Sujets de mini-projets

Sujet I: Décodage neuronal

Assistant responsable: Samuel.Gmehlin@epfl.ch et Lukas.Rytz@epfl.ch

Motivation

Certains neurones du cerveau sont corrélés avec des stimuli complexes, par exemple visuels. L'objet de ce projet est de construire un neurone artificiel de ce type, capable de décoder un signal bruité issu de la rétine.

Formalisation

La présence ou l'absence d'un objet dans le champ visuel donne lieu à un certain stimulus. On notera ce stimuli ζ, la valeur $\zeta = 0$ correspondant à l'absence de l'objet, $\zeta = 1$ à sa présence. La probabilité que l'objet soit présent est de $p = \frac{1}{2}$. Ce stimulus ζ lié à la perception de l'image est modélisé sur la rétine par un signal composé de 20 bits $x_1, \ldots, x_{20}$, correspondant à l'activation de 20 neurones à la sortie de la rétine. La présence et l'absence de l'objet produisent deux motifs différents:

$$\begin{aligned} \zeta = 0 \quad &\rightarrow \quad (x_1, \ldots, x_{20}) = 11111000000000011111 \\ \zeta = 1 \quad &\rightarrow \quad (x_1, \ldots, x_{20}) = 11111000001111100000 \end{aligned}$$

Ces motifs sont ensuite transmis par l'intérmédiaire du nerf optique jusqu'au cortex visuel. Cette transmission est bruitée. On notera $y_1, \ldots, y_{12}$ les activations binaires des 12 neurones du cortex visuel. Chaque y_i a une probabilité $p = 0.9$ d'être identique au x_i, une probabilité $p = 0.1$ d'être inversé. Ce bruit modélise les erreurs de transmission dans le nerf optique.

L' activation de ces 20 neurones du cortex visuel est ensuite intégrée par un neurone dont l'activation témoigne du stimulus initial.

Le comportement de ce dernier neurone est modulé par ses poids synaptiques $\omega_1, \ldots, \omega_{20}$. Son activation ξ est égale à 1 si $\sum_i \omega_i \, y_i \geq 0$, à 0 sinon.

On cherchera à sélectionner des poids synaptiques adéquats pour que la sortie du

neurone ξ soit corrélée avec le stimulus initial ζ.

$$\begin{array}{ccc} \zeta & \overset{\text{IM ?}}{\dashrightarrow} & \xi \\ \downarrow & & \uparrow \\ x_1, \ldots, x_{20} & \longrightarrow & y_1, \ldots, y_{20} \end{array}$$

Structure du programme

Les stimuli avant bruitage (x_i) et après bruitage (y_i) peuvent être représentés à l'aide de tableaux d'entiers contenant des 0 et des 1. Les poids synaptiques ω_i peuvent être représentés sous la forme d'un tableau de 20 valeurs flottantes.

1. Écrire une routine qui reoit en paramètre la valeur de ζ et remplit un tableau de 20 entiers avec les valeurs de $x_1, \ldots, x_{20}$ correspondant au motif.
2. Écrire une routine qui reoit en paramètre les valeurs des x_i et remplit un second tableau avec les valeurs des y_i.
3. Écrire une routine qui remplit un troisième tableau avec les valeurs des poids synaptiques $\omega_1, \ldots, \omega_{20}$, valeurs flottantes aléatoirement réparties entre -15 et 15.
4. Écrire une routine qui reoit en paramètre les y_i et les ω_i et retourne la valeur binaire ξ de l'activation du neurone.
5. Écrire une routine qui reoit en paramètre les tableaux contenant des valeurs de ζ et de ξ et calcule l'information mutuelle entre les deux variables.
6. Écrire le programme complet qui génère aléatoirement 10000 neurones (c.a.d. 10 000 tableau de poids synaptiques) et estime pour chacun d'eux l'information mutuelle entre ζ et ξ, suite à une simulation sur $1,000$ stimuli. Le programme devra conserver les valeurs des poids synaptiques du meilleur neurone.

Rapport

Le rapport devra faire entre 2 et 4 pages.

