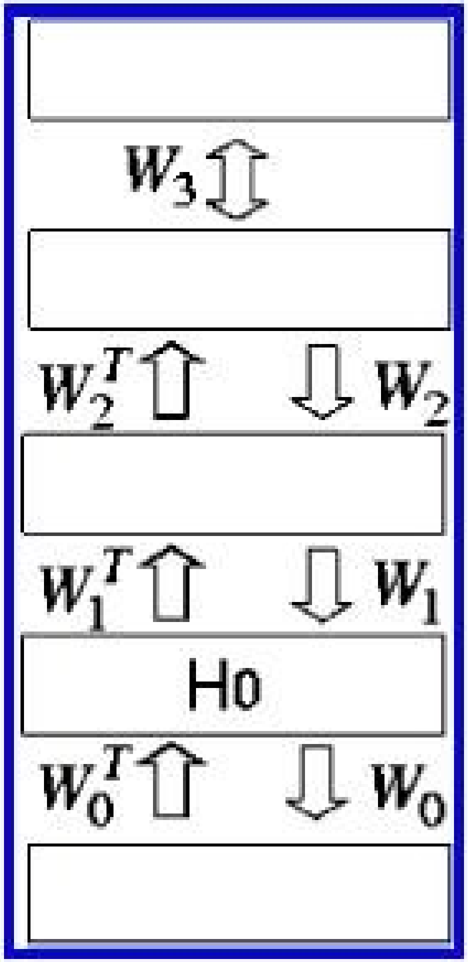
DEEP BELIEF NETWORKS

DEEP BELIEF NETWORKS



Estructura general:

- RBM's apiladas, aprendizaje no supervisado

- Última capa: aprendizaje supervisado

Múltiples niveles de representación

→ *Backpropagation*: antecedente ilustre, pero . . .

<u>Problemas</u>:
- Requiere datos etiquetados (labeled)
- Poco eficiente -lento- en aprendizaje con estructuras profundas
- Mínimos locales

→ Hinton (2007): *método para aprendizaje de representaciones distribuidas de a una capa por vez*

DBN: <u>*modelo generativo*</u> de datos no etiquetados, entrenables de a una capa de características por vez.

MODELOS ESTADÍSTICOS

- *Generativos*: generan todas las variables del proceso (observables y objetivo, calculables a partir de las observadas) → generan entradas y salidas, dados ciertos parámetros ocultos.

En la práctica: *tomar como entrada muestras de una cierta distribución y aprender un modelo que represente esa distribución*
Se busca: $P_{modelo} \sim P_{datos}$

- *Discriminativos*: sólo modelan las variables objetivo, en base a las observables→ infieren salidas de entradas.

En Redes Neuronales:

Modelo discriminativo típico: perceptrón (simple o multicapa)

Modelo generativo típico: máquinas de Boltzmann
Deep Belief Networks

APRENDIZAJE

<u>Etapa I</u> (no supervisada): *un algoritmo “goloso” (greedy) de aprendizaje*

- Aprender W_0 asumiendo que las matrices de pesos (superiores) están fijas (via CD).

- Congelar W_0 y usar W_0' para inferir distribuciones a posteriori (factorizables) aproximadas sobre los estados de las variables en la primera capa oculta (aunque ulteriores cambios en los pesos de las capas superiores pueden indicar correcciones a esta inferencia).

- Aprender un modelo RBM de los “datos” de nivel superior, que fue producido usando W_0' para transformar los datos originales.

Para la capa visible de cada RBM, puede usarse como activaciones las probabilidades a posteriori de las ocultas de la RBM anterior (en lugar de muestrear) → reducción del ruido.

- Entrenar cada capa en orden ascendente como una RBM, hasta la penúltima, usando como datos de entrenamiento las muestras generadas para la capa h de la anterior, muestreando siempre con las probabilidades

$$P(h \mid v) = \prod_i P(h_i \mid v) = \prod_i \text{sigmoid}\left(\sum_j W_{ji} v_j + d_i\right)$$

$$P(v \mid h) = \prod_j P(v_j \mid h) = \prod_j \text{sigmoid}\left(\sum_i W_{ji} h_i + b_j\right)$$

que equivale a muestrear según

$$P(h_j=1/v) = \sigma\,(d_j + \sum w_{ji} v_i)$$

$$P(v_i=1/h) = \sigma\,(b_i + \sum w_{ij} h_j)$$

Teorema: *a medida que se agregan capas ocultas (de features), mejora la cota inferior de la probabilidad logarítmica de los datos de entrenamiento.*

