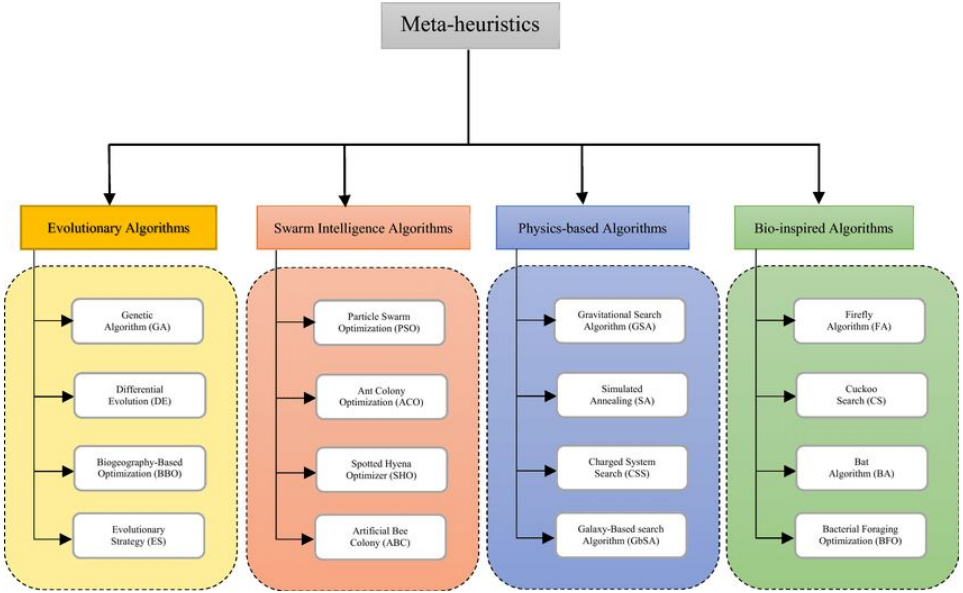
# BackPropagation? Anche No...

DevFest Modena 2025

# Metaeuristica

- Famiglia di Algoritmi
  - Non Intuitivi
- Problemi di Ottimizzazione
  - Min, Max
- Nature Inspired, Evolutionary Inspired, …
  - ACO, PSO, GA, …



# Crystal Structure Algorithm

- Nature Inspired
  - Reticolo Cristallino
- Parametric Free
  - Senza Iperparametri
  - Senza Operatori

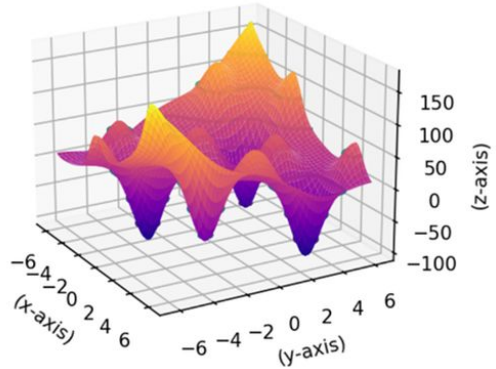
# Crystal Structure Algorithm

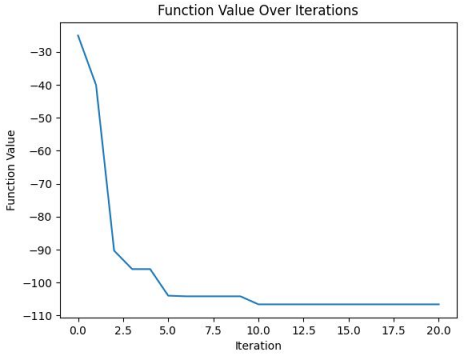
- Sia Ricerca Globale che Locale
  - Creazione Cristalli
  - Bootstrapping
  - No Minimi/Massimi Locali

```
for i in range(0, nb_iterations):
    for crystal_idx in range(0, nb_crystal):
        new_crystals = np.array([])
        Cr_main = self._take_random_crystals(crystals=cry
        Cr_old = crystals[crystal_idx]
        Fc = self._take_random_crystals(crystals=crystals
        r, r_1, r_2, r_3 = self.__compute_r_values()
        Cr_new = self._compute_simple_cubicle(Cr_old=Cr_c
        new_crystals = np.hstack((new_crystals, Cr_new))
        Cr_new = self._compute_cubicle_with_best_crystals
        new_crystals = np.vstack((new_crystals, Cr_new))
        Cr_new = self._compute_cubicle_with_mean_crystals
        new_crystals = np.vstack((new_crystals, Cr_new))
        Cr_new = self._compute_cubicle_with_best_and_mean
        new_crystals = np.vstack((new_crystals, Cr_new))
```

