Séparateurs à vastes marges

En supposant qu'il existe un séparateur linéaire $\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$, où $\boldsymbol{w} = (w_1, w_2)$. Autrement dit

$$w_1 x + w_2 y + b = 0 \Leftrightarrow y = \frac{-w_1}{w_2} x - \frac{b}{w_2}$$

On souhaite maximiser la distance entre ce séparateur et les points les plus proches. C'est à dire que l'on cherche à maximiser les marges, dont les équations sont:

$$\begin{aligned} \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = \pm 1 \quad &\Leftrightarrow w_1 x + w_2 y + b = \pm 1 \\ &\Leftrightarrow y = \tfrac{-w_1}{w_2} x - \tfrac{b \mp 1}{w_2} \end{aligned}$$

Pour maximiser les marges, il faut maximiser leur distance l'une avec l'autre, ce qui se traduit par maximiser la quantité : $M = \frac{2}{\|\boldsymbol{w}\|}$.

1 SVM sur données synthétiques

1.1 Exercice théorique

Soit un classifieur SVM avec une frontière de décision

$$\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b = 0$$

pour un vecteur de caractéristiques à 3 dimensions. Le vecteur de poids est

$$\boldsymbol{w} = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}^T$$

et $b = 4$.

1. Lesquels des points suivants sont incorrectement classifiés ?

$$\boldsymbol{x}_1 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}^T \qquad \text{et } y_1 = -1$$

$$\boldsymbol{x}_2 = \begin{pmatrix} -1 & -1 & -1 \end{pmatrix}^T \qquad \text{et } y_2 = +1$$

$$\boldsymbol{x}_3 = \begin{pmatrix} -3.99 & -3 & 2 \end{pmatrix}^T \qquad \text{et } y_2 = +1$$

2. Que seraient les variables d'ajustement (*slack variables*) ξ_i pour les trois points ci-dessus, s'ils étaient dans l'ensemble d'entraînement (en supposant que $\boldsymbol{w}$ et b ne changent pas)

1.2 Données séparables linéairement

La documentation sur les fonctions utilisées dans cet exercice sont disponibles en ligne à l'adresse suivante
`https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`.

1. En utilisant la fonction `make_blobs` de `scikit-learn`, générer deux groupes des 50 points chacun, distribués selon des lois gaussiennes isotropiques d'écart-type suffisamment faible (`cluster_std=0.6`) pour qu'ils soient séparables linéairement.

2. Classer ces données au moyen du séparateur à vaste marge contenu dans la librairie `scikit-learn`, appelé `SVC` pour *Support Vector Classifier*. Ce classifieur choisira ainsi le "meilleur" classifieur linéaire en maximisant la marge, représentée à la Figure **??**.

3. Donner les coordonnées des vecteurs de support.

4. Visualiser le résultat.

5. Utiliser un nouveau jeu de données composé des vecteurs de support et de 10 points supplémentaires choisis au hasard dans le jeu de données initial. Recalculer le classifieur optimal. Que constatez-vous ?

1.3 Données presque séparables linéairement

En changeant la déviation standard des deux groupes de points générés dans l'exercice précédent, il est possible de faire se chevaucher légèrement les deux groupes de points. Une classification parfaite avec un classifieur linéaire devient alors impossible, mais l'utilisation de variables d'ajustement (*slack variables*, ξ_i) permet de continuer à travailler avec le formalisme développé dans le cadre de données séparables linéairement en utilisant le lagrangien

$$L(\boldsymbol{w}) = \frac{\| \boldsymbol{w} \|^2}{2} + C \sum_{i=1}^{N} \xi_i.$$

Le paramètre C permet alors de pondérer l'importance de la pénalité attribuée aux points mal classés.

- pour des grandes valeurs de C, la marge est "dure", la pénalité est très importante et presque aucun point n'est autorisé à s'y trouver.

- pour des petites valeurs de C, la marge est plus souple, et peut englober certains points

On demande de:

1. générer deux groupes de points avec la commande `make_blobs` comme précédemment, mais avec une déviation standard supérieure (`cluster_std=1.2`), et visualiser le résultat.

2. étudier l'importance du paramètre C en construisant plusieurs classifieurs SVM avec des valeurs croissantes de C ($c = 0.1, \ldots, c = 10$), et en comparant les résultats.

1.4 Données non séparables linéairement

1. Générer un jeu de données synthétique où les données sont disposées en cercles concentriques, avec la commande `make_circles` de `scikit-learn`:

```
from sklearn.datasets.samples_generator import make_circles
X, y = make_circles(100, factor=.1, noise=.1)
```

2. Essayer de classer ces données avec un classifieur linéaire et visualiser le résultat.

3. Projeter les données dans un espace de dimension supérieure où elles deviennent séparables linéairement et déterminer, dans ce nouvel espace, le meilleur séparateur linéaire. Visualiser le résultat.

