MÉMOIRE DE PROJET

PRÉDICTION DE SÉRIES TEMPORELLES PAR RÉSEAUX DE NEURONES

Application à l'équation logistique

Présenté par :

TIANARIMBOLA Andraina

Mention IGGLIA4

N° 39

Mai 2025

SOMMAIRE

---

1. INTRODUCTION

1.1. Contexte et motivations

La prédiction de séries temporelles constitue un défi majeur dans de nombreux domaines tels que la finance, la météorologie, l'épidémiologie ou encore l'analyse de marché. L'enjeu principal consiste à déterminer, à partir de données historiques, l'évolution future d'une grandeur variant dans le temps. Cette tâche est complexifiée par la nature parfois chaotique des systèmes étudiés, où de petites variations dans les conditions initiales peuvent engendrer des comportements radicalement différents à long terme.

Ce projet s'intéresse particulièrement à l'équation logistique, un modèle mathématique simple mais capable de générer des comportements complexes allant de la convergence vers une valeur fixe jusqu'au chaos déterministe. Cette équation, malgré sa simplicité apparente, constitue un excellent terrain d'expérimentation pour les techniques de prédiction de séries temporelles.

1.2. Objectifs du projet

Ce projet poursuit plusieurs objectifs complémentaires :

1. Implémenter un générateur de séries temporelles basé sur l'application logistique, permettant d'illustrer différents comportements dynamiques selon les paramètres choisis.
2. Développer un système de prédiction de ces séries temporelles en utilisant des réseaux de neurones artificiels, capable de prédire les valeurs futures à court terme (prédiction à un pas) et à plus long terme (prédiction multi-pas).
3. Étudier les performances et les limites de l'approche par réseaux de neurones face à des comportements dynamiques de complexité variable.
4. Créer une interface graphique permettant à l'utilisateur de visualiser, manipuler et analyser facilement les séries générées et les prédictions associées.

1.3. Organisation du mémoire

Ce mémoire est structuré en cinq chapitres principaux. Après cette introduction, le deuxième chapitre présente les fondements théoriques nécessaires à la compréhension du projet, notamment les caractéristiques des séries temporelles, les propriétés de l'application logistique et les principes des réseaux de neurones appliqués à la prédiction. Le troisième chapitre détaille la conception et l'implémentation du système, en décrivant l'architecture générale et les différents modules développés. Le quatrième chapitre est consacré à l'analyse des résultats obtenus et à la discussion des performances

du système. Enfin, le cinquième chapitre conclut ce travail en proposant une synthèse et des perspectives d'amélioration.

---

2. FONDEMENTS THÉORIQUES

2.1. Séries temporelles : définition et enjeux

Une série temporelle est une suite de valeurs numériques représentant l'évolution d'une quantité spécifique au cours du temps. Ces données, ordonnées chronologiquement, peuvent être régulièrement ou irrégulièrement espacées et proviennent de domaines très variés : cours de bourse, données météorologiques, signaux électroencéphalographiques, etc.

La prédiction de séries temporelles vise à déterminer les valeurs futures de la série en se basant sur les observations passées. Cette tâche analytique est essentielle dans de nombreux domaines d'application, mais elle présente plusieurs défis majeurs :

- **Non-stationnarité** : Les propriétés statistiques de nombreuses séries temporelles (moyenne, variance) évoluent au cours du temps.
- **Saisonnalité** : Certaines séries présentent des motifs cycliques qu'il faut identifier et exploiter.
- **Bruit** : Les données réelles contiennent souvent une composante aléatoire qui masque la tendance sous-jacente.
- **Comportement chaotique** : Certains systèmes, bien que déterministes, peuvent présenter une sensibilité extrême aux conditions initiales, rendant la prédiction à long terme extrêmement difficile voire impossible.

Dans ce projet, nous nous intéressons particulièrement à ce dernier aspect en étudiant l'application logistique, un système simple mais capable de comportements chaotiques.

2.2. L'application logistique

L'application logistique (ou équation logistique discrète) est une équation aux différences non linéaire qui modélise l'évolution d'une population dans un environnement aux ressources limitées. Elle s'écrit :

où :

- x_n représente la taille normalisée de la population à l'instant n (comprise entre 0 et 1)
- A est un paramètre positif qui caractérise le taux de croissance de la population

Malgré sa formulation simple, cette équation présente une richesse comportementale remarquable selon la valeur du paramètre A :

- Pour $0 < A < 1$: la population s'éteint progressivement (convergence vers 0)

- Pour $1 < A < 3$: la population converge vers une valeur d'équilibre fixe
- Pour $3 < A < 3.57$ (approximativement) : la population oscille entre plusieurs valeurs (bifurcations)
- Pour $A > 3.57$ (approximativement) : comportement chaotique, avec une extrême sensibilité aux conditions initiales

Cette transition vers le chaos, connue sous le nom de "route vers le chaos par doublement de période", fait de l'application logistique un modèle paradigmatique en théorie du chaos. Pour des valeurs comme $A = 4$, le système devient pleinement chaotique, rendant la prédiction à long terme extrêmement difficile malgré la nature déterministe du système.

