{w_1} E_W = w_1^T S_W w_1$

matrice de dispersion inter-classe: $S_B = \sum_{c=1}^{C} N_c (\mu_c - \mu)(\mu_c - \mu)^T$ | $\max_{w_1} E_B = w_1^T S_B w_1$

Normalization: $(x - x_{min}) / (x_{max} - x_{min})$

Standardization (ou normalization): $\frac{x-\mu}{\sigma}$

Pour chaque feature $j = 1 \dots D$: $\mu_j = \dots$, $\sigma_j = \sqrt{1/N \sum_{i=1}^{N} (x_{ij} - \mu_j)^2}$

Vanishing gradient – ReLU $= \max(0, x)$

Softmax $= \frac{e^{z_i}}{\sum_{j=1} e^{z_j}}$

Dropout: on masque, à chaque forward pass, certains neurones (on les désactive). Évite l'overfitting (on regarde TOUJOURS les mêmes features).

MSE: $\frac{1}{N}\sum_{i=1}^{N}(y_i - t_i)^2$

MAE: $1/N \sum |y_i - t_i|$

Linear regression – MSE closed form: $w = (X^T X)^{-1} X^T y$

Multiclass entropy: $E(w_1, \dots, w_K) = -\sum_n \sum_k t_n^k \ln(y^k(x_n))$

k-Fold Cross Validation

- k-1 ensembles pour train
- 1 ensemble de validation
- on répète k fois

Puis on fait pour chaque fold, la moyenne de l'accuracy, le f1 score, etc.

Distances

- L_1 Manhattan: $\sum |x_d - x'_d|$
- L_2 Euclidean → pénalise bcp les grandes variations: $\sqrt{\sum |x_d - x'_d|^2}$
- L_p: $\left(\sum |x_d - x'_d|^p\right)^{1/p}$

Auto-encoders

MLP qui prend en entrée des données et les mappe vers petite dimension. Décodeur essaye de retrouver les données d'origine.

Perceptron

Minimize $E(\vec{w}) = -\sum_{n=1}^{N} \text{sign}(\vec{w} \cdot \vec{x}_n)\, t_n$ (step function)

- center the x_ns so that $w_0 = 0$
- set $\vec{w}_1$ to 0
- iteratively, pick a random n: if x_n well classified, do nothing; otherwise, $\vec{w}_{t+1} = \vec{w}_t + t_n \vec{x}_n$

HP: epochs, learning rate, batch size

no guarantee that it works (no max margin); at most $\frac{R^2}{r^2}$ mistakes (marge → r^2, max data norm: $\max_i \|\vec{x}_i\|$)

Logistic Regression

- use $\sigma(z) = \frac{1}{1+e^{-z}}$ instead of step func ($\sigma(\vec{w}\cdot\vec{x})$ instead of $\text{sign}(\vec{w}\cdot\vec{x})$) → is differentiable
- minimize the cross-entropy loss (convex)

$E(\vec{w}) = -\sum_n \{t_n \ln(y_n) + (1-t_n)\ln(1-y_n)\}$

avec $y_n = \sigma(\vec{w}\cdot\vec{x}) \Rightarrow$ 0.5 si x est sur la frontière; 0 ou 1 si x est loin

$\nabla E(\vec{w}) = \sum_n (y_n - t_n)\vec{x}_n$

$\vec{w} = \vec{w} - \alpha \nabla E(\vec{w})$ (α: LR)

Multi-class
- on entraîne K classifiers $\sigma(w_k^T x)$

Max-Margin classifier (SVM)

$d_n = t_n \frac{(\vec{w}\cdot\vec{x}_n)}{\|w\|}$ distance to hyperplane

$w^* = \arg\max_w \min_n d_n$

(we scale w to λw until we have $\forall m\ t_m(\vec{w}\cdot\vec{x}_m) \geq 1$ for all points (with equality for at least one point) and $w^* = \arg\min_w \frac{1}{2}\|w\|^2$ (avec la contrainte)) pb quadratic / convex

Soft Margin:

$w^* = \arg\min_w \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$

subject to $y_i(\vec{w}\cdot\vec{x}_n) \geq 1 - \xi_n$

find C with cross-validation (grid search)

SVM Dual form (easier to solve)

Primal pb: $\min \frac{1}{2}\|w\|^2 + C\sum \xi_i$ devient dual pb:

$\max \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \vec{x}_i^T \vec{x}_j$

subject to $0 \leq \alpha_i \leq C$ and $\sum \alpha_i y_i = 0$

$\alpha_i = 0$ not a support vector / $0 < \alpha_i < C$ on margin

$\alpha_i = C$, point is missclassified or inside the margin

$\vec{w} = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i$ (only support vectors count)

Pb: pas lin sep. On remplace x par $\phi(x)$

SVM kernel

Au lieu de calculer $\Phi(x_i), \Phi(x_j), \Phi(x_i)\Phi(x_j)$, on a juste $k(x_i, x_j) = \Phi(x_i)\Phi(x_j)$

$\vec{y}(\vec{x}) = \text{sign}\left[\sum_{i \in SV} \alpha_i y_i k(\vec{x}_i, \vec{x})\right]$ (similarité x_i, x)

Common kernels:
- Linear: $k(\vec{x}, \vec{x}') = \vec{x}^T\vec{x}'$
- Poly: $k(\vec{x}, \vec{x}') = (\vec{x}^T\vec{x}' + c)^d$
- RBF: $\exp(-\gamma \|\vec{x} - \vec{x}'\|^2)$

kernel ridge regression

$\vec{y}(\vec{x}) = k(\vec{x})(K + \lambda I)^{-1}\vec{T}$ (T: labels; $k_{i,j} = k(\vec{x}_i, \vec{x}_j)$; λ: Regularization)

solution pour minimiser $E(\vec{w}) = \frac{1}{2}\sum \|W^T\phi(x_n) - t_n\|^2 + \frac{\lambda}{2}\|w\|^2$

(RBF Gaussian kernel: $\exp\left(-\frac{\|x - x'\|^2}{\sigma^2}\right)$)

Linear kernel: $x^T x'$

Poly. kernel: $(1 + x^T x')^d$