Il devra présenter un graphique pour la distribution des scores d'information mutuelle des 10000 neurones générés et donner les valeurs des poids synaptiques du meilleur neurone.

Une courte analyse devra expliquer la cohérence de ces poids.

Le code source du programme, correctement indenté, sera joint en annexe.

Sujet II: Optimalité du code génétique

Assistants responsables: Stefano.Pennese@epfl.ch et Khaled.Ouafi@epfl.ch

Motivation

Ce projet à pour but d'étudier l'optimalité du code génétique, c'est-à-dire la correspondance des triplets de bases azotées (AUCG) et des acides aminés. On considérera d'une part le code "naturel", d'autre part un code artificiel et on comparera leur performance lors de la dégradation des triplets de bases azotées, dégradation modélisant grossièrement l'effet des radiations.

Formalisation

Il existe 4 bases azotées différentes, autorisant un total de 64 triplets (43). Le code génétique est une application qui associe un des 20 acides aminés à chacun de ces 64 triplets. Le code biologique est donné ci-dessous.

On notera $t_1, \ldots, t_N$ la série considérée de triplets avant dégradation et $t'_1, \ldots, t'_N$ la série de triplets après dégradation. On notera $a_1, \ldots, a_N$ les acides aminés correspondant aux triplets avant dégradation et $a'_1, \ldots, a'_N$ les acides aminés correspondant aux triplets dégradés. On peut voir la transformation $a \rightarrow a'$ comme un canal bruité.

On estime la qualité d'un code à l'aide de l'information mutuelle entre l'entrée et la sortie du canal, entre a et a', soit entre l'acide aminé correspondant à un triplet tiré au hasard et l'acide aminé produit par le même triplet dégradé par des radiations.

$$\begin{array}{ccc} t & \longrightarrow & t' \\ \downarrow & & \downarrow \\ a & \overset{\text{IM ?}}{\dashrightarrow} & a' \end{array}$$

Lors de la dégradation d'un triplet, il existe une probabilité $p = 1 - \epsilon$ qu'un codon (3 bases) reste inchangé. La troisième base a une probabilité 2 fois plus importante d'être mutée que les première et deuxième bases. Par exemple, la troisième base aura une probablit $p = \epsilon/2$ soit remplacée par l'une des trois autres bases possibles, et modifie ainsi l'information du codon. On prendra par exemple $\epsilon = 0.06$.

UUU	→	Phe	UCU	→	Ser	UAU	→	Tyr	UGU	→	Cys
UUC	→	Phe	UCC	→	Ser	UAC	→	Tyr	UGC	→	Cys
UUA	→	Leu	UCA	→	Ser	UAA	→	*STOP*	UGA	→	*STOP*
UUG	→	Leu	UCG	→	Ser	UAG	→	*STOP*	UGG	→	Trp
CUU	→	Leu	CCU	→	Pro	CAU	→	His	CGU	→	Arg
CUC	→	Leu	CCC	→	Pro	CAC	→	His	CGC	→	Arg
CUA	→	Leu	CCA	→	Pro	CAA	→	Gln	CGA	→	Arg
CUG	→	Leu	CCG	→	Pro	CAG	→	Gln	CGG	→	Arg
AUU	→	Ile	ACU	→	Thr	AAU	→	Asn	AGU	→	Ser
AUC	→	Ile	ACC	→	Thr	AAC	→	Asn	AGC	→	Ser
AUA	→	Ile	ACA	→	Thr	AAA	→	Lys	AGA	→	Arg
AUG	→	Met	ACG	→	Thr	AAG	→	Lys	AGG	→	Arg
GUU	→	Val	GCU	→	Ala	GAU	→	Asp	GGU	→	Gly
GUC	→	Val	GCC	→	Ala	GAC	→	Asp	GGC	→	Gly
GUA	→	Val	GCA	→	Ala	GAA	→	Glu	GGA	→	Gly
GUG	→	Val	GCG	→	Ala	GAG	→	Glu	GGG	→	Gly

Table 1: Code génétique

Programme

On représentera les acides aminés tel que présenté (table 1) sous la forme d'un tableau de 64 entiers compris entre 0 et 20 (par exemple, Phe aura le numéro 1 et Pro le numéro 2).

