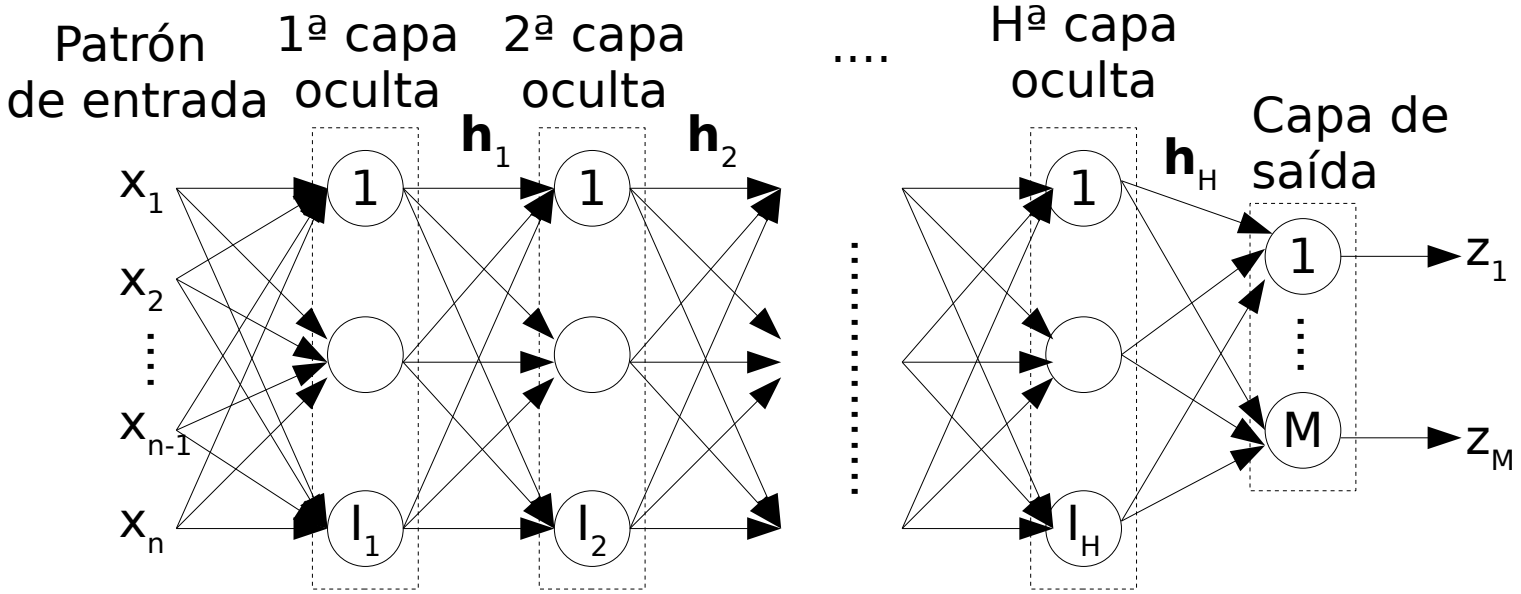
# Tema 5. Redes profundas

- Rede neuronal cun número elevado de capas ocultas, Deep Learning (DL)
- Orientado especialmente á análise automática de imaxes
- Non usa retro-propagación como na rede MLP: capas especializadas en detección de características sobre imaxes (p.ex., capas convolucionais)
- Inicialización non aleatoria como MLP, senón usando aprendizaxe non supervisado para facilitar o entrenamento posterior por retropropagación
- Para evitar entrenamentos lentos e caídas en mínimos locais, usa función de activación linear rectificada (ReLU)
- Intervalo con derivada non nula máis grande: mellores propiedades de derivabilidade, facilita o entrenamento, reduce a probabilidade de caída en mínimos locais

# Rede neuronal profunda (I)




1) **Pre-entrenamento non supervisado** de cada capa por separado, secuencialmente: máquina de Boltzman restrinxida (RBM). Non se usan as saídas desexadas

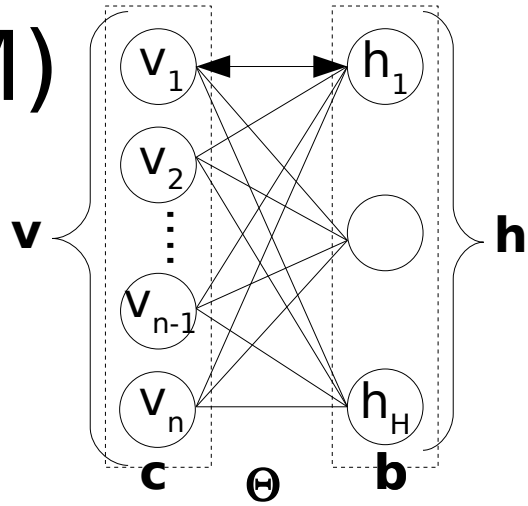
2) **Entrenamento supervisado** dos pesos de saída

3) **Sintonización precisa** dos pesos intermedios e de saída mediante retropropagación

