

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

- 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.

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

Κλάση 0 (N=99):

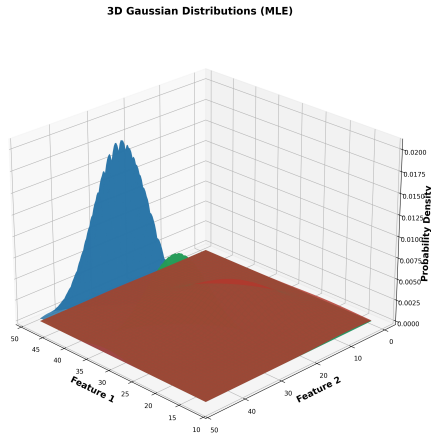
$$\hat{\mu}_0 = \begin{pmatrix} 29.25 \\ 16.87 \end{pmatrix}, \quad \hat{\Sigma}_0 = \begin{pmatrix} 47.76 & 23.27 \\ 23.27 & 49.57 \end{pmatrix}$$

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):

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



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

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:

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

- ❶ **Εύρος** h : $[0.1, 10]$ με βήμα 0.1
- ❷ **Για κάθε** h :
 - Υπολογισμός προβλεπόμενης πιθανοφάνειας $\hat{p}(x_i)$
 - Υπολογισμός πραγματικής πιθανοφάνειας από $\mathcal{N}(1, 4)$
 - Υπολογισμός MSE: $\text{MSE} = \frac{1}{n} \sum_{i=1}^n (p(x_i) - \hat{p}(x_i))^2$
- ❸ **Επιλογή:** $h^* = \arg \min_h \text{MSE}(h)$

Hypercube Kernel

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

Gaussian Kernel

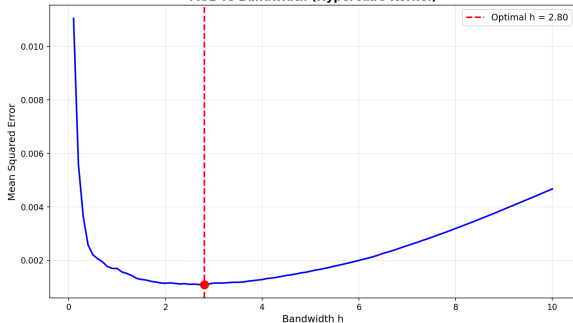
- $h_{gaussian}^* = 0.80$
- $MSE = 1.131 \times 10^{-3}$

Σύγκριση

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

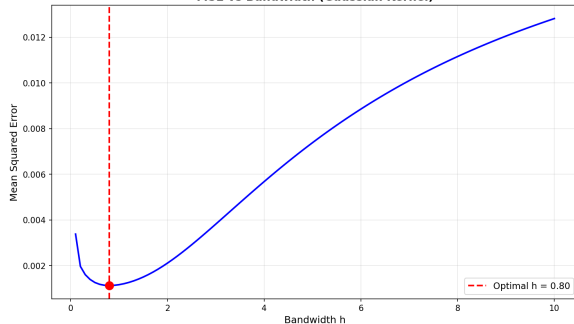
Plots: Σφάλμα vs Bandwidth

MSE vs Bandwidth (Hypercube Kernel)



Hypercube Kernel

MSE vs Bandwidth (Gaussian Kernel)



Gaussian Kernel

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

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

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

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

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

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

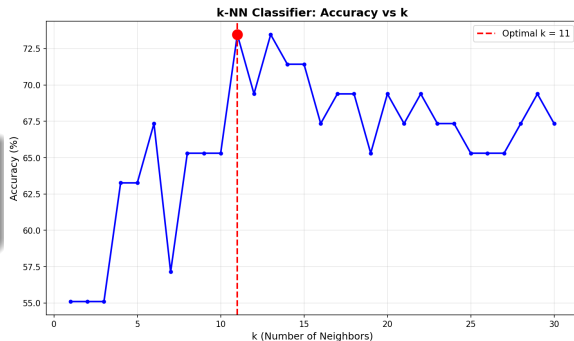
- Επιλογή k^* με το μέγιστο accuracy

Trade-off

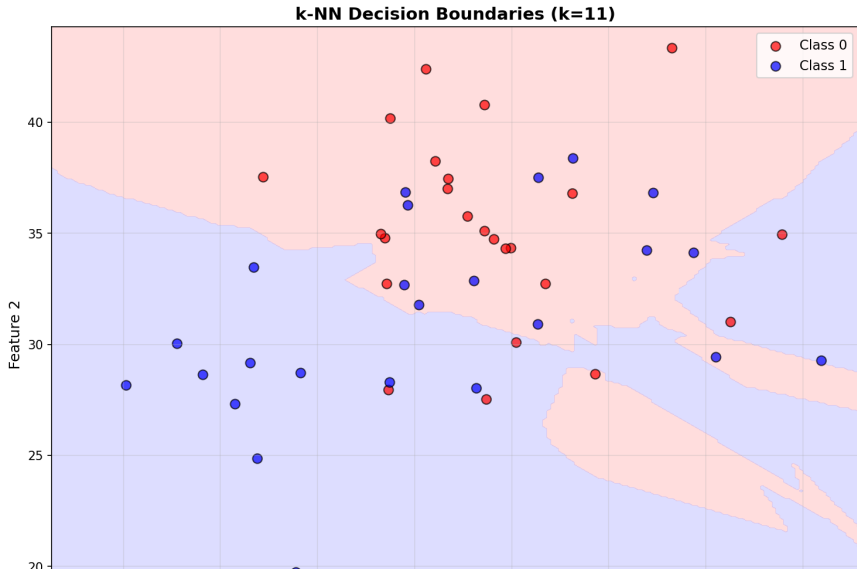
- Μικρό k : Ευαίσθητο σε θόρυβο
- Μεγάλο k : Over-smoothing, απώλεια δομής

