

Εργασία: Αναγνώριση Προτύπων & Μηχανική Μάθηση

Μέρη Α, Β, Γ, Δ

Ευάγγελος Μόσχου
ΑΕΜ: 10986

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Ιανουάριος 2026

Περιεχόμενα

- 1 Μέρος Α: Εκτίμηση Παραμέτρων με Μέγιστη Πιθανοφάνεια
- 2 Μέρος Β: Εκτίμηση Συνάρτησης Πυκνότητας με Παράθυρα Parzen
- 3 Μέρος Γ: k-Nearest Neighbors Classifier
- 4 Μέρος Δ: Υβριδικές Μέθοδοι Ensemble για Πινακοποιημένα Δεδομένα
- 5 Συνολικά Συμπεράσματα

Μέρος Α: Περιγραφή Προβλήματος

Στόχος

Εκτίμηση παραμέτρων τριών κανονικών κατανομών χρησιμοποιώντας την τεχνική της Μέγιστης Πιθανοφάνειας (Maximum Likelihood Estimation).

- **Σύνολο Δεδομένων:** dataset1.csv
- **Δείγματα:** 300 δείγματα (100 ανά κλάση)
- **Διαστάσεις:** 2 χαρακτηριστικά (features)
- **Κλάσεις:** 3 (0, 1, 2)

Περιορισμός

Υλοποίηση χωρίς χρήση έτοιμων συναρτήσεων βιβλιοθηκών για MLE.

Θεωρητικό Υπόβαθρο

Πολυδιάστατη Κανονική Κατανομή

Για κάθε κλάση c , η πυκνότητα πιθανότητας είναι:

$$p(\mathbf{x}|\mu_c, \Sigma_c) = \frac{1}{(2\pi)^{d/2} |\Sigma_c|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \mu_c)^T \Sigma_c^{-1} (\mathbf{x} - \mu_c)\right)$$

Εκτιμητές Μέγιστης Πιθανοφάνειας

Για N_c δείγματα της κλάσης c :

$$\hat{\mu}_c = \frac{1}{N_c} \sum_{i=1}^{N_c} \mathbf{x}_i^{(c)}$$

$$\hat{\Sigma}_c = \frac{1}{N_c} \sum_{i=1}^{N_c} (\mathbf{x}_i^{(c)} - \hat{\mu}_c)(\mathbf{x}_i^{(c)} - \hat{\mu}_c)^T$$

- **Βήμα 1:** Διαχωρισμός δεδομένων ανά κλάση
- **Βήμα 2:** Υπολογισμός μέσου όρου $\hat{\mu}_c$ για κάθε κλάση
- **Βήμα 3:** Υπολογισμός πίνακα συνδιακύμανσης $\hat{\Sigma}_c$
- **Βήμα 4:** Οπτικοποίηση με 3D plot

Τεχνικές Λεπτομέρειες

- Χρήση NumPy για πράξεις πινάκων
- Matplotlib για 3D visualization
- Meshgrid για δημιουργία επιφανειών

Αποτελέσματα - Εκτιμημένες Παράμετροι

Κλάση 0 (N=99):

$$\hat{\mu}_0 = \begin{pmatrix} 29.25 \\ 16.87 \end{pmatrix}, \quad \hat{\Sigma}_0 = \begin{pmatrix} 47.76 & 23.27 \\ 23.27 & 49.57 \end{pmatrix}$$

Κλάση 1 (N=100):

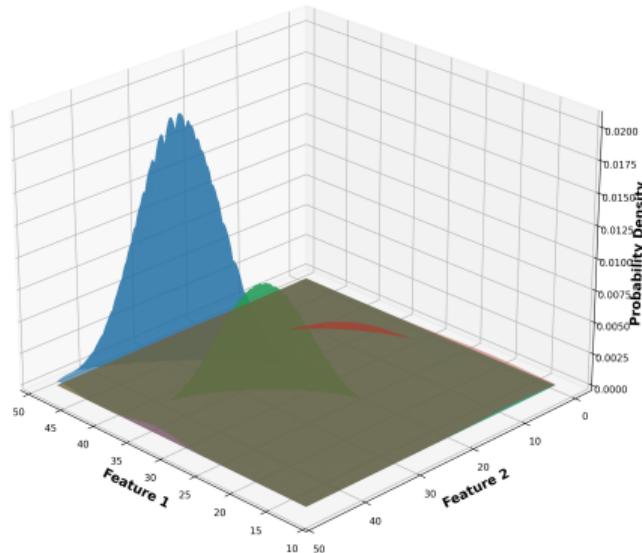
$$\hat{\mu}_1 = \begin{pmatrix} 40.20 \\ 34.28 \end{pmatrix}, \quad \hat{\Sigma}_1 = \begin{pmatrix} 9.52 & 11.61 \\ 11.61 & 20.31 \end{pmatrix}$$

