Classification of Phase Transitions Using Machine Learning

Morten Hjorth-Jensen

University of Oslo

May 20, 2025

Outline

Ising Model Order Parameter

- Spins: $\sigma_i = \pm 1$
- **Order parameter (magnetization):**

$$M = \left\langle \frac{1}{N} \sum_i \sigma_i \right\rangle$$

- **Susceptibility:**

$$\chi = \frac{N}{T}(\langle M^2 \rangle - \langle M \rangle^2)$$

- **Energy:**

$$E = \left\langle -J \sum_{\langle ij \rangle} \sigma_i \sigma_j \right\rangle$$

- **Specific heat:**

$$C = \frac{1}{NT^2}(\langle E^2 \rangle - \langle E \rangle^2)$$

Potts Model Order Parameter

- q-state spins: $\sigma_i \in \{1, 2, ..., q\}$
- Order parameter: fraction of majority state

$$M_q = \left\langle \frac{q \cdot \max_k n_k - N}{N(q-1)} \right\rangle, \quad n_k = \text{ of spins in state } k$$

- Susceptibility and specific heat computed analogously
- Critical behavior depends on q and dimensionality

Ising Model: Metropolis Algorithm (PyTorch)

```
def local_energy(config, i, j):
    L = config.shape[0]
    spin = config[i, j]
    neighbors = config[(i+1)%L, j] + config[(i-1)%L, j] + \
                config[i, (j+1)%L] + config[i, (j-1)%L]
    return -spin * neighbors

def metropolis_step(config, beta):
    L = config.shape[0]
    for _ in range(L * L):
        i, j = torch.randint(0, L, (2,))
        dE = 2 * local_energy(config, i, j)
        if dE <= 0 or torch.rand(1) < torch.exp(-beta * dE):
            config[i, j] *= -1
    return config
```

Compute Observables (Ising)

```python
def compute_observables(config, J=1.0):
    L = config.shape[0]
    energy, magnet = 0.0, config.sum().item()
    for i in range(L):
        for j in range(L):
            S = config[i, j]
            neighbors = config[(i+1)%L, j] + config[i, (j+1)%
                L]
            energy -= J * S * neighbors
    norm = L * L
    return energy / norm, magnet / norm
```

Dataset Generator (Ising Model)

```
def generate_ising_data(L, T_vals, n_samples):
    configs, energies, mags, labels = [], [], [], []
    for T in T_vals:
        beta = 1.0 / T
        for _ in range(n_samples):
            config = torch.randint(0, 2, (L, L)) * 2 - 1
            for _ in range(500):  # Thermalization
                config = metropolis_step(config, beta)
            e, m = compute_observables(config)
            configs.append(config.clone())
            energies.append(e)
            mags.append(m)
            labels.append(int(T < 2.3))  # crude binary label
    return torch.stack(configs), torch.tensor(labels),
        energies, mags
```

Visualize Magnetization vs Temperature

```
import matplotlib.pyplot as plt

def plot_magnetization(T_vals, mags):
    mags = torch.tensor(mags).reshape(len(T_vals), -1)
    avg_mag = mags.abs().mean(dim=1)
    plt.plot(T_vals, avg_mag)
    plt.xlabel("Temperature T")
    plt.ylabel("Magnetization |M|")
    plt.title("Order Parameter vs Temperature")
    plt.grid(True)
    plt.show()
```

CNN Classifier (PyTorch)

```python
class PhaseClassifier(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 32, 3, padding=1)
        self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
        self.fc1 = nn.Linear(64*8*8, 128)
        self.fc2 = nn.Linear(128, 2)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.max_pool2d(x, 2)
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, 2)
        x = x.view(x.size(0), -1)
        x = F.relu(self.fc1(x))
        return self.fc2(x)
```

Training + Results

- Train CNN using configurations labeled by T
- Accuracy high away from T_c, drops near T_c
- Predict critical point by analyzing output confidence
- Extension: regression instead of classification to learn T

Summary

- Order parameters help interpret ML models
- PyTorch enables efficient simulation and classification
- ML can discover phase boundaries without explicit physical models
- Open challenges: interpretability, generalization, scaling

References

- Carrasquilla & Melko, "Machine learning phases of matter", Nature Phys. (2017)
- Wang, "Discovering phase transitions with unsupervised learning", PRB (2016)
- Mehta et al., "A high-bias, low-variance introduction to ML for physicists", Phys. Rep. (2019)