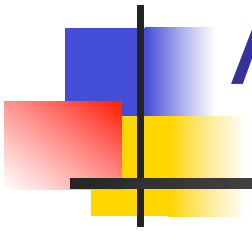
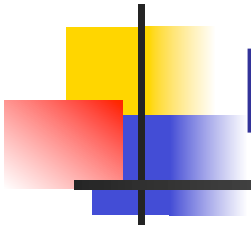


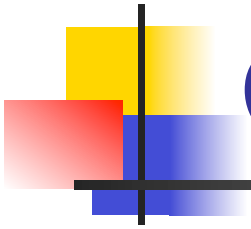
Aprendizaje de Máquina

ITAM



Menu

- Complejidad de una Red Neuronal
  - Parámetros y técnicas
- Redes Profundas
  - Problemas para entrenarlas
  - Algunas soluciones
  - Redes Convolucionales

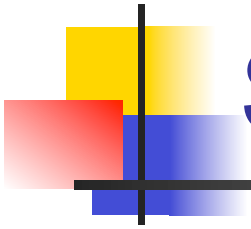


Complejidad de la Red

- La complejidad de la red esta dada por
  - La cantidad de pesos (grados de libertad)
  - El número de capas intermedias
- El concepto de deep learning se refiere a entrenar redes con muchas capas y, consecuentemente, muchas neuronas

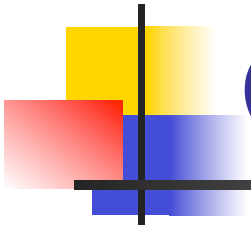
Problemas para entrenar una red profunda

- Problemas para entrenar redes profundas (más de 2 capas)
  - Existen muchos parámetros (pesos) a ajustar
  - Las capas aprenden a distintas velocidades
- Gradiente inestable
  - Diminishing gradient
    - Conforme propagamos el error la información acerca del mismo disminuye en cada capa
    - Del algoritmo de retro-propagación vemos que para cada capa multiplicamos por el error de la capa subsiguiente (que es normalmente menor a 1) (la derivada de la sigmoide tiene máximo en 0.25)
    - La sigmoide (y otras) se saturan
  - En casos donde las w´s son muy grandes el efecto puede ser el contrario y los gradientes crecer (!!!). Pero es difícil si regularizamos. Además hacer muy grande w reduce la derivada de la sigmoide ... pero existen valores para los cuales explota



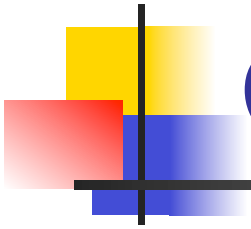
Soluciones

- No hay una solución total, pero la investigación indica:
  - Controlar la complejidad de la red (técnicas de regularización)
  - Escoger bien los pesos iniciales
  - Usar otras funciones de activación (ReLU)
  - Modificaciones del algoritmo de descenso en gradiente e.g.,uso de momento
    - El momento añade una fracción (otro parámetro!!) del cambio en el peso de la iteración i-1 al nuevo peso
  - Más ejemplos y más poder de cómputo (hace 5 años era totalmente impráctico)
  - Otras topologías de red



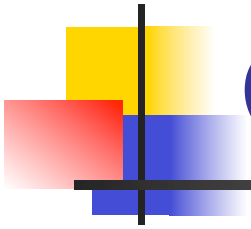
Complejidad de la Red

- Podemos controlar la complejidad de la red con:
  - Regularización
    - Lasso (L1)
    - Ridge (L2)
    - Elastic net (combinación lineal de ambas)



Complejidad de la Red

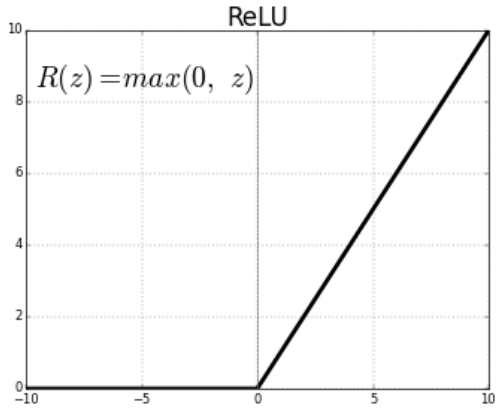
- Drop out
  - Durante el entrenamiento, para cada ejemplo, cada neurona es elegida con probabilidad p (usualmente 0.5 o usar validación) para ser ignorada
  - Durante la prueba se usan todas las neuronas pero sus pesos se escalan usando p
  - La idea es que esto aproxima el entrenar muchas redes diferentes para luego promediarlas. El efecto es reducir la varianza