Κλάση 2 (N=100):

$$\hat{\mu}_2 = \begin{pmatrix} 27.55 \\ 34.79 \end{pmatrix}, \quad \hat{\Sigma}_2 = \begin{pmatrix} 14.11 & 11.89 \\ 11.89 & 25.54 \end{pmatrix}$$

3D Οπτικοποίηση

3D Gaussian Distributions (MLE)



Οι τρεις εκτιμημένες κανονικές κατανομές οπτικοποιούνται ως τρισδιάστατες επιφάνειες πυκνότητας πιθανότητας.

Μέρος Β: Περιγραφή Προβλήματος

Στόχος

Εκτίμηση της συνάρτησης πυκνότητας πιθανότητας (PDF) χρησιμοποιώντας τη μέθοδο παραθύρων Parzen.

- **Σύνολο Δεδομένων:** dataset2.csv
- **Δείγματα:** 200 μονοδιάστατα
- **Υπόθεση:** Δεδομένα από $\mathcal{N}(1, 4)$
- **Kernels:** Υπερκύβος και Gaussian

Ζητούμενο

Εύρεση της βέλτιστης τιμής h (bandwidth) για κάθε kernel.

Εκτιμητής Parzen

$$\hat{p}(x) = \frac{1}{n \cdot h} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)$$

όπου K η kernel function και h το bandwidth.

Υπερκύβος (Hypercube):

$$K(u) = \begin{cases} 1 & |u| \leq \frac{1}{2} \\ 0 & \text{αλλιώς} \end{cases}$$

Gaussian:

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}}$$

Μεθοδολογία Βελτιστοποίησης

① **Εύρος** h : [0.1, 10] με βήμα 0.1

② **Για κάθε** h :

- Υπολογισμός προβλεπόμενης πιθανοφάνειας $\hat{p}(x_i)$
- Υπολογισμός πραγματικής πιθανοφάνειας από $\mathcal{N}(1, 4)$
- Υπολογισμός MSE: $MSE = \frac{1}{n} \sum_{i=1}^n (p(x_i) - \hat{p}(x_i))^2$