# Deep learning (II)

- A capa oculta i crea un vector de características $\mathbf{h}_i$ : representación distribuida dispersa do patrón no i-ésimo nivel de características
- Extraen características automáticamente, capturan regularidades nos datos, en cada capa máis abstractas
- Evita a extracción manual de características, que require coñecemento do problema (p.ex. en imaxes)
- A rede DL extrae as características axeitadas en problemas complexos nos que é difícil extraer boas características
- O pre-entrenamento non supervisado descubre regularidades implícitas nos datos e sitúa os pesos en zonas do espazo con boas solucións
- Theodoridis: Machine learning: a Bayesian and optimization perspective, Academic Press, Cap. 18.8

# Pre-entrenamento: máquina restrinxida de Boltzman (RBM)




- Estructura da rede (n entradas, H saídas): conexións bilaterais
- Paráms: $\boldsymbol{\Theta}=\{\theta_{ij}\}_{ij=1}^{nH}, \mathbf{b}=\{b_i\}_{i=1}^{H}, \mathbf{c}=\{c_i\}_{i=1}^{n}$
- Enerxía da rede:

$v_i, h_j \in \{0,1\}$
$i=1,\ldots,n$
$j=1,\ldots,H$

$$E(\boldsymbol{v},\boldsymbol{h})=\sum_{i=1}^{n} v_i \sum_{j=1}^{H} \theta_{ij} h_j + \sum_{j=1}^{H} b_j h_j + \sum_{i=1}^{n} c_i v_i = \boldsymbol{v}^T \boldsymbol{\Theta} \boldsymbol{h} + \boldsymbol{b}^T \boldsymbol{h} + \boldsymbol{c}^T \boldsymbol{v}$$

- Maximiza logaritmo da probabilidade (log-likelihood) L:

$$L=\frac{1}{N}\sum_{n=1}^{N} \log P(\boldsymbol{v}_n,\boldsymbol{h}) \qquad P(\boldsymbol{v}_n,\boldsymbol{h})=\frac{\mathrm{e}^{-E(\boldsymbol{v}_n,\boldsymbol{h})}}{Z} \qquad Z=\sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$$

$$L=\frac{1}{N}\sum_{n=1}^{N} \log\left(\frac{1}{Z}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}_n,\boldsymbol{h})}\right)=\frac{1}{N}\sum_{n=1}^{N} \log\left(\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v}_n,\boldsymbol{h})}\right)-\frac{1}{N}\log\left(\sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}\right)$$

# RBM (II)

- Gradiente: $\frac{\partial L}{\partial \theta_{ij}} = A\left(h_i^1, v_j^1\right) - B\left(h_i^2, v_j^2\right)$
- O superíndice é o tempo: $v_j^1 = v_{nj}$, $\mathbf{v}^1 = \mathbf{v}_n^1 = \mathbf{v}_n$
- $h_i^1 \sim P(h_i=1|\mathbf{v}^1)$: $h_i^1=1$ aleatoriamente con prob. $P(h_i=1|\mathbf{v}^1)$

$$P\left(h_i=1|\boldsymbol{v}^1\right) = 1-\sigma\left(\sum_{j=1}^{H}\theta_{ij}v_j^1 + b_i\right) \qquad \sigma(x) = \frac{1}{1+\mathrm{e}^{-x}}$$

← Función sigmoide

- $v_j^2 \sim P(v_j=1|\mathbf{h}^1)$: $v_j^2=1$ con probabilidade $P(v_j=1|\mathbf{h}^1)$

$$P\left(v_j=1|\boldsymbol{h}^1\right) = 1-\sigma\left(\sum_{i=1}^{n}\theta_{ij}h_i^1 + c_i\right)$$

- $h_i^2 \sim P(h_i=1|\mathbf{v}^2)$: $h_i^2=1$ con probabilidade $P(h_i=1|\mathbf{v}^2)$

$$\bar{v}_j^1 = \frac{1}{Q}\sum_{n=1}^{Q} v_{nj}$$

$$A\left(h_i^1, v_j^1\right) = \frac{h_i^1}{N}\sum_{n=1}^{N} v_{nj}\sum_{\boldsymbol{h}} P\left(\boldsymbol{h}^1|\boldsymbol{v}_n\right) = \frac{h_i^1}{N}\sum_{n=1}^{N} v_{nj} = h_i^1\bar{v}_j^1$$

$$\sum_{\boldsymbol{h}} P\left(\boldsymbol{h}^1|\boldsymbol{v}_n\right) = 1$$

$$B\left(h_i^2, v_j^2\right) = \sum_{\boldsymbol{v}}\sum_{\boldsymbol{h}} P(\boldsymbol{v},\boldsymbol{h})\, h_i^2 v_j^2 = h_i^2 v_j^2$$

$$P\left(\boldsymbol{h}|\boldsymbol{v}_n\right) = \frac{P\left(\boldsymbol{v}_n,\boldsymbol{h}\right)}{\sum_{\boldsymbol{h}'} P\left(\boldsymbol{v}_n,\boldsymbol{h}'\right)}$$