# Crystal Structure Algorithm

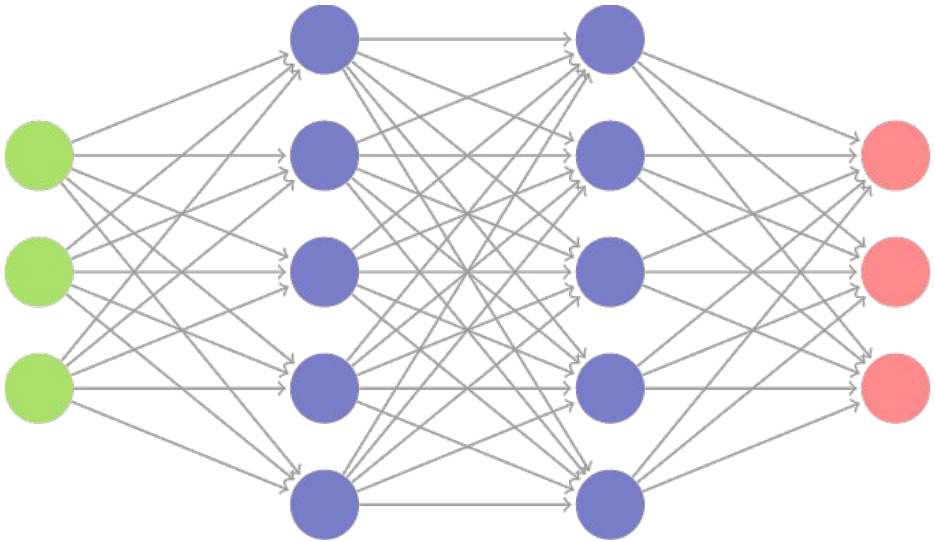
- Valutazione dei Cristalli tramite Fitness
  - Il migliore è la soluzione



# Ottimizzazione Pesi

- Utilizzato su NN
  - Modifica Dei Pesi
- Senza BackProp
  - No Over/Under Fitting
  - No Minimi Locali
  - No Differenziabilità



# Ottimizzazione Pesi

- Iris Dataset
  - Classificazione
- Steps
  - Flat pesi. Ogni array è un Cristallo.
    - Basato su Shape Rete
  - Ottimizzazione
  - Ricostruzione shape e Assegnazione pesi
  - Valutazione: Fitness = Loss

```python
def _flat_weights(self) -> np.ndarray:
    weights = []
    for p in self._model.parameters():
        weights.append(p.data.view(-1))
    weights = torch.cat(weights).detach().cpu().numpy()
    return weights

def _create_crystals(self, lb: int, ub: int, nb_crystal: int) -> np.array:
    base_weights = self._flat_weights()
    dim = base_weights.size
    random_crystals = np.random.uniform(low=lb, high=ub, size=(nb_crystal - 1, dim))
    crystals = np.vstack([base_weights, random_crystals])
    print(f"Created {crystals.shape[0]} Crystals With {crystals.shape[1]} Elements")
    return crystals
```

```python
def _assign_weights(self, crystal: np.ndarray) -> None:
    cum_p = 0
    for p in self._model.parameters():
        total_params = p.numel()
        tensor = torch.tensor(crystal[cum_p:cum_p + total_params])
        reshaped_tensor = tensor.view(p.shape)
        p.data.copy_(reshaped_tensor)
        cum_p += total_params

def _evaluate_crystals(self, crystals: np.ndarray, X_train: torch.T
    losses = []
    for crystal in crystals:
        self._assign_weights(crystal)
        with torch.no_grad():
            outputs = self._model(X_train)
            loss = self._loss_function(outputs, y_train)

        losses.append(loss.item())

    fitnesses = np.array(losses).reshape(-1, 1)
    best_index = np.argmin(fitnesses)
    best_value = fitnesses[best_index, 0]
    return fitnesses, best_value, best_index
```

# Ottimizzazione Pesi

```
Accuracy Before Train On Test Set with Backprop: 36.67%
Start optimization for Backprop Strategy
Epoch 0, Loss: 0.3089
Epoch 10, Loss: 0.0071
Epoch 20, Loss: 0.0010
Accuracy on Train Set: 97.50%
Accuracy After Train On Test Set with Backprop: 96.67%
Elapsed: 4.4316s
Accuracy Before Train On Test Set with Crystal: 36.67%
Start optimization for Crystal Strategy
Created 15 Crystals With 131 Elements
Iter 0. Current Best Crystal Fitness Is 1.5699747800827026
Iter 10. Current Best Crystal Fitness Is 0.38595184683799744
Iter 20. Current Best Crystal Fitness Is 0.16885297000408173
Iter 30. Current Best Crystal Fitness Is 0.14107681810855865
Iter 40. Current Best Crystal Fitness Is 0.093870282173156 74
Iter 50. Current Best Crystal Fitness Is 0.093870282173156 74
Accuracy on Train Set: 96.67%
Accuracy After Train On Test Set with Crystal: 93.33%
Elapsed: 1.1045s
```

# Blog