③ **Επιλογή:** $h^* = \arg \min_h MSE(h)$

Αποτελέσματα - Βέλτιστο Bandwidth

Hypercube Kernel

- $h_{hypercube}^* = 2.80$
- MSE = 1.091×10^{-3}

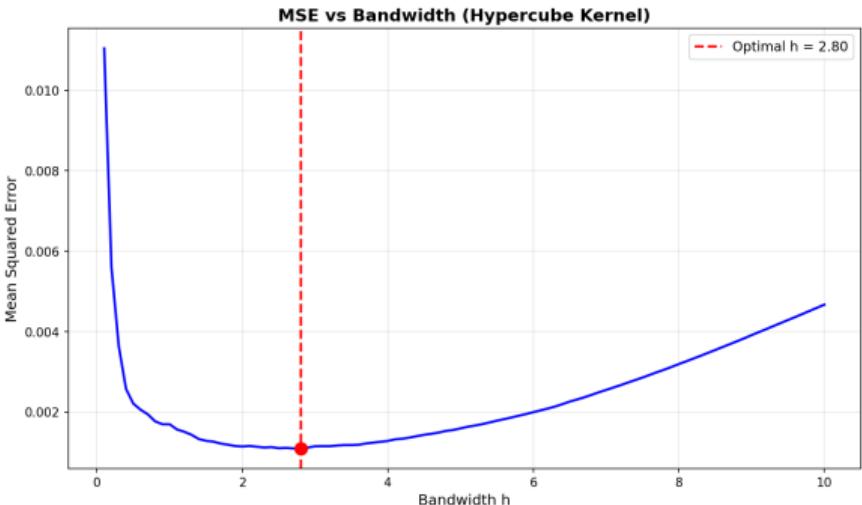
Gaussian Kernel

- $h_{gaussian}^* = 0.80$
- MSE = 1.131×10^{-3}

Σύγκριση

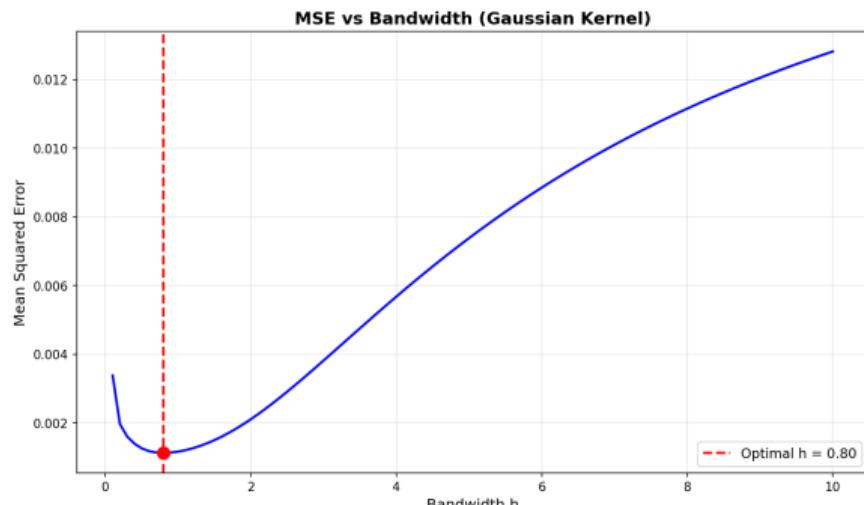
Το Hypercube kernel έχει λίγο μικρότερο MSE αλλά το Gaussian παρέχει ομαλότερη εκτίμηση.

Plots: Σφάλμα vs Bandwidth



Hypercube Kernel

Η βέλτιστη τιμή h είναι στο ελάχιστο σημείο MSE (κόκκινη διακεκομμένη γραμμή).



Gaussian Kernel

Μέρος Γ: Περιγραφή Προβλήματος

Στόχος

Υλοποίηση k-Nearest Neighbors (KNN) classifier από την αρχή.

- **Training Set:** dataset3.csv (50 δείγματα, 2D)
- **Test Set:** testset.csv (50 δείγματα, 2D)
- **Κλάσεις:** 2 (0, 1)
- **Εύρος k:** [1, 30]

Υλοποίηση

Χωρίς χρήση έτοιμων βιβλιοθηκών KNN (π.χ., sklearn.neighbors).

Αλγόριθμος KNN

Για ένα test δείγμα \mathbf{x} :

- ① Υπολογισμός απόστασης από όλα τα training δείγματα
- ② Επιλογή των k πλησιέστερων γειτόνων
- ③ Υπολογισμός πιθανότητας ανά κλάση: $P(y = c|\mathbf{x}) = \frac{\text{count}(c)}{k}$
- ④ Πρόβλεψη: $\hat{y} = \arg \max_c P(y = c|\mathbf{x})$

Ευκλείδεια Απόσταση

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{d=1}^D (x_{i,d} - x_{j,d})^2}$$

Υλοποίηση - Συναρτήσεις

① **eucl(x, trainData):**

- Υπολογίζει ευκλείδεια απόσταση από όλα τα training δείγματα
- Επιστρέφει διάνυσμα αποστάσεων

② **neighbors(x, trainData, k):**

- Καλεί eucl() για υπολογισμό αποστάσεων
- Ταξινομεί κατά αύξουσα σειρά
- Επιστρέφει τα k κορυφαία σημεία

③ **predict(testData, trainData, k):**

- Καλεί neighbors() για κάθε test δείγμα
- Υπολογίζει πιθανότητες κλάσεων
- Επιστρέφει πίνακα πιθανοτήτων

Βελτιστοποίηση k

- Δοκιμή όλων των τιμών $k \in [1, 30]$
- Υπολογισμός accuracy για κάθε k :

$$\text{Accuracy} = \frac{\text{Σωστές Προβλέψεις}}{\text{Σύνολο Test Δειγμάτων}}$$