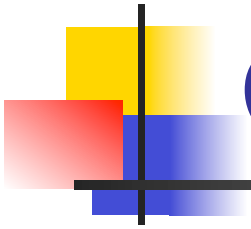
Complejidad de la Red

- Early stopping
  - Dejar de entrenar la red cuando el error de validación de una iteración a otra comience a subir (preservar los pesos de la iteración previa)

Otras funciones de activación

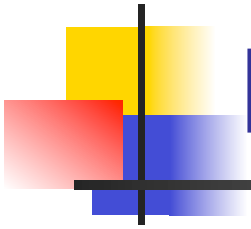


- ReLu (Rectified linear unit)
  - Max(0,$w^T x$)
  - Ayuda al problema de diminishing gradient, pues no se saturan (de un lado)
  - Aproximadamente la mitad desactivadas al inicio del entrenamiento (usando pesos aleatorios)



Cambio de Topología

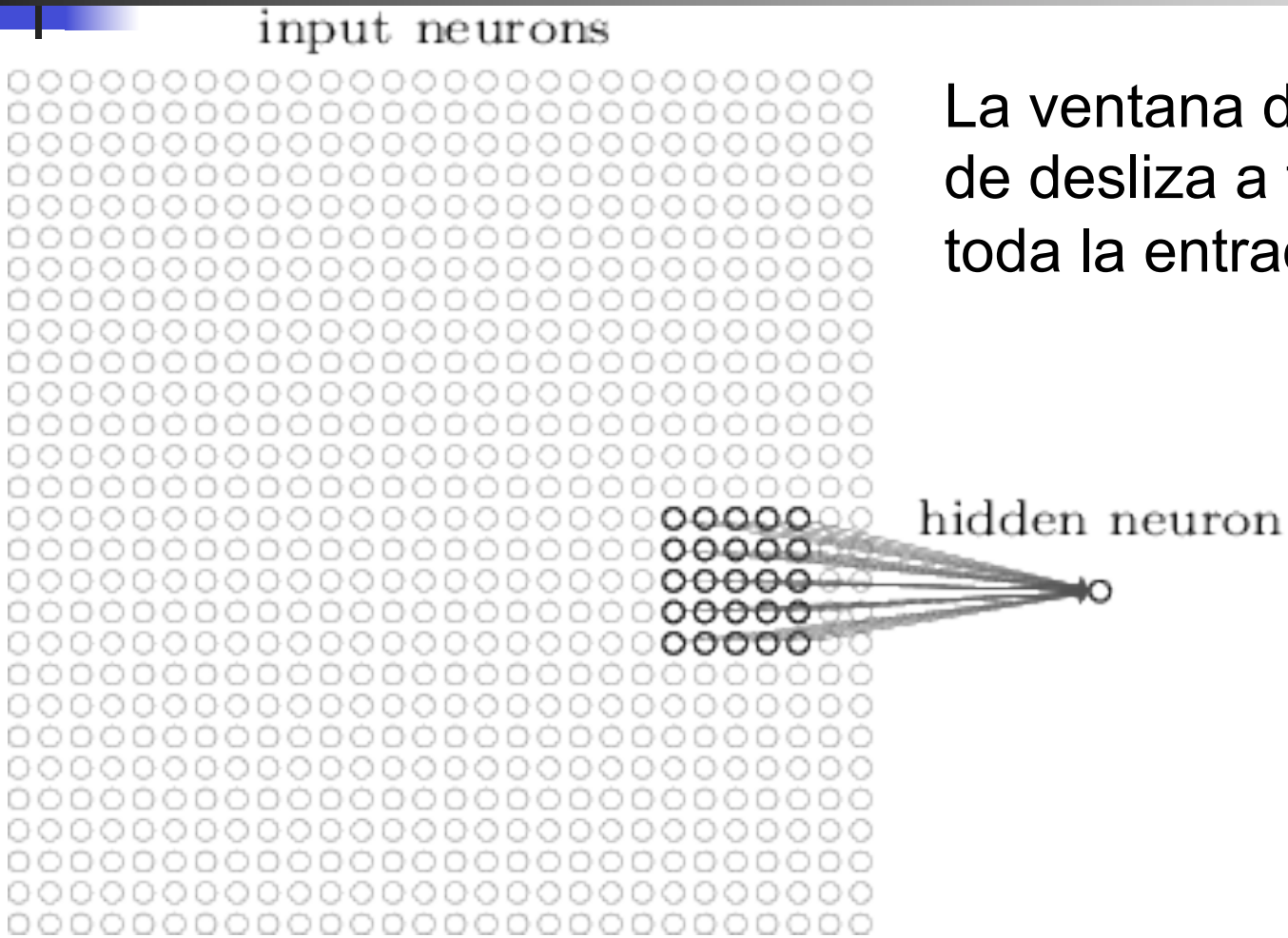
- Redes Convolucionales
  - Particularmente buenas para imágenes
  - Toman en cuenta la estructura espacial de una imagen
    - Algo menos que aprender!
  - Tres conceptos importantes
    - Localización (Local receptive field)
    - Pesos compartidos (shared weights)
    - Aggregación (pooling)