2.3. Réseaux de neurones pour la prédiction de séries temporelles

Les réseaux de neurones artificiels ont démontré leur efficacité pour modéliser des relations non linéaires complexes dans de nombreux domaines, y compris la prédiction de séries temporelles. Contrairement aux méthodes statistiques traditionnelles comme ARIMA, les réseaux de neurones n'imposent pas d'hypothèses fortes sur la structure sous-jacente des données et peuvent capturer des dépendances non linéaires.

Dans le contexte de la prédiction de séries temporelles, l'approche classique consiste à transformer le problème de prédiction temporelle en un problème d'apprentissage supervisé. Pour ce faire, les données temporelles sont restructurées en paires (entrée, sortie) où :

- L'entrée est une fenêtre glissante de taille fixe (les n dernières observations)
- La sortie est la valeur suivante à prédire

Cette approche, souvent appelée "méthode de la fenêtre glissante" ou "embedding", permet d'utiliser les architectures classiques de réseaux de neurones pour la prédiction de séries temporelles.

Dans notre projet, nous utilisons un perceptron multicouche (MLP) implémenté avec la bibliothèque scikit-learn. Ce type de réseau est constitué de :

- Une couche d'entrée dont la taille correspond au nombre de valeurs passées considérées (paramètre "look_back")
- Une ou plusieurs couches cachées avec des neurones utilisant des fonctions d'activation non linéaires (ReLU dans notre cas)
- Une couche de sortie avec un seul neurone pour prédire la valeur suivante

L'apprentissage du réseau s'effectue en minimisant l'erreur quadratique moyenne entre les prédictions et les valeurs réelles, à l'aide de l'algorithme d'optimisation Adam, une variante efficace de la descente de gradient stochastique.

---

3. CONCEPTION ET IMPLÉMENTATION

3.1. Architecture générale du système

Le système développé dans ce projet est structuré autour de deux classes principales :

1. **LogisticMapPredictor** : Classe responsable de la génération des séries temporelles basées sur l'application logistique et de leur prédiction à l'aide de réseaux de neurones.
2. **LogisticMapApp** : Classe qui implémente l'interface graphique utilisateur, permettant d'interagir avec les fonctionnalités du prédicteur de manière intuitive.

Cette architecture modulaire permet une séparation claire entre la logique métier (génération et prédiction des séries) et l'interface utilisateur, facilitant ainsi la maintenance et l'évolution du code.

Le système utilise plusieurs bibliothèques Python pour ses différentes fonctionnalités :

- **NumPy** pour les calculs numériques efficaces
- **scikit-learn** pour l'implémentation des réseaux de neurones
- **Matplotlib** pour la visualisation des séries temporelles et des prédictions
- **Tkinter** pour l'interface graphique

3.2. Module de génération de séries temporelles

La génération des séries temporelles s'appuie sur l'application logistique, dont le comportement peut être ajusté via deux paramètres principaux :

- Le **paramètre A** qui détermine la nature du comportement dynamique (convergence, oscillations, chaos)
- La **valeur initiale x0** qui sert de point de départ à l'itération

La méthode `generate_logistic_map` de la classe `LogisticMapPredictor` implémente cette génération en calculant itérativement les valeurs successives de la série selon l'équation :

```python
x[i] = self.A * x[i-1] * (1 - x[i-1])
```

Pour éviter les problèmes numériques, notamment dans le cas de comportements chaotiques, des précautions sont prises pour maintenir les valeurs dans l'intervalle [0,1] à l'aide de la fonction `np.clip`.

L'utilisateur peut choisir entre deux valeurs prédéfinies pour le paramètre A :

- A = 2.0 : comportement convergent vers une valeur fixe
- A = 4.2 : comportement chaotique

Ces choix permettent d'illustrer les différents régimes dynamiques de l'application logistique et d'évaluer les performances du prédicteur dans des contextes de complexité variable.