- Επιλογή k^* με το μέγιστο accuracy

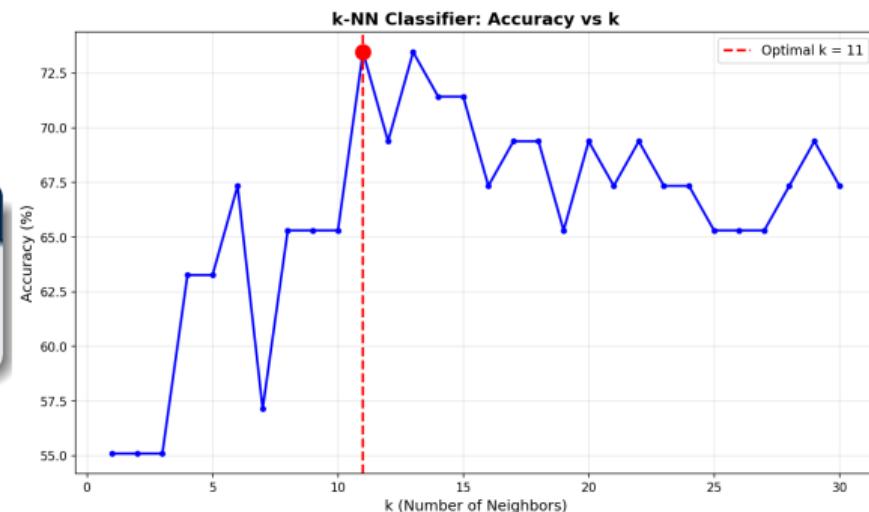
Trade-off

- Μικρό k : Ευαίσθητο σε θόρυβο
- Μεγάλο k : Over-smoothing, απώλεια δομής

Αποτελέσματα

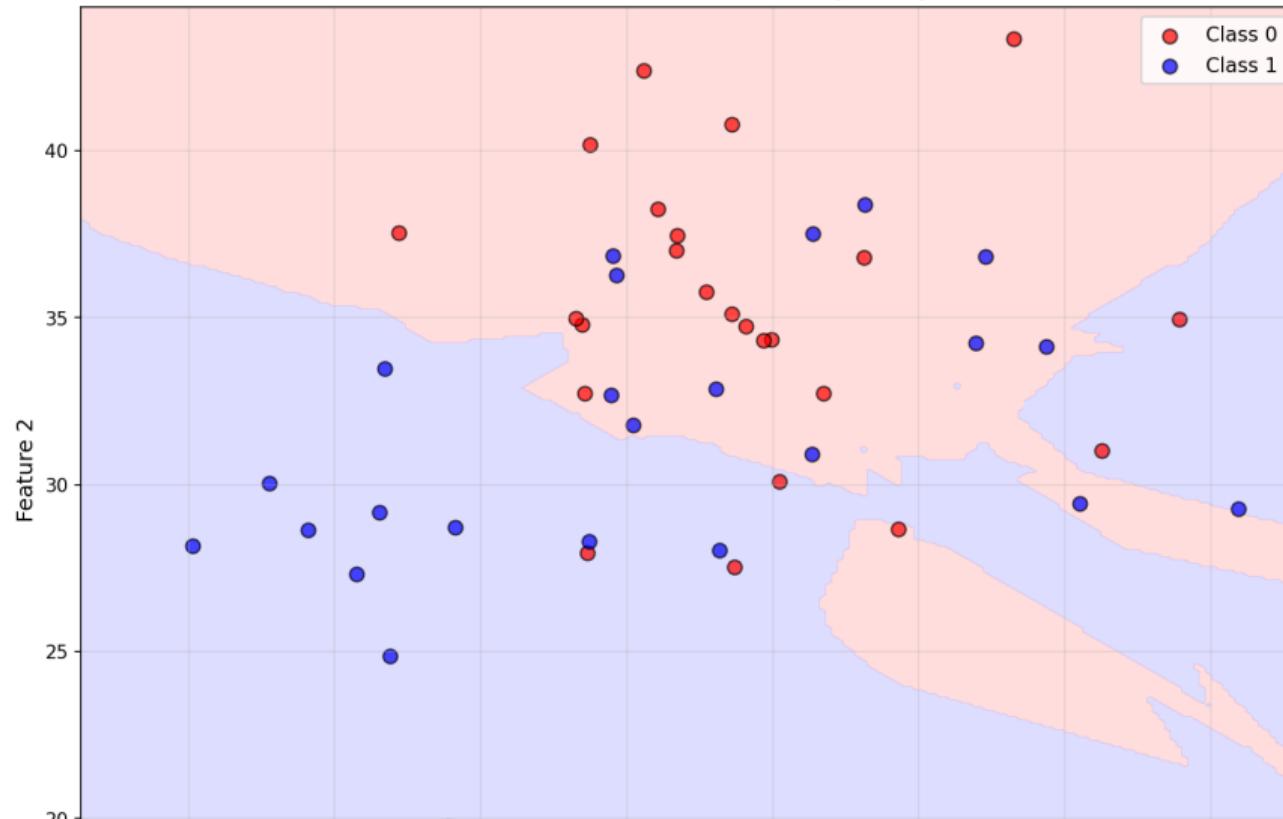
Βέλτιστο k

- $k^* = 11$
- Accuracy = 73.47%



Decision Boundaries

k-NN Decision Boundaries (k=11)



Μέρος Δ: Επισκόπηση

Στόχος

Ταξινόμηση 8743 training samples (224 features) σε 5 κλάσεις, πρόβλεψη για 6955 test samples.

Μεθοδολογία:

- Υβριδική αρχιτεκτονική (Gradient Boosting + Neural Networks)
- Προηγμένη μηχανική χαρακτηριστικών
- Stochastic ensemble με πολλαπλά seeds
- Τεχνικές αιχμής (DART, SAM, Langevin)

Βασική Ιδέα

Συνδυασμός πολλαπλών ορθογώνιων τεχνικών για μέγιστη γενίκευση

Pipeline

- ① **Raw Data:** 224 features, 8743 samples
- ② **Preprocessing:** Quantile transformation, feature selection
- ③ **Feature Engineering:** LID, PageRank, Transductive DAE
- ④ **Ensemble Models:** XGBoost DART, CatBoost Langevin, TabR, TabPFN
- ⑤ **Calibration:** Isotonic regression, LID temperature scaling
- ⑥ **Final Predictions:** NNLS Stacking (5 seeds)

Μηχανική Χαρακτηριστικών (Feature Engineering)

1. Quantile Transformation:

- $x' = \Phi^{-1}(F(x))$
- Robust σε outliers

2. Feature Selection (Razor):

- Per-model masks (Cat: 179, XGB: 180)
- Μείωση θορύβου