   Remarque : On pourra utiliser la transformation suivante :

$$(x, y) \longrightarrow (x, y, e^{-(x^2+y^2)})$$

   On pourra définir une fonction $z(x, y) = e^{-(x^2+y^2)}$. Pour tracer, on pourra s'aider des instructions suivantes :

```
tmp = np.linspace(-2,2,51)
x_grid,y_grid = np.meshgrid(tmp,tmp)
fig = plt.figure()
ax  = fig.add_subplot(111, projection='3d')
ax.plot_surface(x_grid, y_grid, z(x_grid,y_grid))
ax.scatter3D(X[:, 0], X[:, 1], r, c=y, s=50, cmap='autumn')
```

4. Changer le paramètre `kernel='linear'` en `kernel='rbf'` et déterminer le classifieur SVM.

   Cette astuce du noyau (*kernel trick*) permet donc, sans devoir explicitement effectuer la transformation des données, de déterminer le meilleur classifieur (Figure 3).

   On pourra tracer les séparateurs en utilisant la fonction suivante :

```
def plot_svc_decision_function(model, ax=None, plot_support=True):
    """Plot the decision function for a 2D SVC"""
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
```

```
P = model.decision_function(xy).reshape(X.shape)

# plot decision boundary and margins
ax.contour(X, Y, P, colors='k',
           levels=[-1, 0, 1], alpha=0.5,
           linestyles=['--', '-', '--'])

# plot support vectors
if plot_support:
    ax.scatter(model.support_vectors_[:, 0],
               model.support_vectors_[:, 1],
               s=300, linewidth=1.5, facecolors='none');
ax.set_xlim(xlim)
ax.set_ylim(ylim)
```

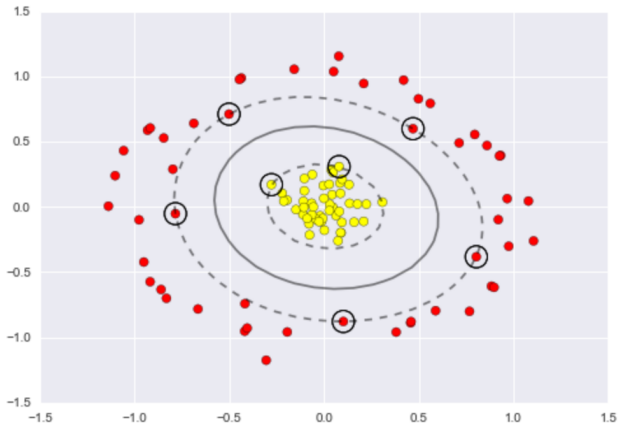

Figure 3: Classifieur SVM avec des données non séparables linéairement

2 Application : reconnaissance faciale

Appliquons les SVMs pour un problème concret de reconnaissance faciale.

1. Télécharger le jeu de données *Labeled Faces in the Wild*, disponible dans la librairie `scikit-learn`. Le téléchargement peut prendre quelques dizaines de secondes, car il contient 1348 images de 62×47 pixels chacune. Ces images contiennent les visages de personnalités politiques (Ariel Sharon, Colin Powell, Donald Rumsfeld, George W Bush, Gerhard Schroeder, Hugo Chavez, Junichiro Koizumi, Tony Blair).

   ```
   from sklearn.datasets import fetch_lfw_people
   faces = fetch_lfw_people(min_faces_per_person=60)
   ```

Puis vérifier que les données sont correctement chargées, en imprimant les noms des personnalités et en affichant certaines des images.

```
print(faces.target_names)
fig, ax = plt.subplots(3, 5)
for i, axi in enumerate(ax.flat):
    axi.imshow(faces.images[i], cmap='bone')
    axi.set(xticks=[], yticks=[],
            xlabel=faces.target_names[faces.target[i]])
```

2. Pour classer ces images, on pourrait utiliser chaque pixel de ces images 62×47 comme une caractéristique indépendante, mais il est plus efficace d'effectuer d'abord un prétraitement par Analyse en Composantes Principales pour réduire la dimension du problème :

```
from sklearn.decomposition import PCA
pca = PCA(n_components=150, whiten=True, random_state=42)
X = pca.fit_transform(faces.data)
```

3. Séparer le jeu de données en deux parties (training set et testing set) puis réaliser l'apprentissage d'un (ou plusieurs) classifieur(s) SVM sur ces données. Quel est leur score ?

4. Evaluer la qualité du/des classifieur(s) avec le jeu de données de test, en représentant la matrice de confusion. En notant C la matrice de confusion, on pourra la représenter avec ce qui suit :

```
sns.heatmap(C, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=faces.target_names,
            yticklabels=faces.target_names)
plt.xlabel('true label')
plt.ylabel('predicted label');
```

5. (bonus) Déterminer les paramètres optimaux du modèle, en utilisant `GridSearchCV`.