Performance of the Vector Perceptron With Ephaptic Coupling

A. Chawla

October 28, 2025

Abstract

In this note, the author presents preliminary simulation results comparing the performance of the vector ephaptic perceptron against that of the classical perceptron, under conditions of symmetric, differential and random-differential coupling. The first, symmetric, case is found to outperform the other two, marginally, over the range tested, although statistical error bars are large as well.

1 Introduction

The vector ephaptic perceptron was introduced in [1]. It is a model that is inspired by the phenomenon of ephaptic coupling which has been investigated in detail in the computational neuroscience literature. In ephaptic coupling two or more neurons are involved. This is replicated in the perceptron setting by boosting the scalar perceptron to the vector case and introducing cross-coupling terms.

2 Methods and Results

In this work, we attempt to characterize the vector ephaptic perceptron's performance with respect to the regular Rosenblatt perceptron. We find that for a large number of symmetric coupling matrices, it performs marginally better than the Rosenblatt model. However, statistical significance tests such as the t-test were notably not performed and so this performance enhancement cannot be confirmed at this stage and deemed to be statistically significant. The enclosed algorithm in Python was used to perform the tests and the included figure shows the results.

3 Conclusion

As indicated in the figure, for negative coupling strength, the symmetric case performs well and for positive coupling strength, the random-differential case

performs well. In all settings, the ephaptic perceptron is marginally better than the corresponding regular perceptron for most coupling strengths. We can infer from this performance that the cross-coupling provides a novel degree of freedom, though performance gains are modest. In future work, the multi-layer version will be compared with the corresponding multi-layer (regular) perceptron.

Acknowledgments

This work was generated with the assistance of LLMs.

References

[1] Aman Chawla. A mathematical model for the vector ephaptic perceptron. *Zenodo*, 2022.

Algorithm 1 Peformance characterizing script, part 1.

```
"""
vector_ephaptic_perceptron_differential.py
Studies symmetric and differential ephaptic couplings:
λ12 = λ21 (symmetric)
λ12 = -λ21 (differential)
and evaluates performance statistics for both perceptrons.
"""
import numpy as np
import matplotlib.pyplot as plt
import time
# ============================================================
# PARAMETERS
# ============================================================
N = 2
input_dim = 3
num_samples = 1000
learning_rate = 0.01
num_epochs = 50
num_trials = 10
lambda_values = np.linspace(-0.8, 0.8, 17)
patterns = ["symmetric", "differential", "random_differential"]
# ============================================================
# HELPER FUNCTIONS
# ============================================================
def generate_data():
X = np.random.randn(num_samples, input_dim)
true_w = np.random.randn(input_dim, 1)
y = np.sign(X @ true_w + 0.2 * np.random.randn(num_samples, 1)).flatten()
return X, y
def train_regular(X, y, input_dim, lr, epochs):
w = np.random.randn(input_dim, 1)
b = 0.0
for _ in range(epochs):
for i in range(len(X)):
xi = X[i, :].reshape(-1, 1)
yi = y[i]
y_pred = np.sign(w.T @ xi + b)
if y_pred != yi:
w += lr * yi * xi
b += lr * yi
preds = np.sign(X @ w + b)
return np.mean(preds.flatten() == y)
def train_ephaptic(X, y, input_dim, N, lr, epochs, lambda_mat):
W = np.random.randn(input_dim, N)
b = np.zeros((N, 1))
```

Algorithm 2 Performance characterizing script, part 2.

```
for _ in range(epochs):
for i in range(len(X)):
xi = X[i, :].reshape(-1, 1)
yi = y[i]
v = W.T @ xi + b
v_tilde = v + lambda_mat @ v
y_pred = np.sign(np.mean(v_tilde))
if y_pred != yi:
for k in range(N):
W[:, k:k+1] += lr * yi * xi
b[k] += lr * yi
preds = []
for j in range(len(X)):
v = W.T @ X[j, :].reshape(-1, 1) + b
v_tilde = v + lambda_mat @ v
preds.append(np.sign(np.mean(v_tilde)))
preds = np.array(preds).flatten()
return np.mean(preds == y)
# ============================================================
# MAIN EXPERIMENT
# ============================================================
results = {p: [] for p in patterns}
for pattern in patterns:
print(f"\n=== PATTERN: {pattern.upper()} ===")
for lam in lambda_values:
acc_reg_trials = []
acc_eph_trials = []
for trial in range(num_trials):
np.random.seed(int(time.time()) + trial)
X, y = generate_data()
# Define coupling pattern
if pattern == "symmetric":
lambda_mat = np.array([[0.0, lam], [lam, 0.0]])
elif pattern == "differential":
lambda_mat = np.array([[0.0, lam], [-lam, 0.0]])
elif pattern == "random_differential":
delta = np.random.uniform(-abs(lam)/2, abs(lam)/2)
lambda_mat = np.array([[0.0, lam + delta],
[-lam + delta, 0.0]])
acc_reg = train_regular(X, y, input_dim, learning_rate, num_epochs)
acc_eph = train_ephaptic(X, y, input_dim, N, learning_rate, num_epochs,
lambda_mat)
acc_reg_trials.append(acc_reg)
acc_eph_trials.append(acc_eph)
# Compute stats
stats = {
"lambda": lam,
"reg_mean": np.mean(acc_reg_trials),
"reg_std": np.std(acc_reg_trials),
"eph_mean": np.mean(acc_eph_trials),
"eph_std": np.std(acc_eph_trials)
}
```