# RBM (III)

- Actualización de pesos:

$$\Delta\theta_{ij}=\theta_{ij}(t+1)-\theta_{ij}(t)=\mu\frac{\partial L}{\partial\theta_{ij}}=\mu\left(h_i^1\bar{v}_j^1-h_i^2v_j^2\right)$$

$$\bar{\boldsymbol{v}}^1=\frac{1}{Q}\sum_{n=1}^{Q}\boldsymbol{v}_n$$

- Promedio sobre un bloque de Q patróns:

$$\Delta\theta_{ij}=\frac{\mu}{Q}\sum_{n=1}^{Q}\left(h_i^1v_{jn}^1-h_i^2v_j^2\right)$$

- En forma matricial: $\Delta\boldsymbol{\Theta}=\frac{\mu}{Q}\sum_{n=1}^{Q}\left(\boldsymbol{h}^1\boldsymbol{v}_n^{1T}-\boldsymbol{h}^2\boldsymbol{v}^{2T}\right)=\mu\left(\boldsymbol{h}^1\bar{\boldsymbol{v}}^{1T}-\boldsymbol{h}^2\boldsymbol{v}^{2T}\right)$
- Usando valores medios de **h**:

$$\Delta\boldsymbol{\Theta}=\frac{\mu}{Q}\left(\bar{\boldsymbol{h}}^1\bar{\boldsymbol{v}}^{1T}-\bar{\boldsymbol{h}}^2\boldsymbol{v}^{2T}\right)\qquad\Delta\boldsymbol{b}=\frac{\mu}{Q}\left(\bar{\boldsymbol{h}}^1-\bar{\boldsymbol{h}}^2\right)\qquad\Delta\boldsymbol{c}=\frac{\mu}{Q}\left(\bar{\boldsymbol{v}}^1-\bar{\boldsymbol{v}}^2\right)$$

- **Θ**, **b** e **c** inicialízanse aleatoriamente
- Remata ao cumprirse certo criterio de converxencia

# Algoritmo de entrenamento da RBM

**entrada**: $\{\mathbf{v}_n\}_{n=1}^{N}$, patróns de entrada, sen saída desexada

**inicialización**: $\Theta$, $\mathbf{b}$, $\mathbf{c}$ aleatorios

**for** t=1:T  # t,T=época actual,nº de épocas

  **for** l=1..B  # B=nº de bloques de tamano Q

    $\mathbf{G}=\mathbf{0}, \mathbf{g}_b=\mathbf{0}, \mathbf{g}_c=\mathbf{0}$  # gradentes

    **for** $\mathbf{v}_n$ in $B_l$  # $B_l$=bloque l-ésimo

      $\mathbf{v}^1=\mathbf{v}_n$; $\mathbf{h}^1 \sim P(\mathbf{h}=\mathbf{1}|\mathbf{v}^1)$; $\mathbf{v}^2 \sim P(\mathbf{v}=\mathbf{1}|\mathbf{h}^1)$; $\mathbf{h}^2 \sim P(\mathbf{h}=\mathbf{1}|\mathbf{v}^2)$

      $\mathbf{G}=\mathbf{G}+(\bar{\mathbf{h}}^1 \mathbf{v}_n^{1T} - \bar{\mathbf{h}}^2 \mathbf{v}^{2T})$; $\mathbf{g}_b=\mathbf{g}_b+(\bar{\mathbf{h}}^1 - \bar{\mathbf{h}}^2)$; $\mathbf{g}_c=\mathbf{g}_c+(\mathbf{v}_n^1 - \mathbf{v}^2)$

    **endfor**

    $\Theta=\Theta+\mu\mathbf{G}/Q$; $\mathbf{b}=\mathbf{b}+\mu\mathbf{g}_b/Q$; $\mathbf{c}=\mathbf{c}+\mu\mathbf{g}_c/Q$

  **endfor**

  **if** criterio convergencia **break**

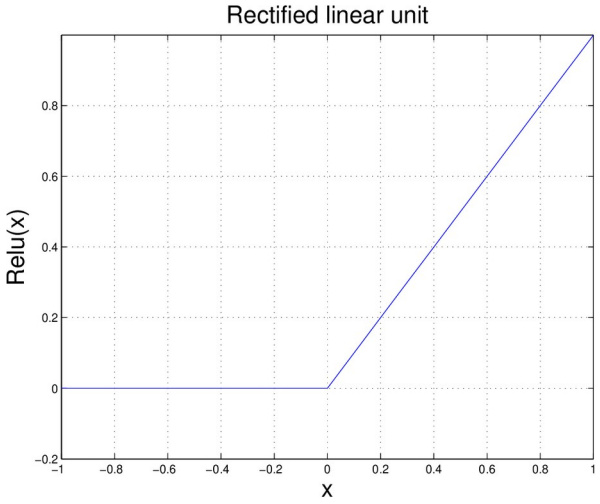
**endfor**

# De RBM a DL

- Para saídas reais, RBM usa neuronas de saída gausianas, con enerxía E(**v**,**h**) definida como:

$$E(\boldsymbol{v},\boldsymbol{h})=\sum_{i=1}^{n}\frac{(v_i-c_i)^2}{2\sigma_i^2}+\sum_{j=1}^{H}b_j h_j-\sum_{i=1}^{n}\frac{v_i}{\sigma_i}\sum_{j=1}^{H}\theta_{ij}h_j=\frac{|\Sigma(\boldsymbol{v}-\boldsymbol{c})^T|^2}{2}+\boldsymbol{b}^T\boldsymbol{h}-\boldsymbol{v}^T\Sigma^T\Theta\boldsymbol{b}$$

