Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería de Transporte y Logística

Sistemas Urbanos Inteligentes

Maquinaria de Deep Learning

Hans Löbel
Dpto. Ingeniería de Transporte y Logística
Dpto. Ciencia de la Computación

En teoría, no hay diferencia entre la teoría y la práctica.
En la práctica, sí la hay.

Tras bambalinas, hacer que los modelos de DL funcionen adecuadamente, y sean prácticos, requiere una gran cantidad de detalles:

- ¿Cómo optimizar?
- ¿Cómo controlar el *overfitting*?

En teoría, no hay diferencia entre la teoría y la práctica.
En la práctica, sí la hay.

Tras bambalinas, hacer que los modelos de DL funcionen adecuadamente, y sean prácticos, requiere una gran cantidad de detalles:

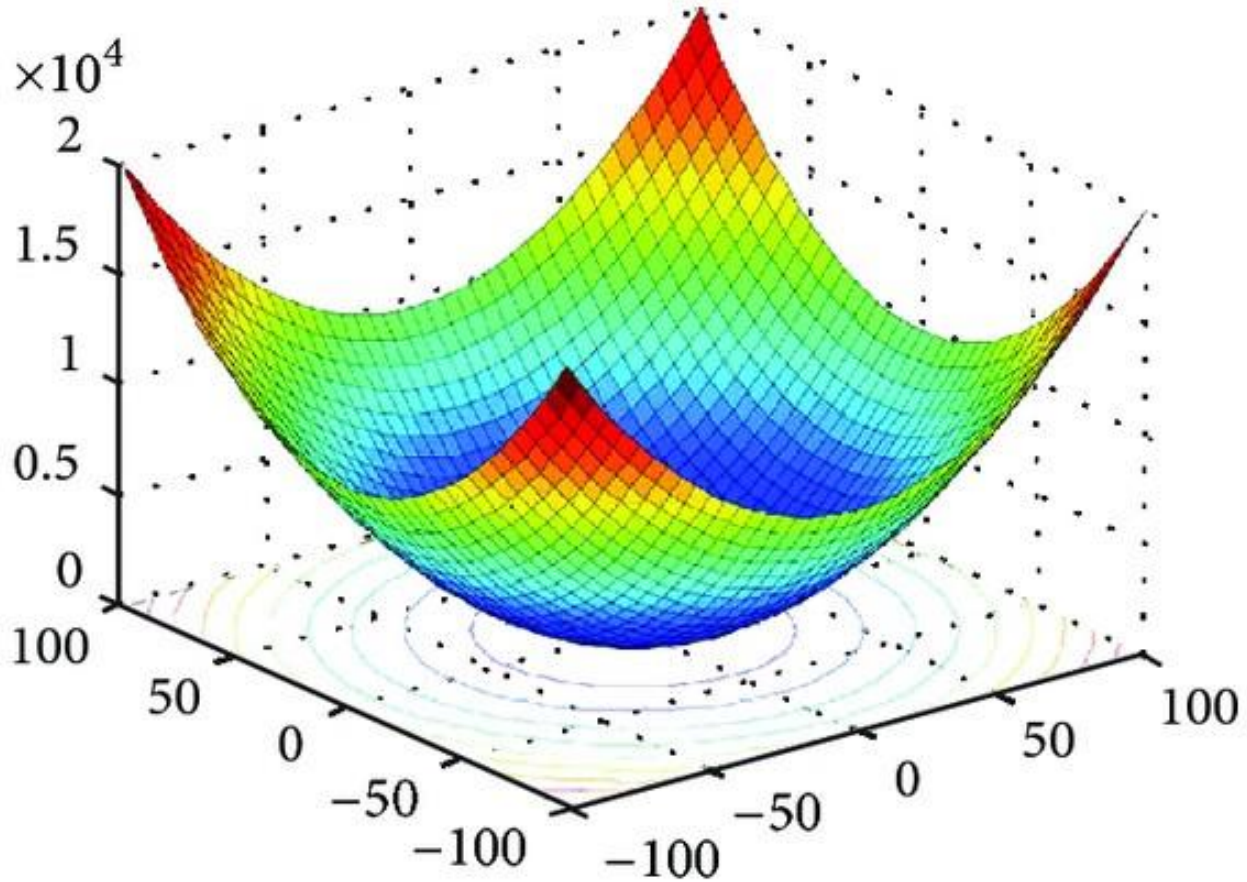
- ¿Cómo optimizar?
- ¿Cómo controlar el *overfitting*?