Localización

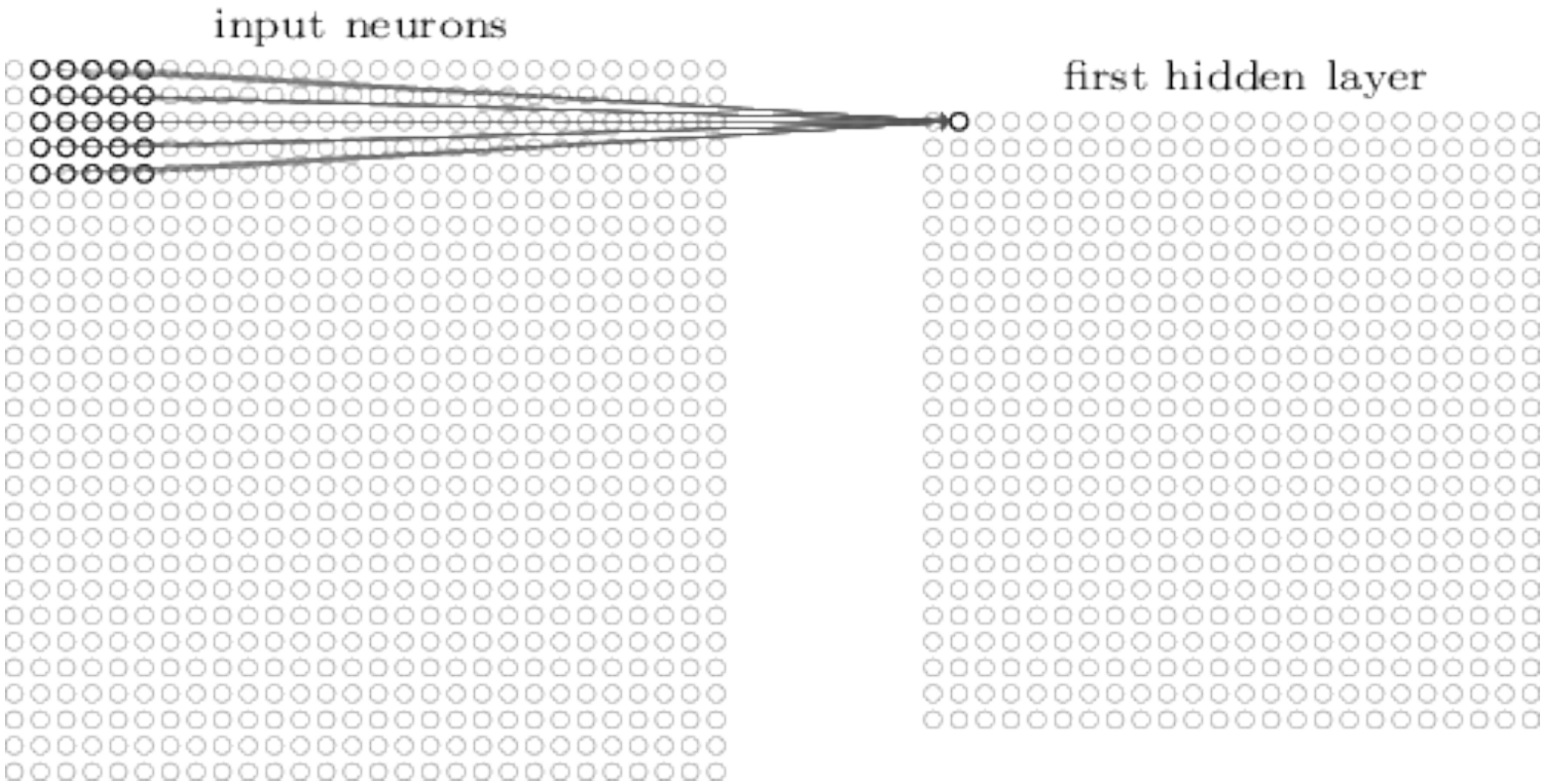
- Solo conectamos algunas neuronas de de una capa con algunas neuronas de la siguiente
  - Hacemos grupos de neuronas relacionadas
    - Como se trata de imágenes vamos a organizarlas en 3 dimensiones: alto, ancho y color (canal)
  - La salida de estas neuronas son la entrada a una neurona de la capa siguiente