- Onde $\boldsymbol{\Sigma}$ é a matriz diagonal con $1/\sigma_1,\ldots,1/\sigma_n$ na diagonal
- DL usa activación **relu** (rectified linear unit): f(t)=R(t)=max(0,t)
- Reduce os problemas da sigmoide e tanh por derivada nula
- Aproximación derivábel de relu: softplus: $f(t)=\ln(1+e^x)$



# Algoritmo Deep Learning completo

Inicialización: $\{\Theta_k, \mathbf{b}_k, \mathbf{c}_k\}_{k=1}^{H}$, $\mathbf{w}_j^H$ aleatorios, $\mathbf{h}_n^1=\mathbf{x}_n$, n=1..N

Fase 1: pre-entrenamento non supervisado (inicialización de pesos) de capas ocultas (RBM)

**for** k=1..H  # H=nº de capas ocultas (redes RBM)

$\mathbf{h}^{k-1}$=actividade da neurona de saída, $\mathbf{h}^k$=actividade da neurona oculta da RBM k

Entrena a RBM k-ésima calculando $\Theta_k$, $\mathbf{b}_k$, $\mathbf{c}_k$ usando o algoritmo de páx. 7

Usa $\Theta_k$, $\mathbf{b}_k$, $\mathbf{c}_k$ para calcular $\{\mathbf{h}^k_n\}_{n=1}^{N}$ como $\mathbf{h}^k_n \sim P(\mathbf{h}|\mathbf{h}^{k-1}_n)$, n=1..N

**endfor**

Fase 2: pre-entrenamento supervisado (inicialización de pesos) da capa de saída

Entrena os pesos $\mathbf{w}_j^H$ da capa de saída ($\mathbf{h}^H$,$\mathbf{y}$) usando $\{\mathbf{h}^H_n, y_n\}_{n=1}^{N}$ para o entrenamento

Fase 3: sintonización fina de tódalas neuronas mediante aprendizaxe supervisada

Usa os pesos $\{\Theta_k, \mathbf{b}_k, \mathbf{c}_k\}_{k=1}^{H}$, $\mathbf{w}_j^H$ como valores iniciais e entrena a rede completa mediante retropropagación usando $\{\mathbf{x}_n, y_n\}_{n=1}^{N}$ como datos de entrenamento

# Algoritmo de entrenamento ADAM

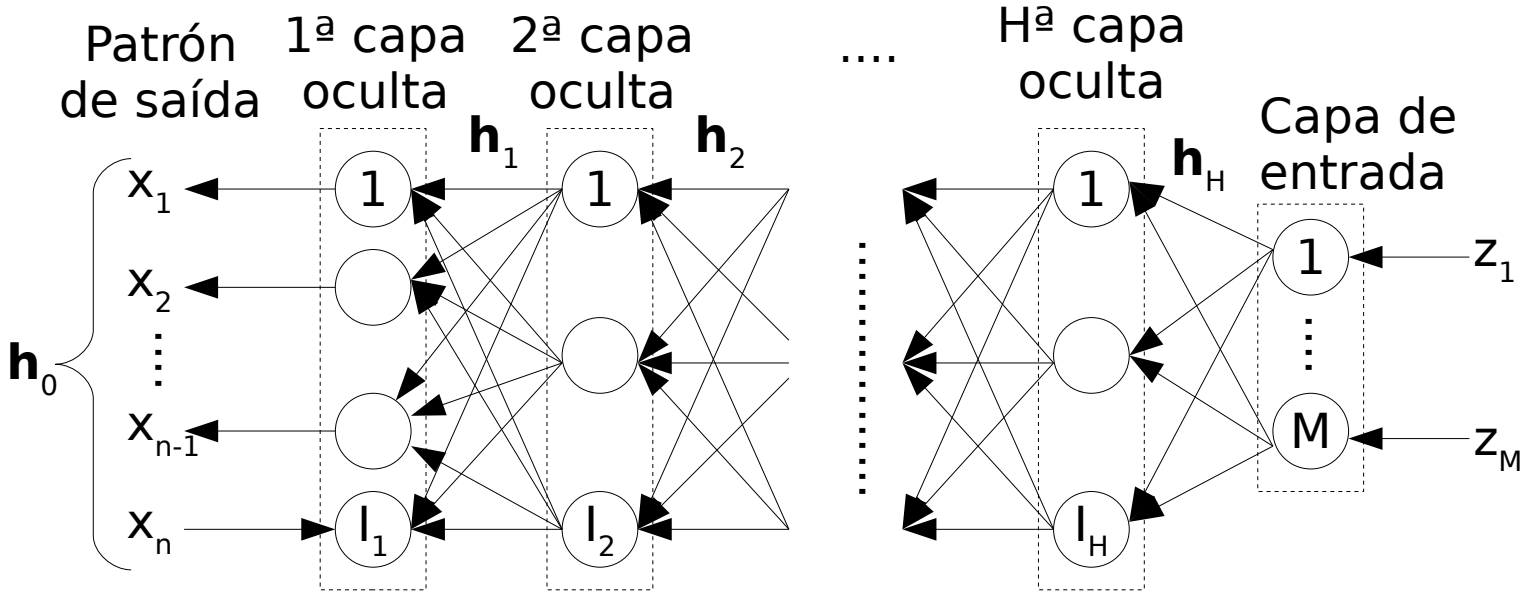
- Adaptive moment estimation
- Deseñado para Deep Learning. Combina RMSprop e descenso de gradente con momento
- Alternativo a RBM