3.3. Module de prédiction par réseau de neurones

La prédiction des séries temporelles est réalisée par un perceptron multicouche implémenté via la classe `MLPRegressor` de scikit-learn. Le processus complet comprend plusieurs étapes :

1. **Préparation des données** :
   - Les données de la série temporelle sont d'abord normalisées à l'aide d'un `MinMaxScaler` pour les ramener dans l'intervalle [0,1], optimisant ainsi l'apprentissage du réseau.
   - Ensuite, les données sont restructurées en séquences d'entrée-sortie à l'aide de la méthode `create_dataset`, qui implémente l'approche de la fenêtre glissante mentionnée précédemment.
2. **Configuration et entraînement du réseau** :
   - L'architecture du réseau peut être configurée via le paramètre `hidden_layers`, qui définit le nombre de neurones dans chaque couche cachée.
   - L'entraînement est réalisé en minimisant l'erreur quadratique moyenne entre les prédictions et les valeurs réelles, à l'aide de l'algorithme d'optimisation Adam.
3. **Prédiction** :
   - Deux modes de prédiction sont implémentés :
     - **Prédiction à un pas** (`predict_one_step`) : prédit la valeur suivante à partir des valeurs passées réelles.
     - **Prédiction multi-pas** (`predict_multi_step`) : réalise des prédictions sur plusieurs pas de temps en utilisant les valeurs prédites comme entrées pour les prédictions suivantes.
4. **Évaluation des performances** :
   - Les performances du modèle sont évaluées en comparant les valeurs prédites aux valeurs réelles, notamment à l'aide de métriques comme l'erreur quadratique moyenne (MSE) et sa racine carrée (RMSE).

Le système inclut également des fonctionnalités de sauvegarde et de chargement des modèles entraînés, permettant de réutiliser un modèle sans avoir à le réentraîner.

3.4. Interface utilisateur graphique

L'interface graphique, implémentée avec Tkinter, est organisée en trois onglets correspondant aux principales fonctionnalités du système :

1. **Onglet "Génération"** :
   - Permet de configurer les paramètres de l'application logistique (A, x0, nombre de pas)
   - Affiche la série temporelle générée sous forme de graphique
   - Pour le cas chaotique (A = 4.2), un sous-graphique montre les 100 premières valeurs pour mieux visualiser le comportement initial
2. **Onglet "Entraînement"** :

- Permet de configurer les paramètres du réseau de neurones (nombre de valeurs passées, architecture, nombre d'itérations)
- Affiche les résultats de l'entraînement, notamment l'erreur quadratique moyenne obtenue

3. **Onglet "Prédiction"** :
   - Permet de choisir le type de prédiction (un pas ou multi-pas) et le nombre de prédictions à réaliser
   - Pour la prédiction multi-pas, permet de spécifier le nombre de pas en avant (3, 10 ou 20)
   - Affiche les résultats des prédictions sous forme de graphique, avec les valeurs réelles et prédites

L'interface inclut également une barre de menu permettant d'accéder à des fonctionnalités supplémentaires, notamment la sauvegarde et le chargement des modèles entraînés.

Des fonctionnalités de visualisation avancées ont été implémentées pour faciliter l'analyse des résultats :

- Affichage des statistiques descriptives des séries générées (minimum, maximum, moyenne)
- Annotation des graphiques avec des informations sur le comportement dynamique
- Visualisation des erreurs de prédiction
- Utilisation de couleurs distinctes pour différencier les séquences de prédiction

---

4. RÉSULTATS ET ANALYSES

4.1. Analyse du comportement de l'application logistique

Les expérimentations menées avec différentes valeurs du paramètre A confirment la richesse comportementale de l'application logistique :

Pour A = 2.0 :
La série temporelle converge rapidement vers une valeur fixe (environ 0.5), indépendamment de la valeur initiale x0 (tant qu'elle est strictement comprise entre 0 et 1). Ce comportement stable et prévisible constitue un cas idéal pour la prédiction.

Pour A = 4.2 :
La série présente un comportement chaotique caractérisé par des fluctuations apparemment aléatoires, sans motif discernable à long terme. L'analyse du graphique révèle une sensibilité extrême aux conditions initiales : deux séries générées avec des valeurs de x0 très proches divergent rapidement.

Ces observations correspondent aux attentes théoriques concernant la dynamique de l'application logistique et fournissent un excellent banc d'essai pour évaluer les capacités et limites des réseaux de neurones en matière de prédiction.

4.2. Performance des prédictions à un pas

Les expériences de prédiction à un pas montrent des résultats contrastés selon le régime dynamique de la série :

Pour le régime convergent (A = 2.0) :

- Erreur quadratique moyenne (MSE) très faible (typiquement de l'ordre de 10^-6)
- Prédictions visuellement indiscernables des valeurs réelles
- Performance stable indépendamment de l'architecture précise du réseau

Ces excellents résultats s'expliquent par la simplicité du comportement à modéliser : après la phase transitoire initiale, la série est pratiquement constante, facilitant grandement la tâche de prédiction.