No olvidemos para qué estamos optimizando

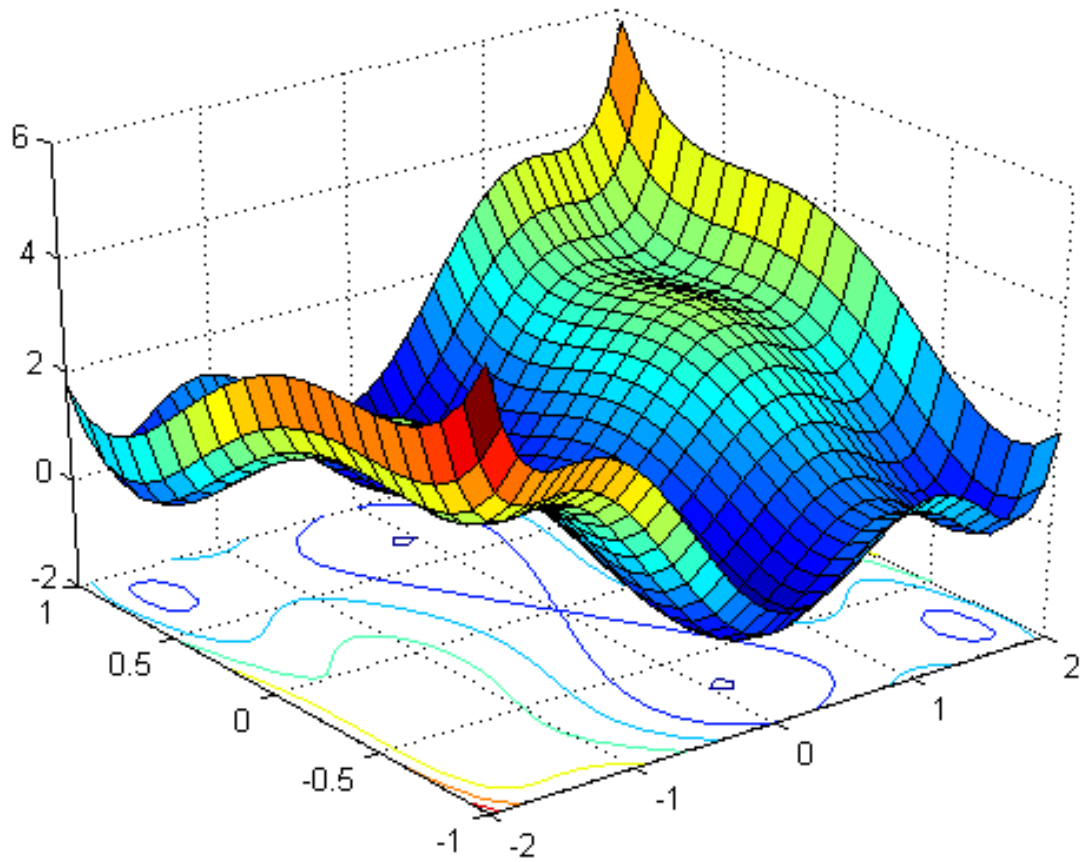
Para obtener los parámetros de una red a través de un proceso de optimización, debemos considerar bastantes elementos:

- ¿Cómo desciendo en la función objetivo?
- ¿Dónde me paro inicialmente en la función a minimizar?
- ¿Debo normalizar las entradas y features?