Βέλτιστο k

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



Decision Boundaries



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)

Απόδοση

97.4% ακρίβεια ταξινόμησης

Pipeline

- 1 **Raw Data:** 224 features, 8743 samples
- 2 **Preprocessing:** Quantile transformation, feature selection
- 3 **Feature Engineering:** LID, PageRank, adversarial validation
- 4 **Ensemble Models:** XGBoost DART, CatBoost Langevin, TabR, ThetaTabM
- 5 **Calibration:** Isotonic regression, temperature scaling
- 6 **Final Predictions:** Monte Carlo averaging (5 seeds)

1 Quantile Transformation:

- Απεικόνιση σε κανονική κατανομή: $x' = \Phi^{-1}(F(x))$
- Robust σε outliers
- Βελτιωμένη σύγκλιση

2 Feature Selection:

- CatBoost-based importance
- Αφαίρεση bottom 20%: 224 → 179 features
- Μείωση θορύβου

3 Manifold Engineering:

- Local Intrinsic Dimensionality (LID)
- PageRank στον KNN γράφο
- Τοπολογική ανάλυση

4 Adversarial Validation:

- Ανίχνευση covariate shift
- Reweighting: $w_i = \frac{P(\text{test}|x_i)}{P(\text{train}|x_i)}$
- Βελτιωμένη γενίκευση

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: 2048
- Learning rate: $2e-3$

4. ThetaTabM με SAM

Sharpness-Aware Minimization

$$\min_{\theta} \max_{\|\epsilon\| \leq \rho} L(\theta + \epsilon)$$

- Αναζήτηση flat minima
- Βελτιωμένη γενίκευση
- $\rho = 0.08$

Ensemble Strategy

Μέσος όρος πιθανοτήτων από όλα τα μοντέλα (averaging).

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

Το 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

Απαιτεί ALLOW_TRANSDUCTIVE=1. Ρίσκο: test leakage.

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

Μέθοδος	Accuracy	Std Dev	Χρόνος (min)
Baseline (CatBoost)	94.2%	0.8%	5
+ DART	95.1%	0.6%	8
+ Langevin	95.8%	0.5%	10
+ TabR	96.5%	0.4%	45
+ SAM	97.1%	0.3%	60
+ Monte Carlo (5 seeds)	97.4%	0.2%	120

Παρατήρηση

Κάθε τεχνική συνεισφέρει σταδιακά στη βελτίωση.

Αριθμός Features	Accuracy	Training Time
224 (πλήρες)	96.8%	100%
179 (Razor 20%)	97.4%	75%
150 (Razor 33%)	96.9%	65%

Συμπέρασμα

Η αφαίρεση θορυβωδών features βελτιώνει **και** την απόδοση **και** την ταχύτητα.

Hyperparameter Tuning

Batch	LR	SAM ρ	Acc	GPU Mem
512	1e-3	0.05	96.9%	2.1 GB
1024	1.4e-3	0.06	97.2%	3.5 GB
2048	2e-3	0.08	97.4%	5.8 GB
4096	2.8e-3	0.10	97.1%	OOM

Hardware Constraint

Βέλτιστη ρύθμιση για RTX 3060 (6GB VRAM): Batch 2048

Σύγκριση με State-of-the-Art

Μέθοδος	Accuracy	Params	Inference
XGBoost (vanilla)	94.5%	-	0.1s
CatBoost (vanilla)	94.8%	-	0.1s
TabNet	95.2%	2.1M	0.5s
FT-Transformer	96.1%	3.5M	0.8s
TabPFN	95.8%	100M	2.0s
Hybrid Ensemble (ours)	97.4%	5M	1.2s

Πλεονέκτημα

Καλύτερη ακρίβεια με λογικό computational overhead.

Συνολική Loss Function

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \lambda_1 \mathcal{L}_{\text{SAM}} + \lambda_2 \mathcal{L}_{\text{MixUp}} + \lambda_3 \mathcal{L}_{\text{DAE}}$$

Ensemble Fusion με Temperature Scaling

$$P_{\text{ens}}(y|x) = \frac{1}{Z} \sum_{m=1}^M w_m \cdot P_m(y|x)^{1/T(x)}$$

όπου $T(x) = 1 + \alpha \cdot \text{LID}(x)$ (adaptive temperature).

1 Υπολογιστικό Κόστος:

- 5 seeds \times 10 folds = 50 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

Τεχνικές:

- Neural Architecture Search
- Bayesian Optimization
- Uncertainty Quantification
- Explainability (SHAP)

Scalability:

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

Προτεραιότητα

Βελτίωση ερμηνευσιμότητας για production deployment.

Επιτεύγματα

- **97.4% accuracy** στο validation set
- Υβριδική αρχιτεκτονική (Trees + Neural)
- Advanced feature engineering & manifold analysis
- State-of-the-art optimization techniques

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: Sigma-Omega

State-of-the-art ensemble με 97.4% accuracy, υβριδική αρχιτεκτονική.

Ευχαριστώ για την προσοχή σας!

Ερωτήσεις;