$V_w=0;S_w=0;V_b=0;S_b=0$ #V=momento,S=varianza
for k=1:K #k=época,K=nº épocas
calcula $\Delta\mathbf{w}$,$\Delta$b usando retropropagación
$V_w=\beta_1 V_w+(1-\beta_1)\Delta\mathbf{w}$; $V_b=\beta_1 V_b+(1-\beta_1)\Delta b$; # Momento
$S_w=\beta_2 S_w+(1-\beta_2)|\Delta\mathbf{w}|^2$; $S_b=\beta_2 S_b+(1-\beta_2)\Delta b^2$; # RMSprop
$V_w=V_w/(1-\beta_1)$; $S_w=S_w/(1-\beta_2)$; $V_b=V_b/(1-\beta_1)$; $S_b=S_b/(1-\beta_2)$

$$\boldsymbol{w}(t+1)=\boldsymbol{w}(t)-\frac{\mu V_w}{\varepsilon+\sqrt{S_w}};b(t+1)=b(t)-\frac{\mu V_b}{\varepsilon+\sqrt{S_b}}$$

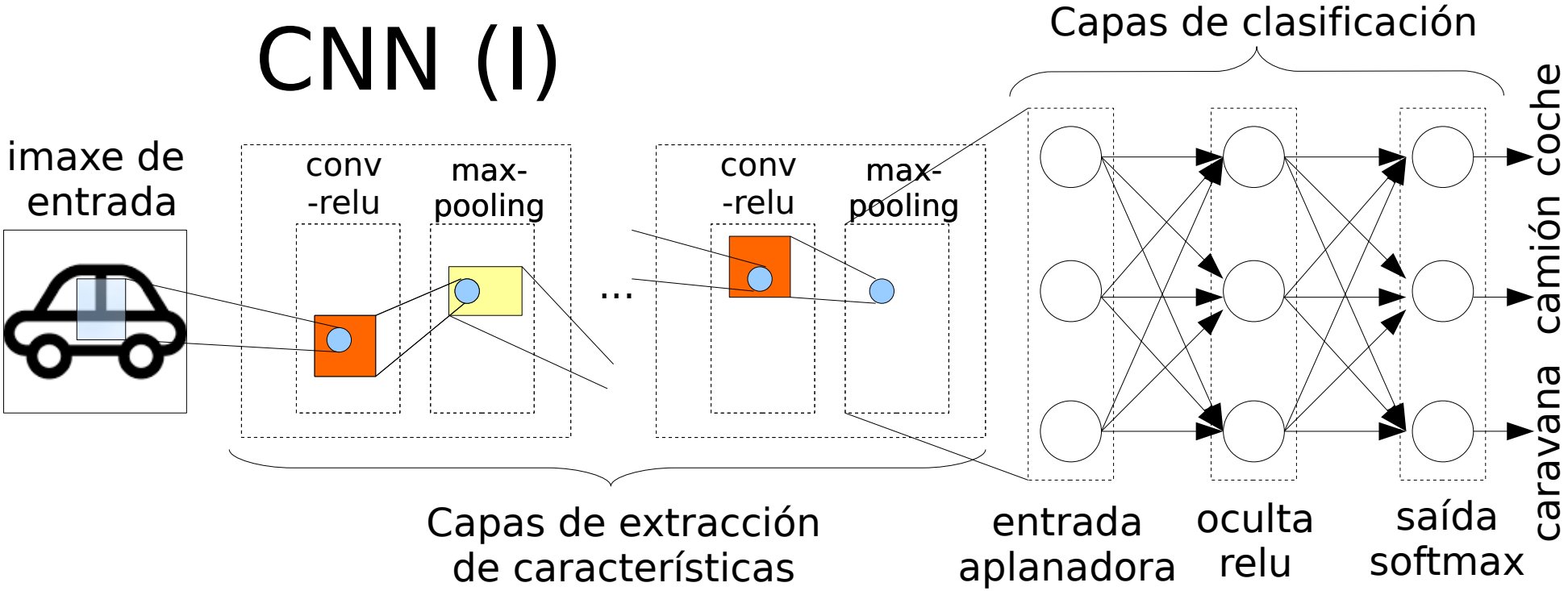
endfor

# Rede de crenzas profunda (deep belief network, DBN)




- Modelo xerativo: non aprende a distribución dos datos: xera datos dacordo coa súa distribución: fluxo dereita→esquerda
- Calcula $\mathbf{h}_{H-1} \sim P(\mathbf{h}|\mathbf{h}_H)$, usando $\mathbf{h}_H = \mathbf{z}_i$ (un patrón de saída)
- Para k=H-1,...,1, calcula $h_i^{k-1} \sim P(h_i|\mathbf{h}_k)$ para $i=1,...,l_k$
- O patrón de saída xerado é $\mathbf{x}=\mathbf{h}_0$