1. Il faut écrire une fonction qui génère un nombre aléatoire en 0 et 63. Par la suite, cette fonction sera utilisée pour tirer un des 64 triplets au hasard. Le mieux : écrire une fonction générique, qui génère une valeur aléatoire entre 0 et N, o N est passé en paramètre. Comme ça, vous pourrez la réutiliser dans la partie 3, avec des paramètres differents.

2. Remplir 2 tableaux : un tableau d'entiers qui servira de code génétique naturel (donné en annexe), c'est à dire que chaque entrée sera un nombre de 0 à 20. Un autre tableau de char doit être créé, qui correspondra à la suite de triplets (UUU, UUA, etc.). Ces 2 tableaux doivent avoir les mêmes tailles, afin d'avoir une correspondance entre les 2 par le biais des indices.

3. Ici, il faut écrire une routine qui prend en paramètre un pointeur vers un tableau (afin de ne pas créer par la suite 1000 tableaux), et qui le remplit de valeurs entières aléatoires (entre 0 et 20). Un code artificiel est en fait une

nouvelle table de correspondance des acides aminés en fonction des triplets de ARN. Cela correspond à la version aléatoire du tableau naturel.

4. Ecrire une routine qui reoit en paramètre un entier représentant un triplet non-dégradé et la valeur de ϵ, le transforme en chaîne de caracères, dégrade le triplet en fonction de la valeur de ϵ, et retourne un entier entre 0 et 63 correspondant au triplet après dégradation. Cette routine doit être écrite de faon à être rapide car elle va être appelée un grand nombre de fois.

5. Vous devez ici écrire une fonction qui prend en paramètre un code (c'est à dire un tableau avec un code triplets/acides aminés), générer un tableau de 100000 entiers de 0 à 63 correspondant à des triplets, puis pour chaque entier ainsi dans le tableau, le passer en paramètre à la fonction degradation, et placer le nouvel entier retourné par cette fonction dans le tableau t', les uns à la suite des autres (ATTENTION : t et t' doivent être de la même taille, et les codons dégradés et non-dégradés doivent être parfaitement alignés). Ensuite, à l'aide du code passé en paramètre, créer un tableau "a" qui contient les 100000 acides aminés générés par ledit code, puis dans a' les 100000 acides aminés générés par le même code, mais à partir des triplets dégradés t'

6. Ecrire une fonction qui recoit deux tableaux de valeurs, a et a', et calcule l'information mutuelle entre les deux. Il faut tout d'abord créer un tableau de probabilités qui sera passé en paramètre de la fonction mutual-information-couple (donnée en annexe). Dans notre programme, la taille du tableau à 2 dimensions est de 21 sur 21, les aa non-dégradés correpondant aux lignes du tableau, et les aa dégradés aux colonnes.

7. Ecrire le programme complet qui calcule l'information mutuelle pour le code naturel présenté (table 1) et les informations mutuelles pour 1000 codes générés aléatoirement.

Rapport

Le rapport devra faire entre 2 et 4 pages.

Il devra présenter la distribution des informations mutuelles des 1000 codes artificiels, ceci pour deux valeurs différentes de ϵ (0.06 et 0.12) et indiquer la valeur de l'information mutuelle pour le code naturel. Une demi-page de discussion devra conclure sur la qualité du code génétique naturel.

Sujet III: Alignement de séquences de protéines

Assistants responsables: Loana.Chatelain@epfl.ch et Julien.Hamilton@epflc.h

Motivation

Certaines séquences dans l'ADN constituent des gènes. Les gènes codent pour des protéines. Ils sont tout d'abord transcrits en ARN, puis traduits en protéines. Il y a donc passage d'une structure sous forme de suite de triplets de bases azotées (codons) à une suite d'acides aminés. Il existe 20 acides aminés différents. Chaque acide aminé est identifié par un caractère de l'alphabet, par exemple G pour Glycine et F pour Phenylalanine. Les 20 caractères utilisés sont:

`G, A, L, M, F, W, K, S, N, D, P, V, I, C, Y, H, R, T, Q, E`

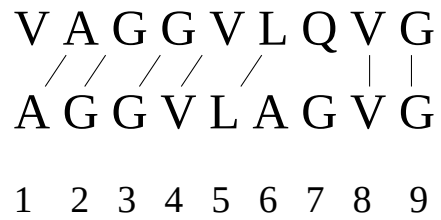
Une protéine est composée d'une séquence d'acides aminés, soit d'une séquence de ces caractères.