→ *e*l entrenamiento se basa en ajustar los pesos entre variables de manera de maximizar la probabilidad de que la red haya generado los datos observados

→ aprender *p*(imagen) y no *p*(etiqueta/imagen)

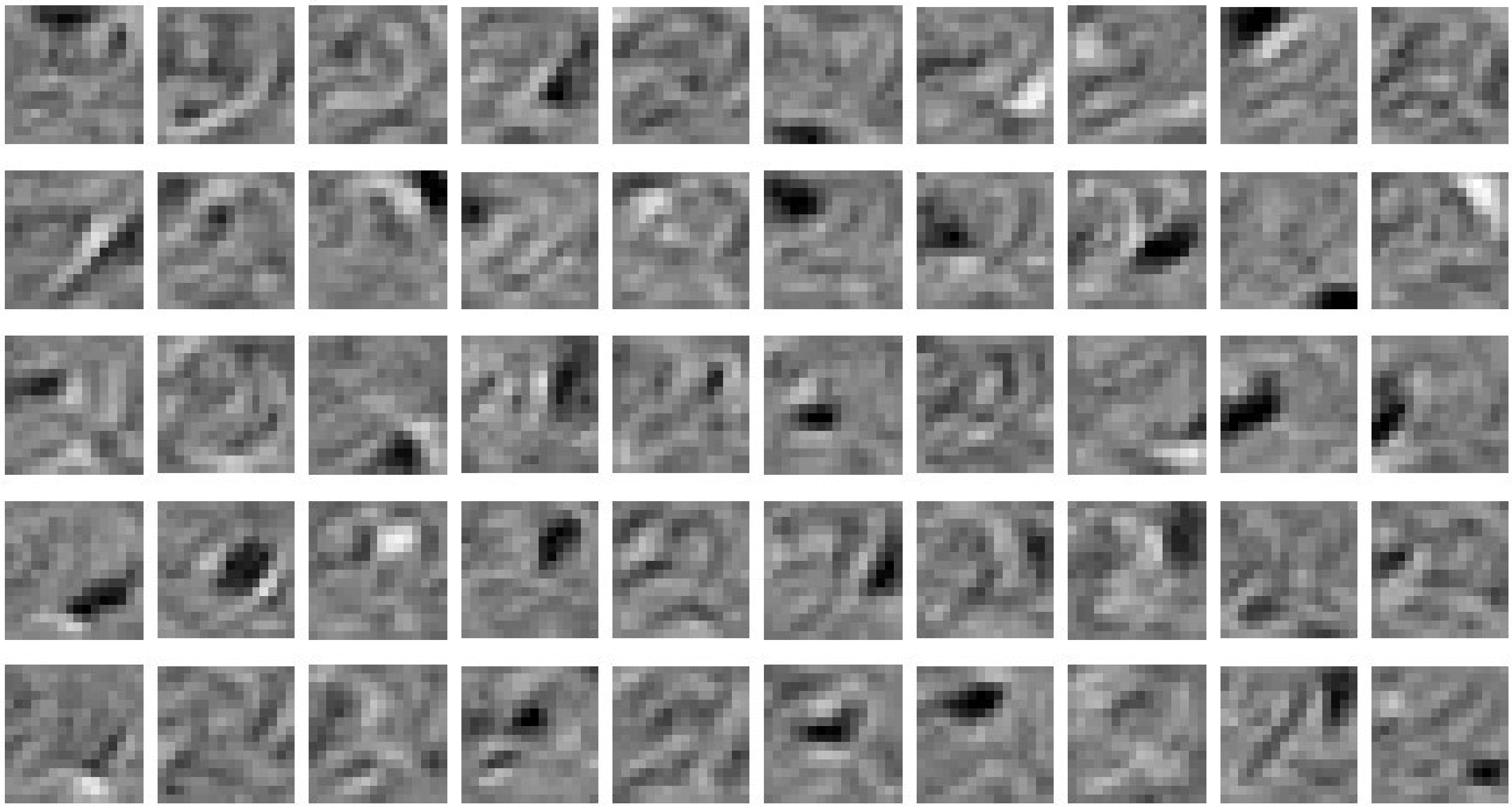
Recordar que

$$\partial \log p(\mathrm{v}) / \partial w_{ij} = \langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_\infty$$

Pero vimos que un solo paso de muestreo da buenos resultados

Observación: el descubrimiento de que las RBM se podían entrenar "golosamente" de una a la vez determinó que las DBN fueran una de las primeras estructuras profundas a considerar.

