Fonctionnement réseau de neurones

Adresse de la démo live : http://90.65.175.190/reseau-neurones--projet-I4/

I. La définition du réseau

La première étape consiste à « dessiner le réseau », c'est-à-dire paramétrer celui-ci afin qu'il puisse effectuer une prédiction.

1. Visuel du formulaire

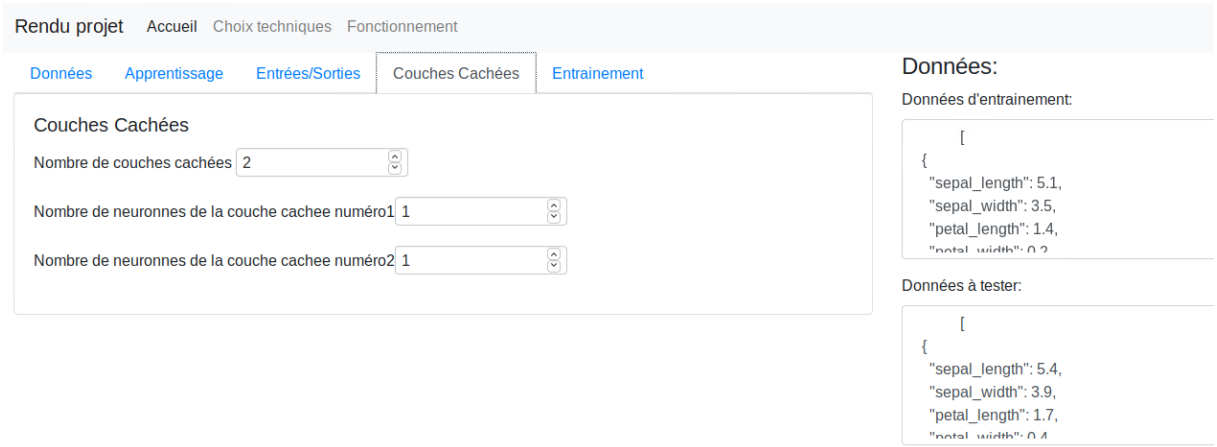


Figure 1 : onglet de paramétrage des couches cachées du réseau

Chaque onglet permet de paramétrer le réseau de neurones, le dernier onglet permet de lancer l'entrainement. Les données d'entrainement et les données à tester sont situées à droite.

2. Génération des couches

```
  //Pour les autres couches cachées
  else if (j > 1) {
    model.add(tf.layers.dense({
      inputShape: [parseInt(document.getElementById("couche_cachee" + (j - 1)))], 1
      activation: fonctionActivation, 2
      units: parseInt(document.getElementById("couche_cachee" + j)), 3
    }))
  }
}
```

Figure 2 : Exemple de génération de couches

La première étape consiste à définir le nombre de paramètres entrants pour chaque neurone de la couche définie. Nous l'avons fixé au nombre de neurones de la couche précédente. (1) Ensuite nous définissons la fonction d'activation (2). Enfin nous déterminons le nombre de neurones présents sur la couche que nous sommes entrain de définir. (3)

II L'entrainement

1. Mapping des sorties

```
/**
 * On map les sorties des données d'entrainement
 * Exemple, si la sortie est Setosa(première sortie)
 * on aura [1,0,0]
 */

const outputData = tf.tensor2d(iris.map(
  function (data) {
    var formattedOutput;
    formattedOutput = [];
    outputList.forEach(element => {
      if (data[outputKey] === element) {
        formattedOutput.push(1);
      } else {
        formattedOutput.push(0);
      }
    });
    return formattedOutput;
  }
));
```

Figure 3 : mapping des sorties

La première étape consiste à transcrire les sorties du jeu de test dans un format compréhensible par Tensorflow.

2. Mapping des paramètres

```
// Mapping des paramètres

for(var i = 0; i< testingSet.length; i++){
  for(var j = 0; j < parametres.length; j++){
    console.log(i, j);
    if (formattedInputs[i] == undefined) {
      formattedInputs[i] = [];
    }
    formattedInputs[i][j] = testingSet[i][parametres[j]];
  }
}
```

Il faut appliquer le même raisonnement aux paramètres à prendre en compte (dans l'exemple, petal_length, petal_width etc ..)

3. Rétropropagation

```
// Fonction d'erreur
model.compile({
  loss: document.getElementById("fonction_erreur").value,
  optimizer: tf.train.adam(document.getElementById("constante").value),
})
// Entrainement du réseau
model.fit(trainingData, outputData, {
    epochs: document.getElementById("training-number").value
  })
  .then((history) => {

    console.log('entrainement fini!');
    $("#irisForm").toggle();
    $("#entrainement").toggle();
    $("#exemple").toggle();
    $("#results").toggle();
    $("#resultats").toggle();


  })
```