3. Manifold Engineering:

- LID, PageRank στον KNN γράφο

4. Transductive DAE:

- Εκπαίδευση σε train+test
- Πλούσια embeddings χωρίς label leakage

Μοντέλα Ensemble (1/3)

1. XGBoost με DART

Dropouts meet Additive Regression Trees

- Random dropout δέντρων κατά boosting
- Αποφυγή over-specialization
- rate_drop=0.1, skip_drop=0.5

2. CatBoost με Langevin Dynamics

Στοχαστική Βελτιστοποίηση

$$\theta_{t+1} = \theta_t - \eta \nabla L + \sqrt{2\eta T} \epsilon_t$$

- Θερμικός θόρυβος για εξερεύνηση
- Διαφυγή από τοπικά ελάχιστα
- diffusion_temperature=1000

3. TabR (Attention-Based Retrieval)

PyTorch Neural Architecture

- ① Encoder: MLP για embedding
- ② Retrieval: k-NN στον embedding space
- ③ Cross-Attention: Query-Key-Value
- ④ Classification head

Βασικές τεχνικές:

- Topology-Aware MixUp
- Batch size: 512 (RTX 3060 constraint)
- Learning rate: 2e-3

4. TabPFN v2.5 (Prior-Data Fitted Network)

In-Context Learning για Tabular Data

- Pre-trained transformer σε synthetic datasets (v2.5)
- Zero-shot inference (χωρίς training στα δικά μας δεδομένα)
- Περιορισμός: max 10000 samples, 500 features (v2.5)
- Παρέχει ανεξάρτητη οπτική (ορθογωνικό μοντέλο)

Στρατηγική Ensemble: NNLS Stacking

Συνδυασμός με Non-Negative Least Squares (όχι απλό averaging). Εύρεση βέλτιστων βαρών $w_i \geq 0$.

