Fiche récap : Initiez-vous à l'apprentissage semi-supervisé

Data

Définitions

Apprentissage semi-supervisé
Méthode qui (1) utilise un petit ensemble de données labellisées pour guider l'apprentissage et, (2) exploite un grand volume de données non labellisées pour affiner la compréhension du modèle.

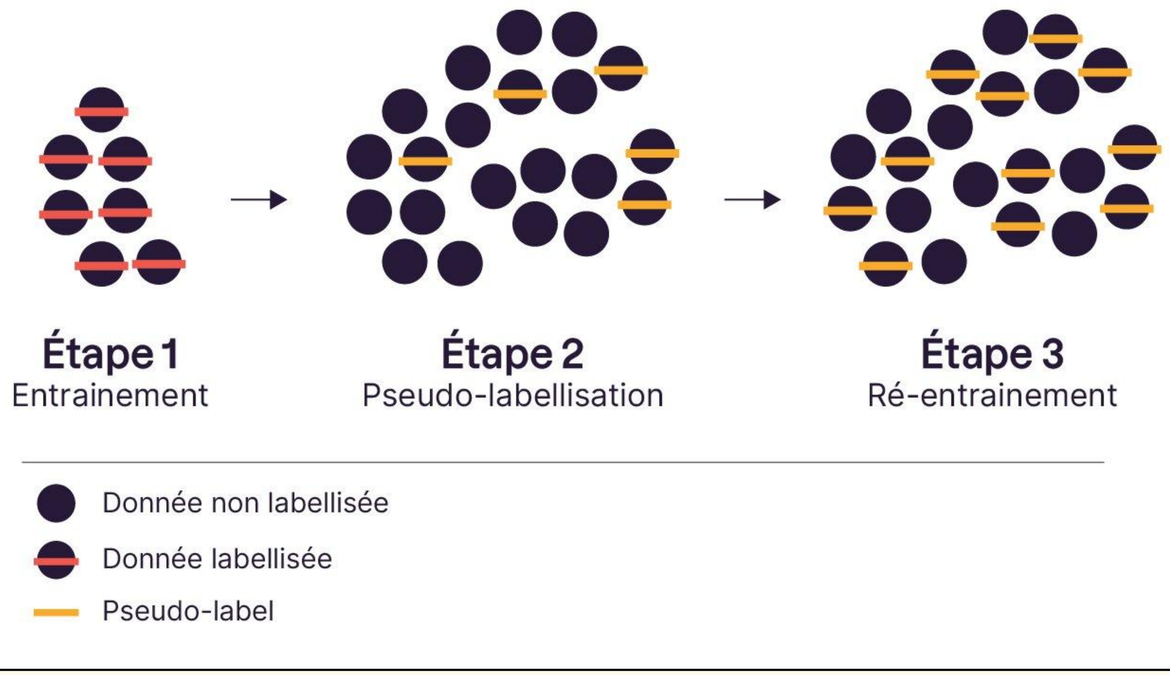
Pseudo-labellisation
Technique permettant au modèle de s'auto-enseigner en attribuant des labels (étiquettes) temporaires aux données non labellisées. Seules les prédictions dépassant un seuil de confiance élevé (ex. : `0.95`) sont retenues et réintégrées à l'entraînement.

Propagation sur graphes
Processus où les données sont modélisées comme des nœuds ; et leur similarité comme des arêtes : les labels connus se diffusent (se propagent) le long de ces connexions pour inférer les classes manquantes par continuité.

SGAN (Semi-Supervised GAN)
Réseau antagoniste génératif où le discriminateur apprend à distinguer le vrai du faux et à prédire la catégorie de la donnée parmi les classes disponibles.

Régularisation par cohérence
Méthode visant à stabiliser les prédictions d'un modèle malgré des perturbations appliquées sur les données d'entrée.

Pseudo-labellisation



Bonnes pratiques

- ✅ Définir un seuil de confiance élevé pour les pseudo-labels.
- ✅ Observer les métriques après chaque itération du modèle.
- ✅ Utiliser un modèle pré-entraîné pour générer des embeddings.
- ✅ Ajuster les hyperparamètres du graphe avec soin.
- ✅ Appliquer des perturbations réalistes pour tester la cohérence.
- ✅ Combiner plusieurs méthodes SSL pour plus de robustesse.
- ✅ Visualiser l'évolution des pertes pour suivre la convergence.
- ✅ Tester différentes variantes d'augmentations de données.