Pour le régime chaotique (A = 4.2) :

- MSE significativement plus élevée (typiquement de l'ordre de 10^-3 à 10^-2)
- Certaines prédictions présentent des écarts importants par rapport aux valeurs réelles
- Performance variable selon l'architecture du réseau et les hyperparamètres choisis

Malgré ces difficultés, les prédictions à un pas restent remarquablement précises pour un système chaotique, démontrant la capacité du réseau à capturer les dépendances à court terme, même dans un contexte dynamique complexe.

L'analyse des erreurs de prédiction en fonction de la position dans la série ne révèle pas de tendance claire, suggérant que les difficultés de prédiction sont intrinsèques à la nature chaotique du système plutôt qu'à des limitations spécifiques du modèle.

4.3. Performance des prédictions multi-pas

Les prédictions multi-pas constituent un défi nettement plus important, particulièrement dans le contexte des systèmes chaotiques. Les résultats obtenus confirment cette difficulté :

Pour le régime convergent (A = 2.0) :

- Prédictions précises même à long terme (20 pas)
- Erreur pratiquement constante indépendamment de l'horizon de prédiction
- RMSE typiquement inférieure à 10^-3, même pour des prédictions à 20 pas

Pour le régime chaotique (A = 4.2) :

- Dégradation rapide de la précision avec l'augmentation de l'horizon de prédiction
- Pour les prédictions à court terme (3 pas), RMSE de l'ordre de 10^-2
- Pour les prédictions à moyen terme (10 pas), RMSE de l'ordre de 10^-1
- Pour les prédictions à long terme (20 pas), les prédictions deviennent essentiellement aléatoires

Cette détérioration exponentielle de la précision avec l'horizon de prédiction est une caractéristique fondamentale des systèmes chaotiques. Elle illustre le concept d'"horizon de prédictibilité", au-delà duquel les prédictions déterministes deviennent impossibles en pratique, même avec un modèle parfait du système.

4.4. Limites et défis rencontrés

Plusieurs défis et limitations ont été identifiés au cours de ce projet :

1. **Sensibilité aux hyperparamètres** : Les performances du réseau de neurones dépendent fortement de choix comme la taille de la fenêtre d'observation (look_back) et l'architecture du réseau. Un équilibrage délicat est nécessaire entre la complexité du modèle et le risque de surapprentissage.
2. **Limites intrinsèques à la prédiction de systèmes chaotiques** : Conformément à la théorie du chaos, nos expériences confirment l'impossibilité pratique de prédictions fiables à long terme pour le régime chaotique, quelle que soit la sophistication du modèle utilisé.
3. **Considérations de mise à l'échelle** : Pour des séries temporelles très longues ou des architectures de réseau très complexes, des problèmes de performance (temps de calcul, consommation mémoire) peuvent apparaître. Notre implémentation actuelle n'est pas optimisée pour le traitement de données massives.
4. **Interface utilisateur** : Bien que fonctionnelle, l'interface graphique présente certaines limitations en termes de réactivité et d'ergonomie, particulièrement lors du traitement de séries longues ou de l'entraînement de réseaux complexes.

Ces limitations ouvrent des pistes d'amélioration pour de futures versions du système.

---

5. CONCLUSION ET PERSPECTIVES

5.1. Synthèse des résultats

Ce projet a permis de développer un système complet de génération et de prédiction de séries temporelles basées sur l'application logistique. Les principales réalisations sont :

1. L'implémentation d'un générateur paramétrable de séries temporelles issues de l'application logistique, capable de produire des comportements dynamiques variés, du régime convergent au régime chaotique.
2. Le développement d'un prédicteur basé sur des réseaux de neurones, capable de réaliser des prédictions à un pas et multi-pas avec des performances variables selon le régime dynamique considéré.
3. La création d'une interface graphique intuitive permettant d'explorer facilement les différents aspects du système, de la génération des séries à l'analyse des prédictions.