# Rede convolucional CNN (I)



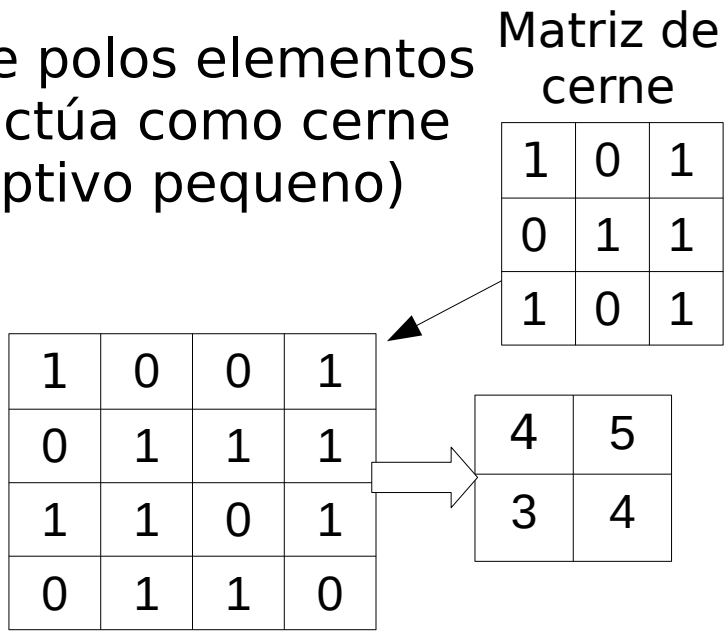

- Extracción de características da imaxe usando filtros. Cada filtro contén 2 capas (mapas): convolución-relu (entrada) e max-pooling (saída)
- Capas de clasificación: capas de aplanado, oculta e softmax
- Previas ao Deep Learning, baseadas na corteza visual do cerebro e na rede neuronal Neocognitron (anos 80)
- Recoñecemento de formas 2D (imaxe en escala de grises) ou 3D (imaxe en cor) con capas de neuronas 2D-3D: mapas de características

# Rede convolucional (II)

- Invarianza a traslacións, rotacións, escalado e distorsións
- Conectividade local entre neuronas e transformacións da imaxe organizadas xerárquicamente
- Cada neurona dun mapa recibe entradas so dende unha área específica do mapa anterior (campo receptivo) porque so ten pesos dende as neuronas dese campo
- Aprende a extraer automáticamente características locais da imaxe e conserva as súas posicións relativas
- Os campos receptivos poden superpoñerse parcialmente, dando saídas correlacionadas espacialmente
- Os nodos do mesmo mapa que reciben entradas dende os mesmos campos receptivos comparten pesos (menos parámetros libres, rede dispersa, invarianza a traslacións)
- Moitas capas ocultas evitando problemas de entrenamento

# Capa conv-relu (I)

- Extrae características de alto nivel (p.ex. bordes, esquinas, extremos) da imaxe
- Convolución: multiplica os valores da imaxe polos elementos (0,1) da matriz de pesos da neurona, que actúa como cerne da convolución, de orde baixa (campo receptivo pequeno)
- Tamano da matriz de cerne: k
- Esta matriz percorre a imaxe (Esquerda→Dereita e Arriba→Abaixo)
- Logo, función de activación relu que propaga as características relevantes




- **Profundidade** (depth, d): nº de neuronas na capa max-pooling conectándose á mesma rexión da capa conv-relu
- **Paso** (stride, s): paso usado polo producto convolucional en dimensións X e Y da imaxe (e Z, se estamos en 3D); s>1 reduce o mapa

# Capa conv-relu (II)

- **Recheo** (padding, p): nº de neuronas con valor 0 nas beiras da imaxe (opcional)

  Evita a perda de información que se produce porque os píxeles das beiras se usan menos

- As neuronas da mesma zona (de tamano dxd) comparten pesos
- Cando existe na imaxe algunha estructura cunha posición específica (p.ex. ollos), as neuronas non comparten pesos
- Na capa conv-relu, a saída dunha neurona é a función relu aplicada sobre a convolución (producto) entre o patrón 2D de entrada e o cerne (matriz cos pesos das neuronas)
- O tamano do campo receptivo é un hiperparámetro sintonizábel da CNN
- A activación relu anula as convolucións negativas

# Capa max-pooling (I)

- Divide a imaxe en rectángulos (usualmente 2x2) non superpostos
- Selecciona o máximo de cada rectángulo
- A saída é unha versión reducida (de menor resolución) da imaxe: subsampling, usualmente con paso s=2 (reduce á metade)


| | | | |
|---|---|---|---|
| 12 | 4 | 5 | 13 |
| 7 | 32 | 1 | 25 |
| 42 | 21 | 22 | 90 |
| 51 | 18 | 61 | 30 |

| | |
|---|---|
| 32 | 25 |
| 51 | 90 |

- Reduce a dimensionalidade do mapa, o consumo de memoria, o custe computacional e os problemas de sobreaprendizaxe porque quedan menos parámetros libres

