Redes neuronales multicapa

Castiglione, Karpovsky, Sturla

Sistemas de Inteligencia Artificial

3 de Mayo de 2012

Tabla de contenidos

El problema

El problema consistió en analizar el comportamiento de una Red de hopfield como memoria asociativa direccionable por el contenido.

- Red de Hopfield implementada en el lenguaje *Java*.

Modelado del problema

Algunas definiciones preliminares:

- Ψ: Conjunto de patrones a memorizar
- N: Longitud de los patrones de entrada (ej. $N = 64$ si cada imagen es de $8x8$ pixels)
- p: Cantidad de patrones distintos que contiene Ψ

Se representó a la Red de Hopfield como una clase abstracta debido a la distinción entre el algoritmo **sincrónico** y el algoritmo **asincrónico**.

Modelado del problema

La clase contiene dos variables: un vector y una matriz que están definidos como sigue:

- **float[N][N] weights**: Matriz de pesos w_{ij} correspondientes a los patrones que se desean memorizar.
- **int[N] states**: Vector con los estados de cada una de las N neuronas.

Nótese que si bien el vector de estados es de tipo entero, los estados definidos para este problema son 1 o -1 dado que las unidades que están siendo utilizadas son **bipolares**.

Modelado del problema

- Representación de μ

Representar a los patrones μ como vectores de enteros **int[] pattern** los cuales contienen un 1 en la posición i en el caso de que el i-ésimo pixel del patrón sea negro o contienen un -1 en la posición i en el caso de que el i-ésimo pixel del patrón sea blanco.

- Los patrones fueron *normalizados* (escala de grises a B&W)

Tabla de contenidos

Inicialización de la red

El método *storePatterns* inicializa la matriz de pesos sinápticos de la red utilizando la regla del producto externo de **Hebb**.
Nótese que la matriz de pesos, una vez inicializada, no cambia durante toda la ejecución del algoritmo.

- **Simetría:** La matriz de pesos es simétrica con $w_{ij} = 0$
- **Matriz de pesos:** Una vez inicializada queda estática.

Función de activación

La función de activación utizada es la función signo: $\mathbf{f(x) = x}$.

Actualización de estados

Regla de actualización

$$S_i(n+1) = sgn\left(\sum_{j=1}^{N} W_{ij} S_j(n)\right)$$

La convergencia está dada cuando el vector de estados permanece invariante respecto del vector en el paso anterior o se detecte un **ciclo de longitud dos** en las actualizaciones del vector de estados.

Red sincrónica: Modelo de Little

Se implementó una red neuronal con el modelo de **Little** en la cual la actualización de los estados es sincrónica.

Actualización de estados

- Actualizar todas las neuronas simultaneamente en cada paso.

Red asincrónica: Modelo de Hopfield

Se implementó una red neuronal con el modelo de **Hopfield** en la cual la actualización de los estados es asincrónica.

Actualización de estados

- En cada paso se actualiza de a una unidad por vez, la misma es elegida al azar.

Tabla de contenidos

Representación de la red neuronal

Se representó la red neuronal como una matriz de pesos.

- Cada neurona es una columna de pesos.
- Cada capa de neuronas es una matriz de pesos.
- La red neuronal, por consiguiente, es un vector de matrices.

Se utilizaron dos funciones de activación distintas:

Sigmoidea exponencial

$$g(h) = \frac{1}{1 + e^{-2\beta h}}$$

Derivada:

$$2\beta g(1 - g)$$

Tangente hiperbólica

$$g(x) = tanh(x)$$

Derivada:

$$\beta g(1 - g^2)$$

Arquitecturas estudiadas

- Perceptron simple?
- Pocas neuronas, pocas capas
- Muchas neuronas, muchas capas
- Punto intermedio
- Conexiones muertas/rotas

Conjunto de entrenamiento y testeo

Se decidió seguir el consejo de la cátedra y al realizar las pruebas se utilizó un subconjunto de los datos seleccionados al azar para la fase de aprendizaje y el subconjunto restante para testeo.

- Elección de puntos al azar? Puntos representativos?

Tabla de contenidos

Eta dinámico (η adaptativo)

- Si el error sube consistenemente incrementar eta aritméticamente: quizás se está siendo muy conservador.
- Si el error incrementa, reducir η exponencialmente.

Momentum

$$w_{ij}(t+1) = -\frac{\partial E}{\partial wij} + \alpha w(t)$$

Cambios dependen de cambios anteriores. Idea de dirección general del error.
Olvido exponencial de cambios anteriores.
Se aplica a cada batch / lote

Ruido

Idea: Escape del mínimo local.
Robustez de la red neuronal: debería poder soportar ruido.

Tabla de contenidos

Tabla de contenidos

Conclusión

- Incrementar la cantidad de neuronas arbitrariamente no necesariamente implica mejoras en cuanto al error (puede llevar a malas generalizaciones y tiempo de más hasta alcanzar el error desdeado).
- No existe tal cosa como una mejor arquitectura o parámetros óptimos. Estos seguramente dependan el problema que se está analizando.

Conclusión cont.

- Momentum no siempre puede acelerar la convergencia.
- La función de activación *exponencial* es más propensa a atascarse en mínimos.
- Tomar pocos puntos puede ser una muestra poco representativa, y por lo tanto, puede haber mala generalización.