Βελτιστοποίηση & Κανονικοποίηση (Regularization)

Monte Carlo Ensemble:

- 5 τυχαία seeds
- Μείωση διακύμανσης: $\text{Var}/\sqrt{5}$
- Robust predictions

10-Fold Cross-Validation:

- Stratified splits
- 90% train, 10% validation
- Τελικό: Μέσος όρος 10 models

Isotonic Calibration:

- Βαθμονόμηση πιθανοτήτων
- Μονότονη παλινδρόμηση
- Καλύτερη αξιοπιστία

Topology MixUp:

- MixUp μόνο με k-NN
- Διατήρηση manifold
- Data augmentation

Cross-Fit Stacking

Meta-learning με OOF πιθανότητες

- ① Κάθε base model παράγει out-of-fold predictions
- ② Meta-learner εκπαιδεύεται στα OOF embeddings
- ③ Επιλογές: Logistic Regression, LightGBM, Mixture-of-Experts

Πλεονέκτημα

To stacking μαθαίνει βέλτιστα βάρη για κάθε μοντέλο, καλύτερα από απλό averaging.

Iterative Pseudo-Labeling

Transductive Learning

- ① Πρόβλεψη με high confidence στο test set
- ② Επιλογή samples με:
 - Confidence > κατώφλι
 - Συμφωνία μεταξύ seeds/views
- ③ Επανεκπαίδευση με pseudo-labels
- ④ Επανάληψη για N iterations

Προσοχή: Transductive Mode

Απαιτεί ALLOW_TRANSDUCTIVE=1. Αυτό επιτρέπει τη χρήση test data για unsupervised τεχνικές (π.χ. DAE, manifold). Κίνδυνος: το visible test set μπορεί να διαφέρει από το hidden test set, οδηγώντας σε overfitting.

CORAL (Covariance Alignment)

Μείωση Covariate Shift

Στόχος: Ευθυγράμμιση των covariance matrices train/test:

$$\min_W \|\mathbf{C}_{\text{test}} - W\mathbf{C}_{\text{train}}W^T\|_F^2$$

Εφαρμογή:

- Μετά από κάθε feature transformation
- Regularization: $\lambda = 10^{-3}$
- Βελτιωμένη adaptability

Στρατηγική Ensemble

- ① **Baseline:** CatBoost με Langevin dynamics
- ② **+DART:** XGBoost με dropout για regularization
- ③ **+TabR:** Attention-based retrieval augmentation
- ④ **+SAM:** Sharpness-aware optimization για flat minima
- ⑤ **+Monte Carlo:** Πολλαπλά seeds για variance reduction

Φιλοσοφία

Κάθε τεχνική στοχεύει διαφορετική πηγή σφάλματος (bias, variance, distribution shift).

Feature Selection (Razor)

Μεθοδολογία - Per-Model Masks

- ① Εκπαίδευση CatBoost & XGBoost scout models (5-fold CV)
- ② Υπολογισμός CV-averaged feature importance ανά μοντέλο
- ③ Δημιουργία ξεχωριστών masks: CatBoost → 179, XGBoost → 180 features

Εξαιρέσεις

- **TabR:** Χρησιμοποιεί Quantile view (όχι Razor mask)
- **TabPFN:** Χρησιμοποιεί raw data (bypasses Razor)

Hyperparameter Configuration

Επιλεγμένες Ρυθμίσεις

- **Batch Size:** 512 (βέλτιστο για σταθερότητα σε RTX 3060)
- **Learning Rate:** 2e-3 με linear scaling
- **SAM ρ :** 0.08 για balance exploration/exploitation
- **Seeds:** 5 για Monte Carlo averaging
- **CV Folds:** 10 για robust validation

Hardware Constraint

Βέλτιστη ρύθμιση για RTX 3060 (6GB VRAM): Batch 512 για σταθερότητα, αφήνοντας περιθώριο για TabPFN/TabR.