Algorithm 3 Performance characterizing script, part 3.

```
results[pattern].append(stats)
print(f"λ={lam:+.2f} | Reg={stats['reg_mean']*100:5.2f}±{stats['reg_std']*100:4.2f}
| "
f"Eph={stats['eph_mean']*100:5.2f}±{stats['eph_std']*100:4.2f}")
# ============================================================
# PLOTS
# ============================================================
plt.figure(figsize=(9,6))
for pattern, color in zip(patterns, ['b', 'r', 'g']):
lam_vals = [s["lambda"] for s in results[pattern]]
eph_mean = [s["eph_mean"] for s in results[pattern]]
eph_std = [s["eph_std"] for s in results[pattern]]
reg_mean = [s["reg_mean"] for s in results[pattern]]
reg_std = [s["reg_std"] for s in results[pattern]]
plt.errorbar(lam_vals, eph_mean, yerr=eph_std, fmt='o--', color=color, la-
bel=pattern)
plt.errorbar(lam_vals, reg_mean, yerr=reg_std, fmt='*--', color=color, la-
bel=pattern+' regular')
plt.axvline(0, color='gray', linestyle='--')
plt.xlabel('Coupling Strength λ')
plt.ylabel('Ephaptic Accuracy (mean ± std)')
plt.title(f'Vector Ephaptic Perceptron under Symmetric and Differential Cou-
pling\n({num_trials} trials per λ)')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

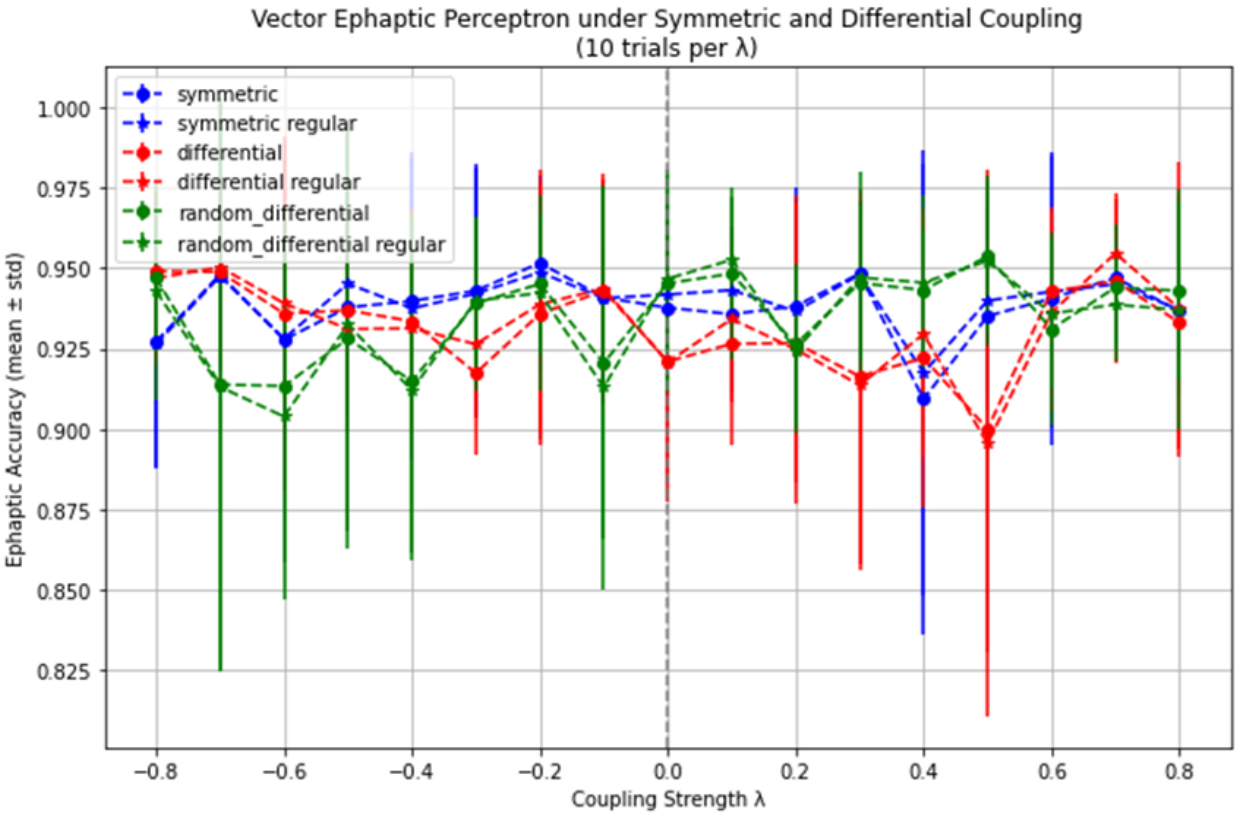

Figure 1: Accuracy of perceptrons with symmetric, differential and "random" differential ephaptic coupling matrices. The "asterisk"-mark indicates the corresponding performance of the regular perceptron. For negative coupling strength, the symmetric case performs well and for positive coupling strength, the random-differential case. In all settings, the ephaptic perceptron is marginally better than the regular perceptron for most coupling strengths.