Ejemplo de un conjunto de 50 features extraidos de una imagen de 16 x 16 (=50 x 256 pesos)



Cada neurona detecta una característica diferente

Etapa II (supervisada): *back-fitting*

Ajuste de los pesos de las capas inferiores después de haber entrenado los de la superior (no dirigidos).

- Ajustar supervisadamente todos los pesos mediante descenso por gradiente sobre E(w) función de costo o error (usando explícitamente los rótulos asociados a las entradas).

Puede usarse backpropagation

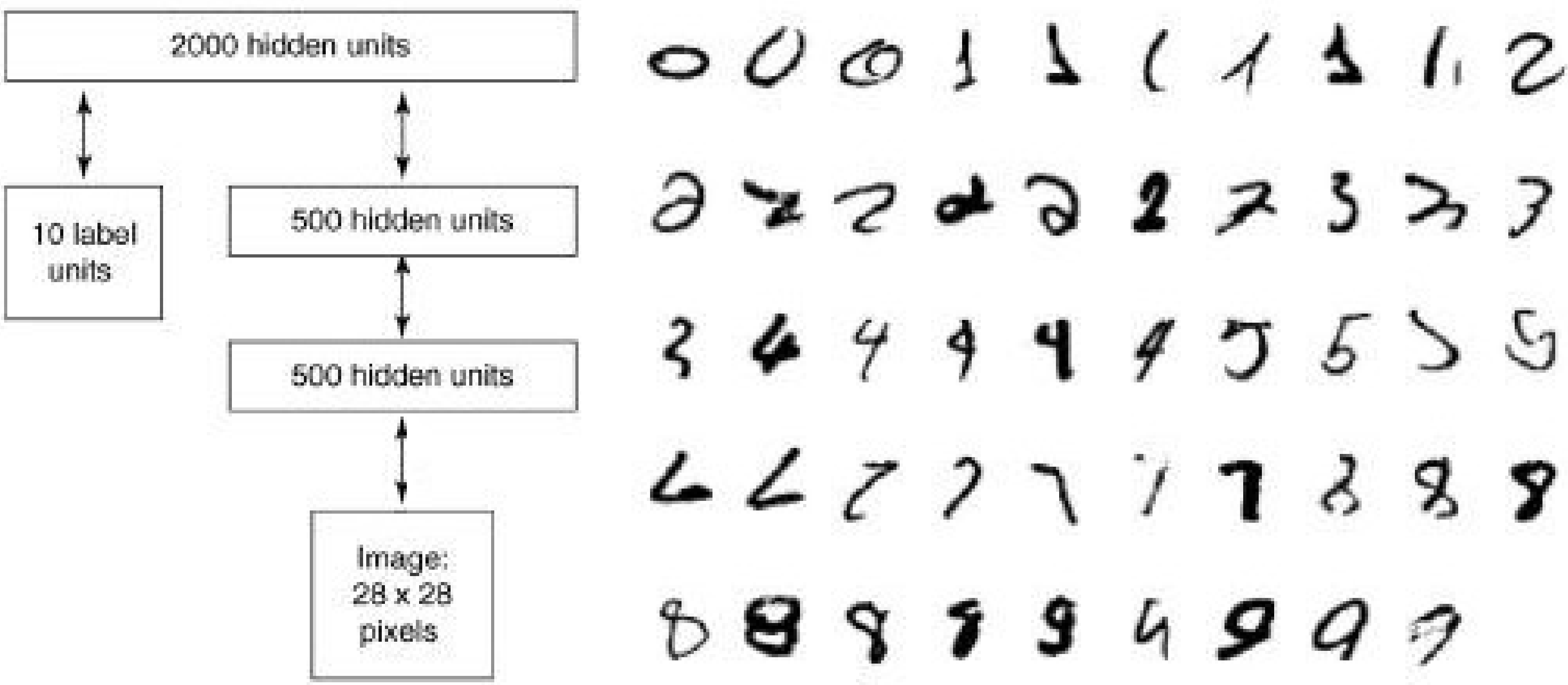
Observación 1: la etapa I puede verse como la inicialización de los pesos, que en la etapa II son ajustados supervisadamente.

Observación 2: las conexiones ascendentes de las capas inferiores (W_0', W_1' y W_2' en el ejemplo) no forman parte del modelo final, sólo para inferencia.

Observación 3: si se desea clasificar probabilísticamente, puede usarse la función Softmax:

$$\sigma(z)_i = exp(z_i) / \sum exp(z_j) \quad \text{para } i = 1, \ldots, K \;\; con \;\; z = (z_1, \ldots, z_K)$$

EJEMPLO (Hinton 2007)



Conjunto de entrenamiento.

Dígitos nunca presentados (test set) correctamente reconocidos