# Capa max-pooling (II)

- Extrae características dominantes invariantes a traslacións e suprime ruido
- A sucesión de filtros reduce progresivamente o tamano da imaxe
- O número m de neuronas dunha capa max-pooling é:

$$m = \left\lfloor 1 + \frac{w - k + 2p}{s} \right\rfloor$$

  w=nº neuronas da capa conv-relu; k=tamano do kernel; p=recheo; s=paso. O valor m debe ser enteiro

- A posición absoluta dunha característica é menos importante que a súa posición relativa ás outras características

# Expresións conv-relu e max-pooling

- Capa conv-relu r-ésima: entrada $\mathbf{A}^r$ de orde 3D de tamano $(n_x^r, n_y^r, n_c^r)$, onde $n_c^r$=nº canles de cor
- Cerne $\mathbf{K}^r = \{K_{ijk}^r\}$ (pesos das neuronas) de tamano $(f_r, f_r, n_c^r)$
- Convolución de $\mathbf{A}^r$ e $\mathbf{K}^r$: $$C_{xy}^r\left(A^r, K^r\right) = \sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\sum_{k=1}^{n_C} K_{ijk}^r A_{x+i-1,\, y+j-1,\, k}^r$$
- Tamano de $C^r_{xy}$: $(n_x+2p-f, n_y+2p-f)$ se s=0 (p=recheo,s=paso) $$\left(\left\lfloor \frac{n_x+2p-f}{2}+1 \right\rfloor, \left\lfloor \frac{n_y+2p-f}{2}+1 \right\rfloor\right)$$ se s>0
- Saída da capa conv-relu: $a_{xyz}^r = R(C^r_{xy})$. Pesos entrenábeis: $\{K^r_{ijk}\}$ $$R(t) = max(0, t)$$
- Capa max-pool: saída $a_{xyz}^r$ de tamano $f_p^r$: $a_{xyz}^r = max\left\{a_{x+i-1,\, y+j-1,\, z}^{r-1}\right\}_{ij=1}^{f_p^r}$
- Esta última capa non ten pesos entrenábeis

# Expresións das capas de clasificación

- Capas completamente interconectadas
- Entrada: **aplanado**: transforma imaxe **A** en vector **z**:

$$\boldsymbol{z} = flat\left(\boldsymbol{A}\right), \boldsymbol{A} = \left\{a_{xyz}\right\}, x = 1..n_x, y = 1..n_y, z = 1..n_c$$

- Oculta: activación **relu**: saída $u_i$ da neurona i=1..n:

$$u_i = R\left(\sum_{j=1}^{n} w_{ij} a_j + b_i\right) \qquad R\left(t\right) = max\left(0, t\right)$$

- Saída $y_i$, i=1..C para clasificación con C clases: función de activación **softmax** S(**u**)

$$y_i = S_i\left(\boldsymbol{u}\right) = \frac{\mathrm{e}^{u_i}}{\sum_{j=1}^{n} \mathrm{e}^{u_j}}, i = 1..C$$

# Rede convolucional (III)

- Regularización por **abandono** (dropout): en cada etapa do entrenamento, suprímense neuronas (e as súas conexións)

  Supresión aleatoria con probabilidade 1-p (p=0.5 en capas ocultas, p moi baixo en capas de entrada)

  Entrénase a rede reducida para acelerar o entrenamento

  No teste, reincorpóranse as neuronas suprimidas cos seus pesos orixinais, e pondérase por p a saída de cada nodo

- Hiperparámetros da CNN:

1) Nº de filtros: maior cantos máis datos dispoñíbeis
2) Forma do filtro: depende do tipo de imaxes e dos obxectos a detectar, existen moitas formas
3) Forma do max-pooling: usualmente 2x2, con imaxes grandes 4x4. Meirande pooling reduce moito o tamano e pérdese información

# Bases de datos de imaxes

- MNIST handwritten datasets: 60,000 train, 10,000 test
- Google house numbers from street view: 600,000 imaxes de números
- CIFAR-10: 60,000 imaxes cor 32x32 de 10 clases
- ImageNet: > 150 GB
- Tiny Images: 80 millóns de imaxes
- Flickr dataset: 100 millóns de imaxes
- Competición ILSVRC: ImageNet large scale visual recognition challenge
- Redes CNN pre-entrenadas: LeNet, AlexNet, VGGNet, GoogLeNet, ResNet, ZFNet