Localización (imágenes tomadas de Michael Nielsen http://neuralnetworksanddeeplearning.com/index.html)

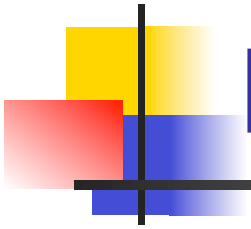


La ventana de 5 x 5 de desliza a través de toda la entrada.

Localización



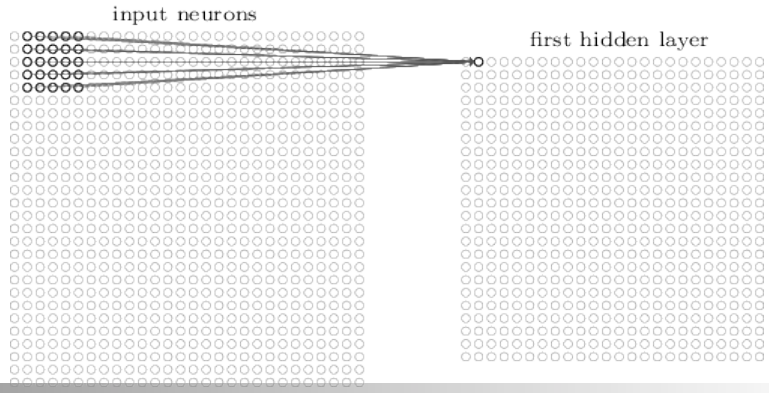
El tamaño del deslizamiento, el paso, se llama *stride length*



Localización

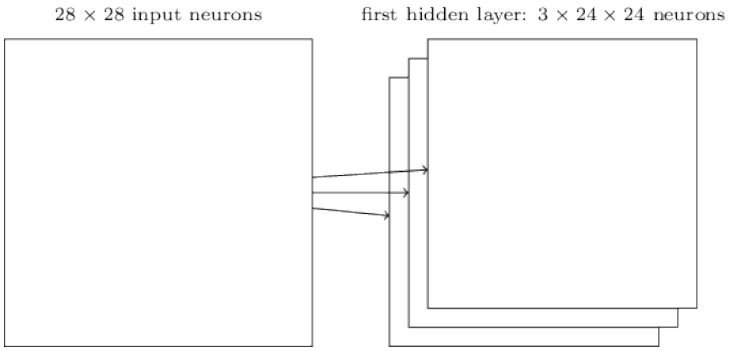
- Si tenemos n * n neuronas en la capa de entrada y una ventana de m*m con paso s
  - Cuántas neuronas hay en la siguiente capa?
    - k ventanas horizontales y k verticales
    - k= (1 + ceil((n-m)/stride)) x (1 + ceil((n-m)/stride))
    - Para 12 x 12, m=3, stride 2
      - 1+ ceil(9/2)= 6
    - Si te pasas rellenas
- Si queremos que el número de neuronas en esta capa sea igual que en la anterior. Rellenas con ceros