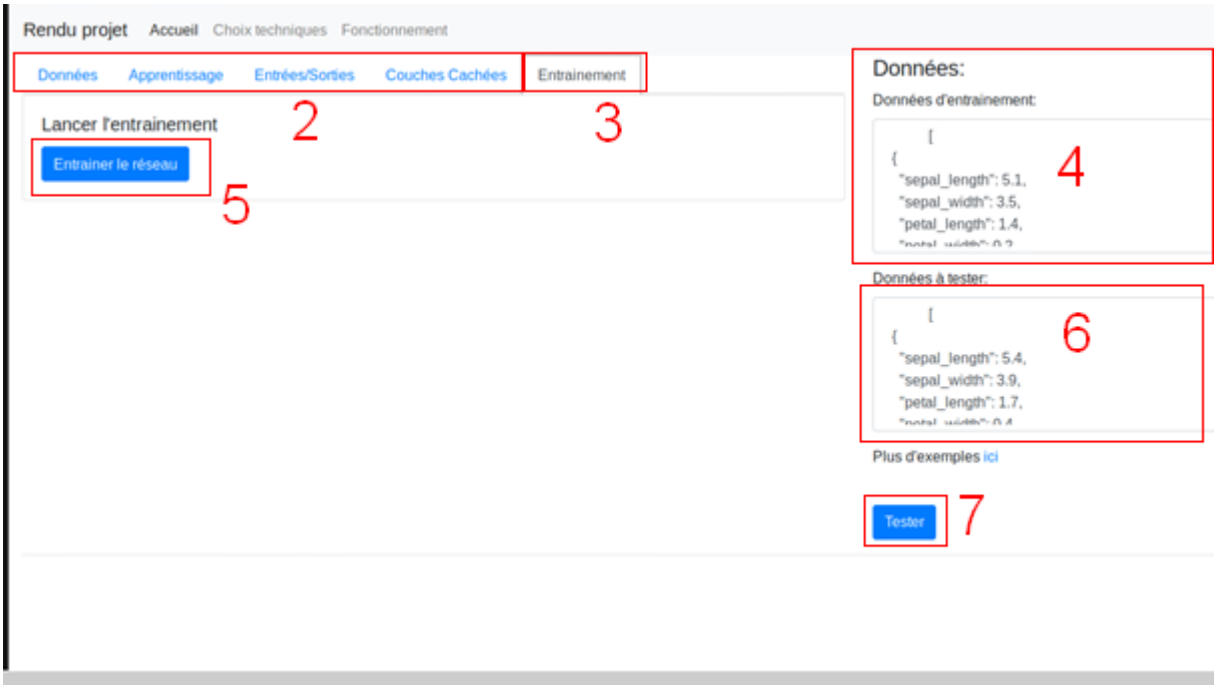
Figure 4 : Rétropropagation

III Les résultats

```
for (var i = 0; i < Object.keys(testingSet).length; i++) {

  compteur++;

  var array = Array.prototype.slice.call(values.slice([i, 0], 1).as1D().dataSync());

  var d = new Date();
  var h = d.getHours();
  var m = d.getMinutes();
  var s = d.getSeconds();

  $("#results").append(
    '<div class="card col-lg-4" id="card_'+ compteur + '">' +
    '<div class="card-body">' +
    '<h5 class="card-title">Test n°' + compteur + '</h5>'
    );

  for(var j = 0; j < outputList.length; j++){
    $("#card_"+compteur).append(
      '<div class="progress">' +
      outputList[j]+
      '<div class="progress-bar bg-success" role="progressbar" style="width: ' + (array[j] * 100) + '%" aria-valuenow="' + (array
      Math.round(array[j] * 100) + '%' +
      '</div>' +
      '</div>'
    );
  }

  $("#card_"+compteur).append(
  '<div class="card-footer">' +
  '<small class="text-muted">Date de mise à jour ' + h + ':' + m + ':' + s + '</small>' +
  '</div>'
  );
```

Figure 5 : affichage des résultats

IV Guide d'utilisation



1. Se rendre sur : http://90.65.175.190/reseau-neurones--projet-I4/
2. Remplir les paramètres du réseau en navigant dans chacun des onglets
3. Aller sur l'onglet « Entrainement »
4. Renseigner un jeu d'entrainement dans « Données »
5. Cliquer sur « Entrainer le réseau »
6. Renseigner un jeu de test dans « Données à tester »
7. Cliquer sur « Tester »

V Annexes

```
// Les entrées
model.add(tf.layers.dense({
  inputShape: [parametres.length],
  activation: fonctionActivation,
  units: parseInt(document.getElementById("couche_entree").value),
}))

for (var j = 1; j <= parseInt(document.getElementById("nb_couches_cachees")); j++) {

  //Pour la première couche cachée
  if (j == 1) {
    model.add(tf.layers.dense({
      inputShape: [parseInt(document.getElementById("couche_entree"))],
      activation: fonctionActivation,
      units: parseInt(document.getElementById("couche_cachee" + j)),
    }))
  }

  //Pour les autres couches cachées
  else if (j > 1) {
    model.add(tf.layers.dense({
      inputShape: [parseInt(document.getElementById("couche_cachee" + (j - 1)))],
      activation: fonctionActivation,
      units: parseInt(document.getElementById("couche_cachee" + j)),
    }))
  }
}

// Couche de sortie
model.add(tf.layers.dense({
  activation: fonctionActivation,
  units: parseInt(document.getElementById("couche_sortie").value),
}))
```

Annexe 1 : génération de toutes les couches du réseau de neurones