

Detailed Mathematical Presentation of Probabilistic Neural Networks (PNNs)

1. Introduction

A **Probabilistic Neural Network (PNN)** is a feedforward neural network designed primarily for classification and pattern recognition tasks. Unlike traditional neural networks that produce deterministic outputs, PNNs estimate the **probability density function (PDF)** of each class and use Bayesian decision theory to classify new inputs with minimum misclassification probability.

PNNs were introduced by D.F. Specht in 1966 and are closely related to kernel-based methods like Parzen windows and Bayesian classifiers.

2. Core Concepts and Architecture

PNNs approximate the **class-conditional probability density functions** using kernel density estimation and apply Bayes' rule to compute posterior probabilities for classification.

2.1 Bayes' Decision Rule

Given an input vector $\mathbf{x} \in \mathbb{R}^d$ and classes $\{C_1, C_2, \dots, C_K\}$, the posterior probability of class C_k is:

$$P(C_k | \mathbf{x}) = \frac{p(\mathbf{x} | C_k)P(C_k)}{p(\mathbf{x})}$$

where:

- $p(\mathbf{x} | C_k)$ is the class-conditional PDF.
- $P(C_k)$ is the prior probability of class C_k .
- $p(\mathbf{x}) = \sum_{j=1}^K p(\mathbf{x} | C_j) P(C_j)$ is the total probability of $\mathbf{x}$.

The classifier assigns $\mathbf{x}$ to the class with the highest posterior:

$$\hat{C} = \arg \max_k P(C_k | \mathbf{x})$$

2.2 Kernel Density Estimation (Parzen Windows)

PNNs estimate $p(\mathbf{x} | C_k)$ non-parametrically using Parzen window estimators:

$$p(\mathbf{x} | C_k) = \frac{1}{N_k} \sum_{i=1}^{N_k} K_{\sigma}(\mathbf{x} - \mathbf{x}_{k,i})$$

where:

- N_k is the number of training samples in class C_k .
- $\mathbf{x}_{k,i}$ is the i -th training sample from class C_k .
- $K_{\sigma}(\cdot)$ is a kernel function with bandwidth parameter σ , typically a Gaussian kernel:

$$K_{\sigma}(\mathbf{u}) = \frac{1}{(2\pi)^{d/2} \sigma^d} \exp\left(-\frac{\|\mathbf{u}\|^2}{2\sigma^2}\right)$$

3. PNN Network Structure

PNNs are organized into four layers:

1. **Input Layer:** Passes input vector $\mathbf{x}$ to the next layer.
2. **Pattern Layer:** Contains one neuron per training sample. Each neuron computes the kernel function output:

$$\phi_{k,i}(\mathbf{x}) = K_{\sigma}(\mathbf{x} - \mathbf{x}_{k,i})$$

3. **Summation Layer:** Sums the outputs of pattern neurons belonging to the same class:

$$S_k(\mathbf{x}) = \frac{1}{N_k} \sum_{i=1}^{N_k} \phi_{k,i}(\mathbf{x})$$

which estimates $p(\mathbf{x} | C_k)$.

4. **Output Layer:** Applies Bayes' rule by multiplying by prior $P(C_k)$ and selects the class with the maximum posterior probability:

$$\hat{C} = \arg \max_k (S_k(\mathbf{x})P(C_k))$$

4. Mathematical Operation of PNN

For each input $\mathbf{x}$:

- Compute kernel responses for all training samples:

$$\phi_{k,i}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_{k,i}\|^2}{2\sigma^2}\right)$$

- Aggregate per class:

$$S_k(\mathbf{x}) = \frac{1}{N_k} \sum_{i=1}^{N_k} \phi_{k,i}(\mathbf{x})$$

- Incorporate priors and decide:

$$\hat{C} = \arg \max_k (S_k(\mathbf{x})P(C_k))$$

5. Training and Parameters

- **Training:** PNNs require no iterative training of weights; the training set itself forms the pattern layer neurons.
- **Smoothing Parameter σ :** Controls kernel width; critical for bias-variance tradeoff.
- **Prior Probabilities $P(C_k)$:** Can be estimated from class frequencies or set uniformly.

6. Advantages and Properties

- Approaches **Bayes optimal classifier** under certain conditions.
- Fast training since no weight optimization is needed.

- Naturally models **probabilistic outputs**, useful for uncertainty estimation.
- Robust to noisy data given appropriate kernel bandwidth.

7. Extensions: Probabilistic Neural Networks for Regression and Aleatoric Uncertainty

Modern variants extend PNNs to regression by modeling output as a **probabilistic distribution** (e.g., Gaussian) with input-dependent mean and variance:

$$p(y \mid \mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$$

where both $\mu(\mathbf{x})$ and $\sigma^2(\mathbf{x})$ are predicted by the network, and training minimizes the **negative log-likelihood (NLL)**:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \left(\frac{(y_i - \mu(\mathbf{x}_i))^2}{2\sigma^2(\mathbf{x}_i)} + \frac{1}{2} \log \sigma^2(\mathbf{x}_i) \right)$$

This allows modeling **aleatoric uncertainty** (data noise) explicitly.

8. Summary of Key Equations

Concept	Equation
Bayes' posterior	$P(C_k \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_k) P(C_k)}{\sum_j p(\mathbf{x} \mid C_j) P(C_j)}$
Kernel density estimate	$p(\mathbf{x} \mid C_k) = \frac{1}{N_k} \sum_{i=1}^{N_k} K_{\sigma}(\mathbf{x} - \mathbf{x}_{k,i})$
Gaussian kernel	$K_{\sigma}(\mathbf{u}) = \frac{1}{(2\pi)^{d/2} \sigma^d} \exp\left(-\frac{\ \mathbf{u}\ ^2}{2\sigma^2}\right)$
Class decision rule	$\hat{C} = \arg\max_k \left(S_k(\mathbf{x}) P(C_k) \right)$
Negative log-likelihood loss (regression)	$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \left(\frac{(y_i - \mu(\mathbf{x}_i))^2}{2\sigma^2(\mathbf{x}_i)} + \frac{1}{2} \log \sigma^2(\mathbf{x}_i) \right)$

9. References

- Specht, D.F., “Probabilistic Neural Networks,” Neural Networks, 1990.
- Wikipedia: [Probabilistic Neural Network](#)
- MATLAB Documentation: [Probabilistic Neural Networks](#)
- Recent research on PNNs modeling aleatoric uncertainty (arXiv 2024).

*
**