Les protéines avec séquences similaires ont souvent une structure tri-dimensionelle similaire et quelquefois même une fonction similaire. Il est donc intéressant de pouvoir quantifier la similitude entre deux séquences données.

Formalisation

Deux séquences peuvent être parfaitement alignées ou présenter des "trous".

Pour évaluer la qualité de l'alignement, on utilise un score égal au nombre de correspondances entre les deux séquences. Par exemple, sur le schéma on peut compter 7 correspondances, ce qui donne un score de 7.



Le score maximal, le nombre maximal de correspondances, est égal au nombre N d'acides aminés qui constituent la séquence, soit 9 dans notre exemple.

Le but est de maximiser le nombre de correspondances en alignant les séquences d'une façon optimale et de s'intéresser à la complexité de l'algorithme en fonction du nombre N d'acides aminés.

Programme

- Implémenter l'algorithme de Needleman-Wunsch ou "Programmation Dynamique" vu au cours.
- Ecrire une routine qui prend comme argument deux chaînes de caractères et construit un tableau à deux dimensions. Nombre de ligne= nombre de caractère chaîne 1; et nombre de colonne= nombre de caractères chaîne 2.
- Ecrire une routine qui remplit le tableau des "match".
- Ecrire une routine prenant en paramètre le numéro de colonne et le numéro de ligne et qui retourne la valeur maximum du sous tableau associé.
- Ecrire une routine qui recalcule la valeur des cases du tableau
- Ecrire la routine qui trouve l'alignement ayant le score maximum.
- Comme premier test, appliquer votre algorithme à la paire de séquences suivantes:

  `SSSGALMPPTLSNDFDDFGMOTQTEEGCM`
  `DTGALMPPTLSNDFDDFGMPCYHDTCMPPP`

  Question (A): Quelle est le score de correspondance maximal ? A quel alignement correspond-il?

Pour un test systématique des performances de l'algorithme, les assistants vous donneront 20 séquences différentes. Toutes les séquences sont similaires, (en fait il contiennent tous le même gène hypothétique), mais ils ont une longueur différentes.

Évaluation de la performance:

- **(B)** Appliquez votre algorithme et notez le score pour tous les pairs possibles en fonction de la longueur N_1, N_2 des 2 séquences.
- **(C)**Tracer le graphe du temps de calcul nécessaire à votre algorithme comme fonction de $N_1 * N_2$.
- Interpréter les résultats.

Rapport

Le rapport devra faire entre 2 et 4 pages.

Il contiendra la réponse à la question (A) et les graphiques et résultats des points (B) et (C) avec leurs commentaires.

Annexe

La routine `mutual_information_couple` ci-dessous calcule $I(X, Y)$. Le tableau `double proba[x][y]` représente la distribution de probabilité d'un couple de valeurs (i.e. $P(X = x, Y = y)$).

Si vous utilisez `gcc` sous GNU/Linux, n'oubliez pas de compiler avec `-lm` pour faire le lien avec la librairie mathémathique.

```
#include <math.h>

inline double xlogx(double x) {
  if(x == 0) return 0.0;
  else return x*log(x)/log(2);
}

double mutual_information_couple(int nb_values, double proba[x][y]) {
  double h_xy = 0.0, h_x = 0.0, h_y = 0.0;
  int i, j;
  for(i = 0; i < nb_values; i++) {
    double p_x = 0.0, p_y = 0.0;
    for(j = 0; j < nb_values; j++) {
      p_x += proba[i][j];
      p_y += proba[j][i];
      h_xy -= xlogx(proba[i][j]);
    }
    h_x -= xlogx(p_x);
    h_y -= xlogx(p_y);
  }
  return h_x + h_y - h_xy;
}
```