L'analyse des résultats a permis de confirmer plusieurs aspects théoriques importants :

- La capacité des réseaux de neurones à capturer efficacement des dynamiques non linéaires
- Les limites fondamentales de la prédiction à long terme pour des systèmes chaotiques
- L'importance du choix des hyperparamètres dans les performances du modèle

5.2. Améliorations possibles

Plusieurs pistes d'amélioration peuvent être envisagées pour de futures versions de ce système :

1. **Architectures de réseau avancées** : L'utilisation de réseaux récurrents (LSTM, GRU) ou de réseaux convolutifs pourrait potentiellement améliorer les performances de prédiction, particulièrement pour les régimes dynamiques complexes.
2. **Optimisation des hyperparamètres** : L'implémentation d'une recherche automatique des hyperparamètres optimaux (via des techniques comme la validation croisée ou des approches bayésiennes) pourrait faciliter l'utilisation du système et améliorer ses performances.
3. **Analyse de sensibilité** : Une étude plus systématique de l'impact des différents paramètres (A, x0, architecture du réseau) sur les performances de prédiction permettrait de mieux comprendre les forces et faiblesses de l'approche.
4. **Visualisations avancées** : L'ajout de visualisations complémentaires (diagrammes de bifurcation, exposants de Lyapunov, etc.) enrichirait l'analyse des séries temporelles générées.
5. **Interface utilisateur améliorée** : L'interface pourrait être enrichie pour permettre une exploration plus interactive des données et des résultats, avec notamment la possibilité de zoomer sur certaines portions des graphiques ou de comparer visuellement différentes configurations.

5.3. Applications potentielles

Bien que ce projet se concentre sur l'application logistique comme cas d'étude, l'approche développée pourrait être adaptée à diverses applications pratiques :

1. **Prévision financière** : Prédiction à court terme de séries temporelles financières (cours boursiers, taux de change).
2. **Météorologie** : Prédiction de variables météorologiques (température, précipitations) à partir de séries historiques.
3. **Consommation énergétique** : Anticipation de la demande électrique pour optimiser la production et la distribution d'énergie.
4. **Systèmes dynamiques industriels** : Monitoring et prédiction du comportement de processus industriels complexes pour la maintenance prédictive.

Pour ces applications réelles, des adaptations seraient nécessaires, notamment pour gérer le bruit inhérent aux données expérimentales, les valeurs manquantes, ou encore la présence de tendances et de saisonnalités.

En conclusion, ce projet constitue une base solide pour l'exploration des techniques de prédiction de séries temporelles non linéaires par réseaux de neurones, avec un accent particulier sur les défis spécifiques posés par les systèmes chaotiques.

---

6. BIBLIOGRAPHIE

1. Strogatz, S. H. (2018). Nonlinear dynamics and chaos: With applications to physics, biology, chemistry, and engineering. CRC Press.
2. May, R. M. (1976). Simple mathematical models with very complicated dynamics. Nature, 261(5560), 459-467.
3. Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, 159-175.
4. Brownlee, J. (2018). Deep Learning for Time Series Forecasting. Machine Learning Mastery.
5. Giles, C. L., Lawrence, S., & Tsoi, A. C. (2001). Noisy time series prediction using recurrent neural networks and grammatical inference. Machine Learning, 44(1), 161-183.
6. Pandas Development Team. (2024). pandas-dev/pandas: Pandas. Zenodo. https://doi.org/10.5281/zenodo.3509134
7. Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90-95.
8. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
9. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.
10. Tkinter - Python interface to Tcl/Tk. (2024). Python Software Foundation. https://docs.python.org/3/library/tkinter.html

---

7. ANNEXES

7.1. Code source principal

Le code source complet du projet est disponible dans le fichier `logistic_map_prediction.py`. Les principales classes implémentées sont :

- `LogisticMapPredictor` : responsable de la génération des séries temporelles et de leur prédiction
- `LogisticMapApp` : implémente l'interface graphique utilisateur

7.2. Guide d'utilisation de l'application

Installation des dépendances :

```
pip install numpy matplotlib scikit-learn
```

Lancement de l'application :

```
python logistic_map_prediction.py
```

Utilisation de l'application :

1. Onglet "Génération" :
   - Sélectionnez le paramètre A (2.0 ou 4.2)
   - Définissez la valeur initiale x0 (entre 0 et 1)
   - Spécifiez le nombre de pas à générer
   - Cliquez sur "Générer la série"
2. Onglet "Entraînement" :
   - Configurez les paramètres du réseau (nombre de valeurs passées, architecture)
   - Cliquez sur "Entraîner le modèle"
   - Consultez les résultats d'entraînement affichés
3. Onglet "Prédiction" :
   - Choisissez le type de prédiction (un pas ou multi-pas)
   - Pour la prédiction multi-pas, spécifiez le nombre de pas en avant
   - Définissez le nombre de prédictions à réaliser
   - Cliquez sur "Effectuer les prédictions"
   - Analysez les résultats graphiques
4. Menu "Fichier" :
   - Utilisez "Sauvegarder le modèle" pour conserver un modèle entraîné
   - Utilisez "Charger un modèle" pour réutiliser un modèle préalablement sauvegardé