Pesos compartidos



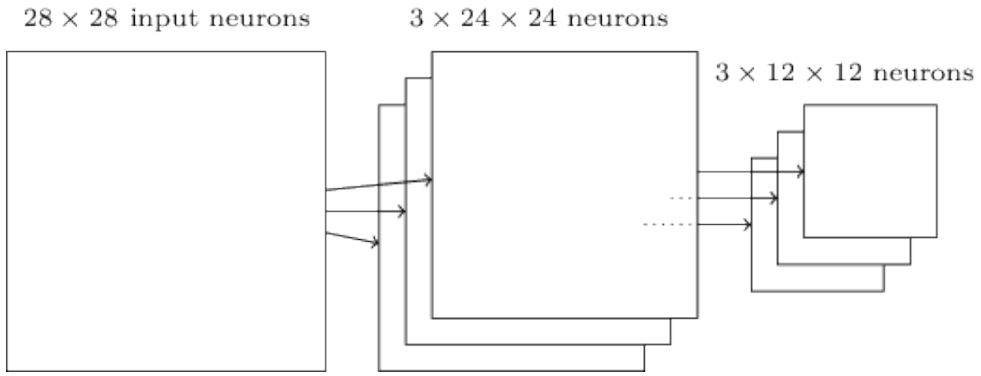
- Cada neurona de la capa intermedia se conecta con m x m neuronas de la capa anterior
  - Tiene m x m pesos más el sesgo (w_0)
- Todas las k x k neuronas de esta capa intermedia van a compartir los mismos pesos!
  - Todas las neuronas van a detectar el mismo patrón pero en diferentes ubicaciones (tiene sentido en imágenes)
  - A estos pesos compartidos se les conoce como kernel o filtro

Capas paralelas



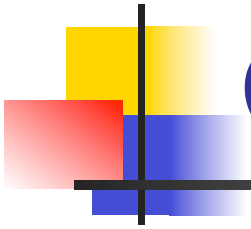
- De esta forma una de estas capas intermedias, o filtro, detecta un rasgo
- Necesitamos incluir muchas de estas capas, todas conectadas a la capa de entrada

Agregación



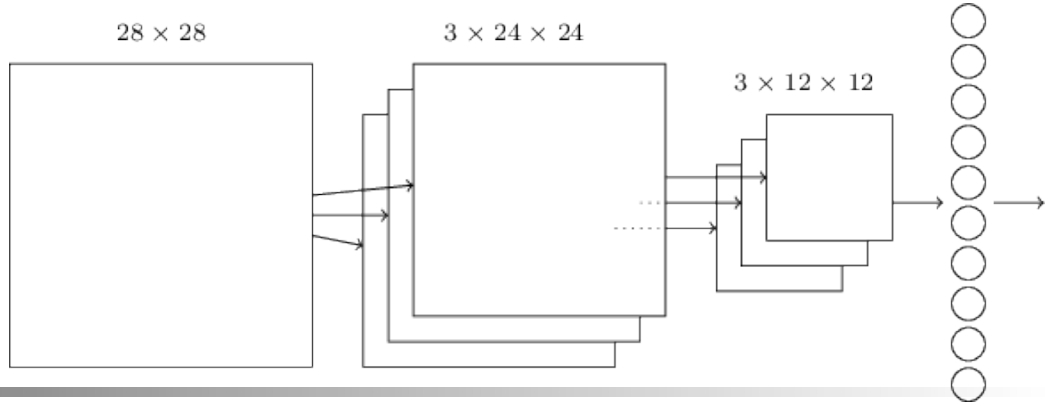
- Esta capa se encuentra inmediatamente después de la capa de localización
  - De manera similar a la capa de localización toma un subconjunto (ventana) de neuronas y produce un resultado
  - Reduce el número de salidas
    - Si la ventana es de 2 x 2 con paso 2 reduce a la mitad
  - La reducción suele ser algo simple como tomar el máximo de las salidas de su ventana

Topología de redes Convolucionales

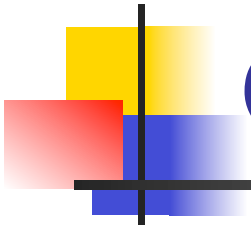


- Capa de entrada
- Capas Convolucionales (varias)
  - Agregación y max pooling
- Capa de salida

Capa de salida



- Al final se configura una capa de salida tradicional, conectando todas las salidas de la capa de agregación anterior a todas las neuronas de salida
- Al final esta red tiene una estructura feed forward y podemos usar, básicamente, el algoritmo de retropropagación que vimos en clase



Otras Arquitecturas

- Máquinas de Boltzman
- Deep belief networks
- LTSM