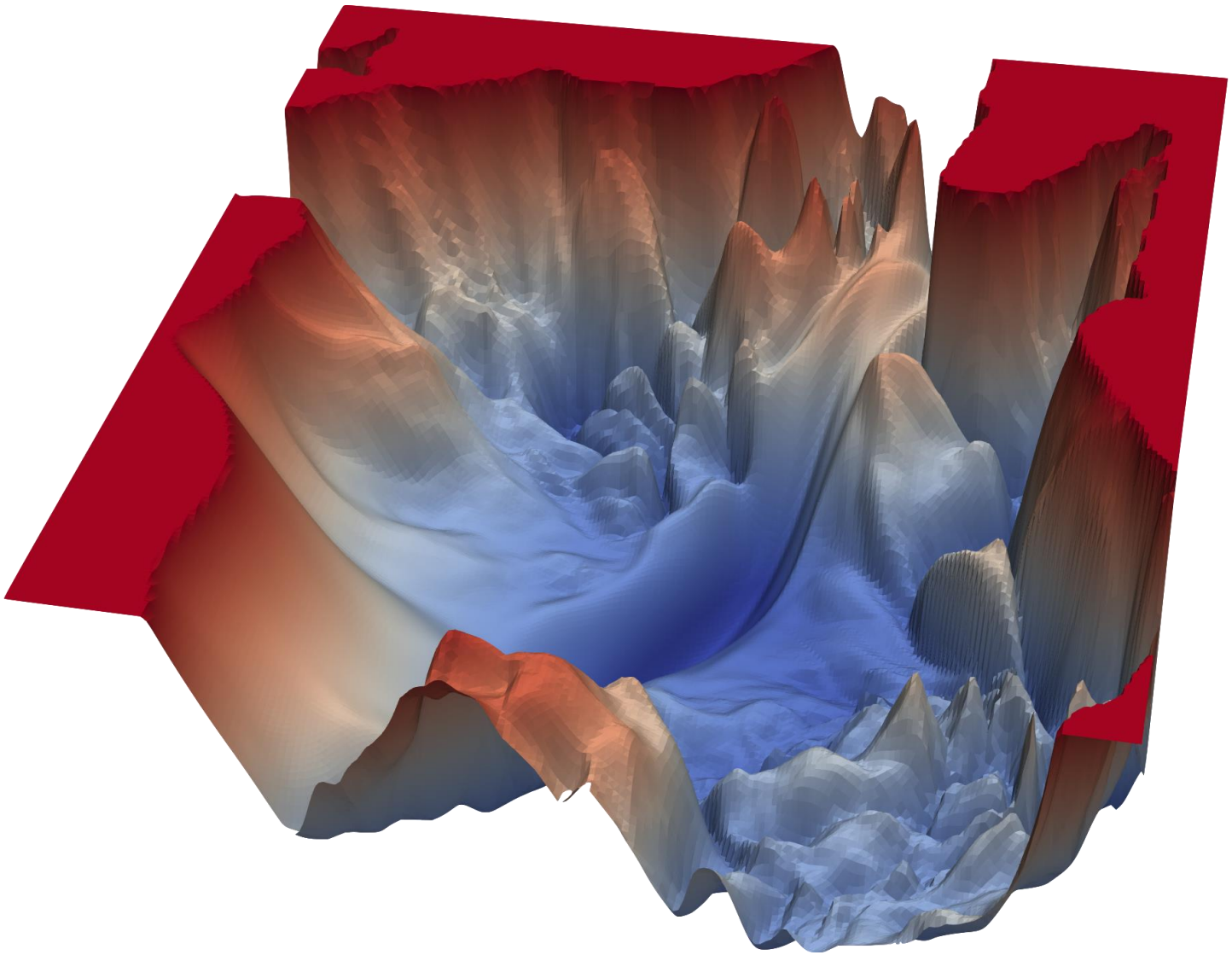
Debido a la estructura de las redes, función objetivo no es convexa