Erreurs classiques

- ❌ Choisir un seuil trop bas pour les pseudo-labels confiants.
- ❌ Ignorer la calibration du modèle pour les prédictions.
- ❌ Négliger l'effet des classes rares sur les performances.
- ❌ Connecter trop de voisins dans les graphes de similarité.
- ❌ Sous-estimer l'impact des embeddings de mauvaise qualité.
- ❌ Oublier de valider sur un jeu de test bien construit.
- ❌ Confondre bruit visuel et signal pertinent dans les images.
- ❌ Négliger les biais dans les données non labellisées.

Fiche récap : Initiez-vous à l'apprentissage semi-supervisé

Data

Pseudo-labellisation

```python
# Le seuil de confiance est crucial pour sélectionner les prédictions fiables
threshold = 0.95

# Calculer les probabilités à partir des logits (outputs)
probabilities = torch.softmax(outputs, dim=1)

# Obtenir la probabilité maximale et la classe prédite pour chaque image
max_probs, predicted_classes = torch.max(probabilities, dim=1)

# Créer un masque pour filtrer les prédictions confidentes
confident_mask = max_probs > threshold

# Appliquer le masque pour obtenir les pseudo-labels et les probabilités correspondantes
pseudo_labels = predicted_classes[confident_mask]
```

Propagation sur graphes

```python
# Préparer des labels initiaux (les 'indics')
# On utilise -1 pour marquer les données non labellisées (les 'mystères')
# 'labels_array' contient les vrais labels, 'labeled_indices' les indices connus.
labels_for_spreading = np.full(len(train_dataset), -1, dtype=int)
labels_for_spreading[labeled_indices] = labels_array[labeled_indices]

# Instancier le  modèle de propagation
# 'kernel='knn'' utilise les plus proches voisins.
# 'n_neighbors=10' définit la taille de la communauté d'influence.
label_spreading_model = LabelSpreading(kernel='knn', n_neighbors=10, n_jobs=-1)

# Propager des labels
# 'all_embeddings' (les coordonnées GPS) et 'labels_for_spreading' sont les entrées.
label_spreading_model.fit(all_embeddings, labels_for_spreading)

# Récupérer des labels propagés (la transduction)
predicted_labels = label_spreading_model.transduction_
```

SGANs

```python
# 1. Perte supervisée (données réelles labellisées)
#classification de 0 à n_classes-1
real_labeled_logits = D(labeled_imgs)
d_sup_loss = supervised_criterion(real_labeled_logits[:, :n_classes], labels)

# 2. Perte non supervisée (données réelles non labellisées)
# corriger pour 'réel' (somme des probs 0-6 élevée)
real_unlabeled_logits = D(unlabeled_imgs)
real_probs = torch.softmax(real_unlabeled_logits, dim=1)[:, :n_classes].sum(1)
d_real_unsup_loss = -torch.log(real_probs + 1e-10).mean()

# 3. Perte non supervisée (données fausses/générées)
#corriger pour 'faux' (prob n_classes (7) élevée)
fake_logits = D(fake_imgs.detach())
fake_probs = torch.softmax(fake_logits, dim=1)[:, n_classes]
d_fake_unsup_loss = -torch.log(fake_probs + 1e-10).mean()

# Perte totale du Discriminateur
d_loss = d_sup_loss + d_real_unsup_loss + d_fake_unsup_loss
d_loss.backward()
d_optimizer.step()
```

Régularisation par cohérence

```python
# Initier la fonction de perte
# Utilisation de l'Erreur Quadratique Moyenne (MSE)
consistency_criterion = nn.MSELoss()
# Poids de la perte de cohérence
consistency_weight = 0.1

# Prédire (après avoir passé les données non labellisées par les modèles)
# teacher_preds : sur l'image faiblement augmentée (weak_unlabeled)
# student_preds : sur l'image fortement augmentée (strong_unlabeled)

# Obtenir les probabilités (softmax) pour la comparaison
# F.softmax est appliqué pour transformer les logits en probabilités
with torch.no_grad():
    teacher_probs = F.softmax(teacher_preds, dim=1)

student_probs = F.softmax(student_preds, dim=1)

# Calculer la Perte de Cohérence (comparaison des probabilités)
loss_con = consistency_criterion(student_probs, teacher_probs)

# Combiner la perte totale (avec la perte supervisée loss_sup)
total_loss = loss_sup + consistency_weight * loss_con
```