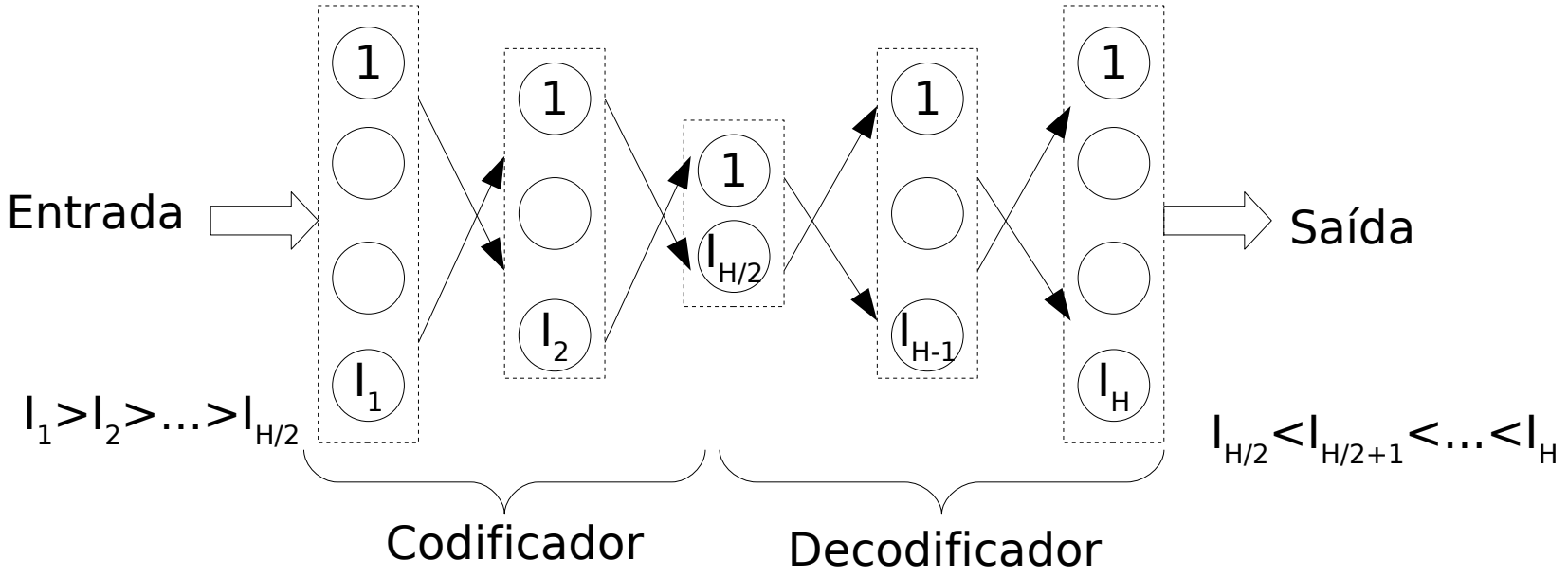
# Rede **Alexnet** (ILSVRC-2012): imaxe 224x224x3 (RGB)

- Filtro 1: conv-relu, 96 mapas de características de tamano 11x11 con paso 4, reduce a imaxe a 55x55x96; max-pooling con filtro 3x3 e paso 2, reduce a 27x27x96
- Filtro 2: conv-relu, 256 mapas 5x5 paso 1; max-pooling 3x3 paso 2; reduce a 13x13x256
- Filtro 4: conv-relu 3x3, 384 mapas, paso 1
- Filtro 4: conv-relu 3x3, 384 mapas, paso 1
- Filtro 5: conv-relu 3x3, 256 mapas, paso 1; max-pooling 3x3 paso 2, 256 mapas
- Capa 6: full-connected aplanadora con 9216 neuronas
- Capas 7-8: full-connected, 4096 neuronas, activación relu
- Capa 9: full-connected, activación softmax, 1000 neuronas

# Características de Alexnet

| | Tipo | Mapas | Tamano | Cerne | Paso | Activación |
|---|---|---|---|---|---|---|
| Filtro | Imaxe | 1 | 227x227x3 | | | |
| 1 | conv-relu | 96 | 55x55x96 | 11x11 | 4 | relu |
| | max-pooling | 96 | 27x27x96 | 3x3 | 2 | relu |
| 2 | conv-relu | 256 | 13x13x256 | 5x5 | 1 | relu |
| | max-pooling | 256 | 13x13x256 | 3x3 | 2 | relu |
| 3 | conv-relu | 384 | 13x13x384 | 3x3 | 1 | relu |
| 4 | conv-relu | 384 | 13x13x384 | 3x3 | 1 | relu |
| 5 | conv-relu | 256 | 13x13x256 | 3x3 | 1 | relu |
| | max-pooling | 256 | 6x6x256 | 3x3 | 2 | relu |
| 6 | full-connected | | 9216 | | | relu |
| 7 | full-connected | | 4096 | | | relu |
| 8 | full-connected | | 4096 | | | relu |
| 9 | full-connected | | 1000 | | | softmax |

# Rede autocodificadora profunda (deep autoencoder)




- Úsase para reducir a dimensionalidade dos datos
- Codificador: primeira metade da rede, reduce o tamano dos datos
- Decodificador: segunda metade da rede, aumenta o tamano dos datos
- Capa central: datos con dimensionalidade reducida

# Autocodificadores (II)

- Rede neuronal profunda con H capas, $I_k$ decrecente para k=1...H/2 e $I_k$ crecente para k=H/2+1...H
- Capas con funcións de activación binaria agás a K/2 e a H, que usan activación linear
- Entrénanse con $\{\mathbf{x}_n,\mathbf{x}_n\}_{n=1}^{N}$ : aprende a reconstruir os datos na capa de saída
- A capa intermedia k=K/2 (linear) é unha representación dos datos cunha dimensionalidade reducida
- O decodificador pode empregarse para xerar datos novos, actuando como modelo xerativo dos datos
- Entrenamento por retropropagación, con función de activación sigmoide ou relu