Συνολική Loss Function

$$\mathcal{L}_{\text{total}} = \underbrace{\mathcal{L}_{\text{CE}}}_{\text{Cross-Entropy}} + \lambda_1 \underbrace{\mathcal{L}_{\text{SAM}}}_{\text{Sharpness}} + \lambda_2 \underbrace{\mathcal{L}_{\text{MixUp}}}_{\text{Consistency}} + \lambda_3 \underbrace{\mathcal{L}_{\text{DAE}}}_{\text{Reconstruction}}$$

- $\mathcal{L}_{\text{CE}} = -\sum_c y_c \log \hat{y}_c$ (με label smoothing $\epsilon = 0.1$)
- \mathcal{L}_{SAM} : Perturbation $\epsilon = \rho \frac{\nabla L}{\|\nabla L\|}$, $\rho = 0.08$
- $\mathcal{L}_{\text{DAE}} = \|x - \text{Dec}(\text{Enc}(x + \text{noise}))\|^2$

NNLS Stacking & LID Temperature Scaling

$$P_{\text{ens}}(y|x) = \text{softmax} \left(\frac{1}{T(x)} \sum_{m=1}^M w_m \cdot \log P_m(y|x) \right), \quad w_m \geq 0$$

όπου $T(x) = 1 + \alpha \cdot \text{LID}(x)$ (υψηλό LID \rightarrow υψηλή θερμοκρασία \rightarrow μειωμένη εμπιστοσύνη)

Περιορισμοί

① Υπολογιστικό Κόστος:

- $5 \text{ seeds} \times 10 \text{ folds} = 50 \text{ model trainings}$
- Συνολικός χρόνος: 2-3 ώρες (RTX 3060)

② Μνήμη:

- Απαιτεί 6GB VRAM
- Δεν κλιμακώνεται εύκολα σε $>1\text{M samples}$

③ Hyperparameter Sensitivity:

- SAM ρ , Langevin temperature
- Topology MixUp k

Μελλοντικές Επεκτάσεις

Υλοποιημένα:

- ✓ Bayesian Optimization (Optuna με TPE sampler)
- ✓ Uncertainty Quantification (Isotonic + LID Scaling, 0.1% gain)

Μελλοντικά:

- Neural Architecture Search (NAS)
- Conformal Prediction

Scalability:

- Federated Learning
- Continual Learning
- Distributed Training
- Model Compression

Σημείωση: LLM Feature Extraction δεν εφαρμόζεται σε ανωνυμοποιημένα αριθμητικά δεδομένα.

Αποτυχημένα Μοντέλα

Μοντέλο	Λόγος Απόρριψης
GANDALF	Neural Decision Tree - υψηλό ρίσκο, ισοδύναμο XGBoost
KAN	Αργό & ασταθές, θεωρητική υπόσχεση χωρίς πρακτικό όφελος
TabM	Υστερεί σε diversity έναντι BatchEnsemble variant
Hopular	Redundant με TabR (και τα δύο memory-based)
CARTE	Απαιτεί σημασιολογικά features (strings)
Mambular	Δύσκολη εγκατάσταση CUDA kernels (*πιθανώς αποτελεσματικό)
DeepInsight	Tabular → Images: αργό, redundant με Manifold features
AutoGluon	Generic AutoML, δεν υποστηρίζει TTT/Pseudo-labeling

Σημείωση

Η απόρριψη αφορά την **transductive φιλοσοφία** του Τελικού Μοντέλου. Τα μοντέλα μπορεί να είναι αποτελεσματικά σε άλλα contexts.

Συμπεράσματα Μέρους Δ

Επιτεύγματα

- Υβριδική αρχιτεκτονική (Trees + Neural Networks)
- Advanced feature engineering & manifold analysis
- State-of-the-art optimization techniques
- Πολλαπλές τεχνικές γενίκευσης (Monte Carlo, βαθμονόμηση)

Κλειδί Επιτυχίας

Ο συνδυασμός πολλαπλών ορθογώνιων τεχνικών:

- Stochastic optimization (DART, Langevin, SAM)
- Topology awareness (MixUp, LID, PageRank)
- Variance reduction (Monte Carlo, Calibration)

Συνολικά Συμπεράσματα

Μέρος Α: Maximum Likelihood

Επιτυχής εκτίμηση παραμέτρων 3 κανονικών κατανομών.

Μέρος Β: Parzen Windows

Εύρεση βέλτιστου bandwidth για 2 kernels, validation με MSE.

Μέρος Γ: k-NN

Υλοποίηση KNN από την αρχή, βελτιστοποίηση k , visualization.

Μέρος Δ: Τελικό Μοντέλο

Υβριδική αρχιτεκτονική ensemble με προηγμένες τεχνικές βελτιστοποίησης.

Ευχαριστώ για την προσοχή σας!

Ερωτήσεις: