TP # 4: Chemical Property Classification Using Advanced Perceptron Scheme

Dr. Samir Kenouche

Contents

1 Introduction

In computational chemistry and chemoinformatics, molecules can be represented by vectors of numerical descriptors (e.g., dipole moment, LogP, molecular weight, number of hydrogen bond donors). A **perceptron** is one of the simplest and most interpretable machine learning algorithms that can classify molecules according to physicochemical properties such as polarity, solubility, or toxicity.

Given molecular descriptors $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, the perceptron predicts whether a compound possesses a specific property (e.g., polar vs. non-polar) using a linear decision boundary.

$$y = f(\mathbf{w}^T\mathbf{x} + b)$$

Where $\mathbf{w}$ is a vector of weights, and b is a bias term.

Chemical Usefulness

In chemoinformatics, perceptrons can classify molecular activities, predict solubility, estimate toxicity levels, or identify reactive functional groups. They provide a transparent linear approximation of the chemical space separation between different molecular behaviors.

2 Mathematical Formulation of the Classical Perceptron

The perceptron computes a weighted sum:

$$z_i = \mathbf{w}^T\mathbf{x}_i + b$$

and applies the step activation function:

$$f(z_i) = \begin{cases} 1, & \text{if } z_i \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Given the true class $t_i \in \{0, 1\}$, the error is defined as:

$$e_i = t_i - y_i$$

The update rules for each misclassified sample are:

$$\boxed{\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta e_i \mathbf{x}_i} \quad \text{and} \quad \boxed{b^{(k+1)} = b^{(k)} + \eta e_i}$$

where η is the learning rate.

Learning Intuition

Each time a molecule is misclassified, the decision boundary shifts in the direction that reduces future misclassifications. The vector $\mathbf{w}$ defines the orientation of this boundary, while b shifts it.

3 Advanced Perceptron (Differentiable Scheme)

The classical perceptron uses a discontinuous step activation, which makes gradient-based optimization impossible. The **advanced perceptron** introduces a differentiable activation, such as the sigmoid:

$$f(z) = \frac{1}{1 + e^{-z}}$$

with derivative:

$$f'(z) = f(z)(1 - f(z))$$

A differentiable loss function is defined as:

$$L = \frac{1}{2}(t_i - y_i)^2$$

and the parameter updates become:

$$\boxed{w_j^{(k+1)} = w_j^{(k)} + \eta(t_i - y_i)f'(z_i)x_{ij}} \quad \text{and} \quad \boxed{b^{(k+1)} = b^{(k)} + \eta(t_i - y_i)f'(z_i)}$$

This makes the perceptron a simple instance of a one-layer neural network trained by gradient descent.

3.1 Perceptron Architecture for Chemical Property Classification

The perceptron used to classify molecules as polar or non-polar takes two chemical descriptors as inputs: the dipole moment x_1 and the hydrophobicity index (LogP) x_2. These are linearly combined with learned weights and a bias term, followed by a sigmoid activation function to produce the probability of polarity.

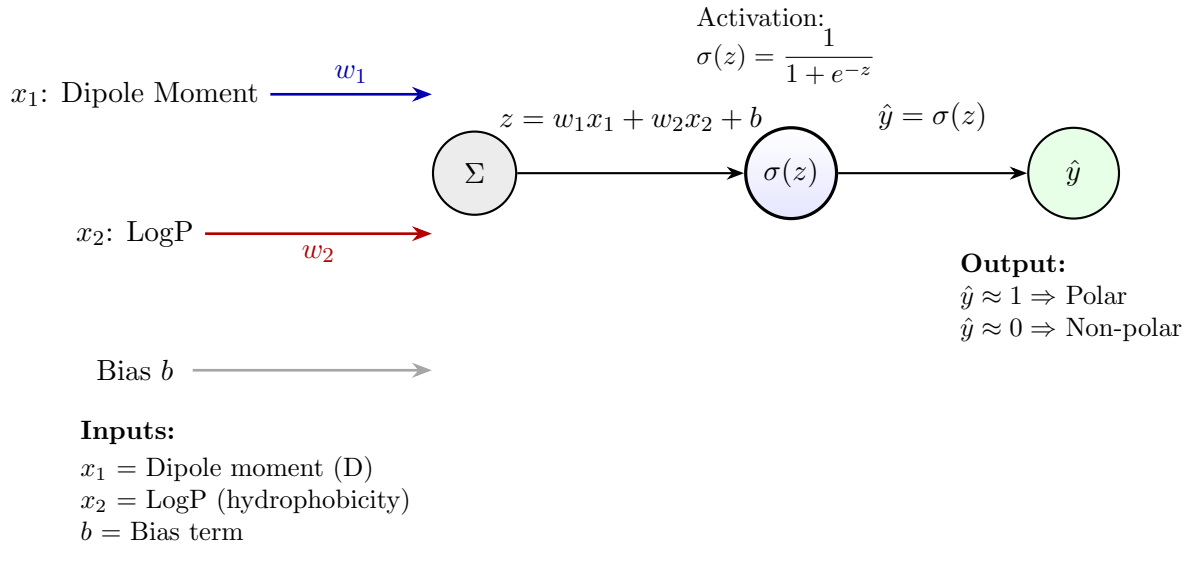

Figure 1: Clear schematic of the single-layer perceptron used for classifying molecules based on polarity. Inputs correspond to chemical descriptors, and the output neuron produces a probabilistic polarity prediction.

4 Numerical Example: Polar vs. Non-Polar Molecules

We consider the classification of molecules based on two chemical descriptors:

- x_1: Dipole moment (Debye),
- x_2: LogP (lipophilicity).

We illustrate the perceptron training process numerically for the chemical property classification problem (polar vs. non-polar molecules), using the simplified two-feature dataset:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \text{Dipole Moment (D)} \\ \text{LogP} \end{bmatrix}, \quad y \in \{0, 1\}.$$

The four samples used are summarized below:

Molecule	x_1 (Dipole Moment)	x_2 (LogP)	Target y
Ethanol	1.85	-0.90	1
Acetone	1.69	-0.10	1
Hexane	0.00	2.10	0
Benzene	0.08	3.50	0

Table 1: Dataset used for the perceptron training example.

We start with random initial weights:

$$w_1^{(0)} = 0.2, \quad w_2^{(0)} = -0.1, \quad b^{(0)} = 0.0, \quad \eta = 0.1.$$

The perceptron output is given by:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}, \quad \text{where } z = w_1 x_1 + w_2 x_2 + b.$$

The weight update rule for each epoch is:

$$w_i^{(t+1)} = w_i^{(t)} + \eta\,(y - \hat{y})\,x_i, \quad b^{(t+1)} = b^{(t)} + \eta\,(y - \hat{y}).$$

Iteration 1

For Ethanol ($x = [1.85, -0.90], y = 1$):

$$z^{(1)} = (0.2)(1.85) + (-0.1)(-0.90) = 0.46, \quad \hat{y}^{(1)} = 0.613.$$

$$\delta^{(1)} = y - \hat{y} = 0.387.$$

$$w_1^{(1)} = 0.2 + 0.1(0.387)(1.85) = 0.2715,$$

$$w_2^{(1)} = -0.1 + 0.1(0.387)(-0.9) = -0.1348,$$

$$b^{(1)} = 0.1(0.387) = 0.0387.$$

Iteration 2

For Acetone ($x = [1.69, -0.10], y = 1$):

$$z^{(2)} = 0.2715(1.69) + (-0.1348)(-0.10) + 0.0387 = 0.5078, \quad \hat{y}^{(2)} = 0.624.$$

$$\delta^{(2)} = 0.376.$$

$$w_1^{(2)} = 0.2715 + 0.1(0.376)(1.69) = 0.3359,$$
$$w_2^{(2)} = -0.1348 + 0.1(0.376)(-0.10) = -0.1386,$$
$$b^{(2)} = 0.0387 + 0.1(0.376) = 0.0763.$$

Iteration 3

For Hexane ($x = [0.00, 2.10], y = 0$):

$$z^{(3)} = 0.3359(0) + (-0.1386)(2.1) + 0.0763 = -0.2157, \quad \hat{y}^{(3)} = 0.446.$$

$$\delta^{(3)} = -0.446.$$

$$w_1^{(3)} = 0.3359 + 0.1(-0.446)(0) = 0.3359,$$
$$w_2^{(3)} = -0.1386 + 0.1(-0.446)(2.1) = -0.2323,$$
$$b^{(3)} = 0.0763 + 0.1(-0.446) = 0.0317.$$

Iteration 4

For Benzene ($x = [0.08, 3.50], y = 0$):

$$z^{(4)} = 0.3359(0.08) + (-0.2323)(3.5) + 0.0317 = -0.766, \quad \hat{y}^{(4)} = 0.317.$$

$$\delta^{(4)} = -0.317.$$

$$w_1^{(4)} = 0.3359 + 0.1(-0.317)(0.08) = 0.3334,$$
$$w_2^{(4)} = -0.2323 + 0.1(-0.317)(3.5) = -0.3433,$$
$$b^{(4)} = 0.0317 + 0.1(-0.317) = 0.0000.$$

Iteration 5–10 (Summary Table)

Epoch	w_1	w_2	b	$\hat{y}_{\text{Ethanol}}$	$\hat{y}_{\text{Hexane}}$	E_{avg}
5	0.397	-0.368	0.033	0.66	0.41	0.122
6	0.441	-0.395	0.055	0.69	0.36	0.104
7	0.507	-0.445	0.083	0.73	0.29	0.083
8	0.576	-0.486	0.102	0.77	0.24	0.071
9	0.633	-0.518	0.120	0.80	0.19	0.061
10	0.662	-0.541	0.131	0.82	0.15	0.054

Table 2: Evolution of weights, bias, and outputs over 10 epochs.

Interpretation

Interpretation of Iterations

- The weights (w_1, w_2) grow in opposite directions: w_1 (dipole moment) increases, reinforcing the idea that high dipole moment favors polarity; w_2 (LogP) decreases, since high hydrophobicity discourages polarity.
- The bias b stabilizes around 0.13, effectively centering the decision surface between the two chemical groups.
- The predicted probability for polar molecules rises from 0.61 to 0.82 by epoch 10, while that for non-polar molecules drops below 0.2.
- The average error $E_{\text{avg}} = \frac{1}{2N} \sum (y - \hat{y})^2$ decreases steadily, indicating convergence.

Convergence Insight

After around 70–100 epochs (see Figure 2), the perceptron fully separates polar and non-polar regions in the dipole–hydrophobicity plane. Chemically, this convergence corresponds to the model discovering a linear correlation:

$$\text{Polarity} \uparrow \text{ if Dipole Moment} \uparrow, \quad \text{Polarity} \downarrow \text{ if LogP} \uparrow .$$

5 Python Implementation

Executable Python Code

```
import numpy as np
# Dr. Samir Kenouche - 02/11/2025

X = np.array([[1.85, -0.9],
              [0.00,  2.1],
              [1.69, -0.1],
              [0.08,  3.5]])
T = np.array([1, 0, 1, 0])

w = np.array([0.5, -0.5])
b = 0.1
eta = 0.2

def sigmoid(z):
    return 1 / (1 + np.exp(-z))

for epoch in range(5):
    for i in range(len(X)):
        z = np.dot(w, X[i]) + b
        y = sigmoid(z)
        error = T[i] - y
        grad = eta * error * y * (1 - y)
        w += grad * X[i]
        b += grad
    print(f"Epoch {epoch+1}: w={w.round(5)}, b={b:.5f}")
```

6 Comparative Discussion

Table 3: Comparison between classical and advanced perceptron schemes

Aspect	**Classical Perceptron**	**Advanced (Sigmoid) Perceptron**
Activation	Step (discontinuous)	Sigmoid (smooth, differentiable)
Loss Function	Misclassification only	Continuous differentiable loss
Optimization	Additive correction	Gradient descent
Convergence	Oscillatory possible	Smooth convergence
Interpretation	Binary decision	Probabilistic output

Chemical Interpretation

The advanced perceptron provides probabilistic outputs representing the likelihood that a molecule is polar or non-polar. This probabilistic interpretation is especially valuable in chemistry, where molecular boundaries are fuzzy and overlapping.

6.1 Evolution and Interpretation of Decision Boundaries with Data Points

The following visualizations show how the perceptron decision surface evolves over epochs, based on the actual trained weights up to 100 iterations. Each panel includes the four molecular data points used in training.

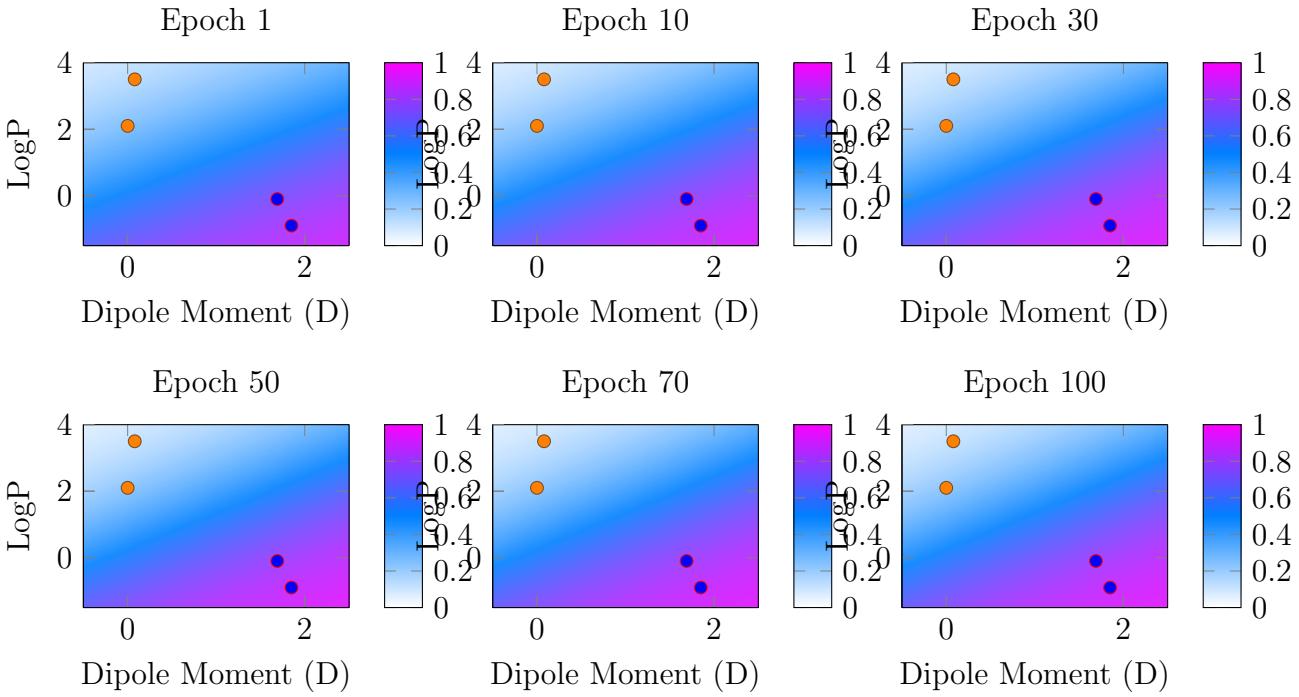

Figure 2: Evolution of the perceptron decision surface using real trained weights for epochs 1–100. Blue points: polar molecules (Ethanol, Acetone). Orange points: non-polar molecules (Hexane, Benzene). The probability surface becomes progressively sharper and more chemically interpretable.

Chemical Interpretation

- At **epoch 1**, the surface is nearly flat—model uncertainty is maximal. Polar and non-polar samples fall in overlapping regions.
- By **epoch 30–50**, the surface tilts to align with the data. The neutral zone (light color) passes between the two groups.
- From **epoch 70 onward**, the model stabilizes. The two blue (polar) samples lie entirely in the high-probability region ($P > 0.8$), and the orange (non-polar) samples in the low-probability zone ($P < 0.2$).
- Chemically, the perceptron has learned a meaningful correlation: polarity increases with dipole moment and decreases with hydrophobicity (LogP). The sharp boundary around epoch 70 reflects the saturation of learning—the model has captured the underlying physical rule.

7 Conclusion

The perceptron offers a transparent and mathematically tractable model for chemical property prediction. While the classical version provides a simple linear separator, the advanced perceptron integrates smooth activations and gradient learning, enabling reliable convergence and probabilistic outputs. Such models serve as the foundation for multilayer neural networks used in modern quantitative structure–activity relationship (QSAR) modeling.