Ni siquiera es no convexa pero suave



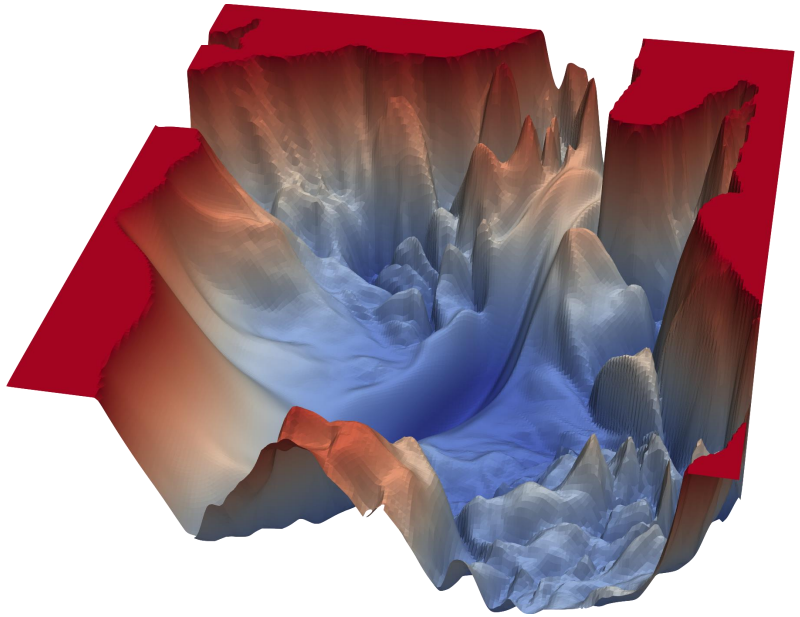
La realidad es mucho peor (en otras palabras, el descenso de gradiente básico no sirve)



Li et al., “Visualizing the Loss Landscape of Neural Nets”, NeurIPS 18

Algunos de los desafíos que estas superficies presentan

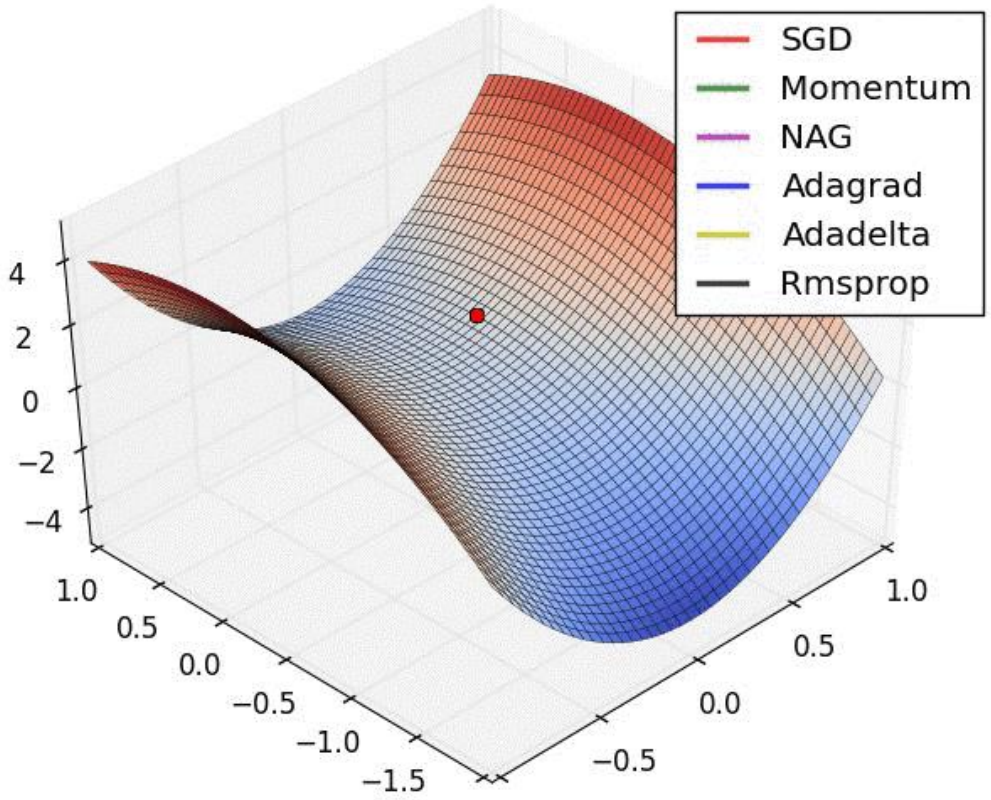
- Costo computacional obliga a calcular gradientes en un batch (SGD vs GD)
- Elección de un learning rate adecuado
- Ajuste dinámico del learning rate (scheduling)
- Escala del learning rate para cada parámetro
- Y lógicamente escapar de puntos críticos y mínimos locales



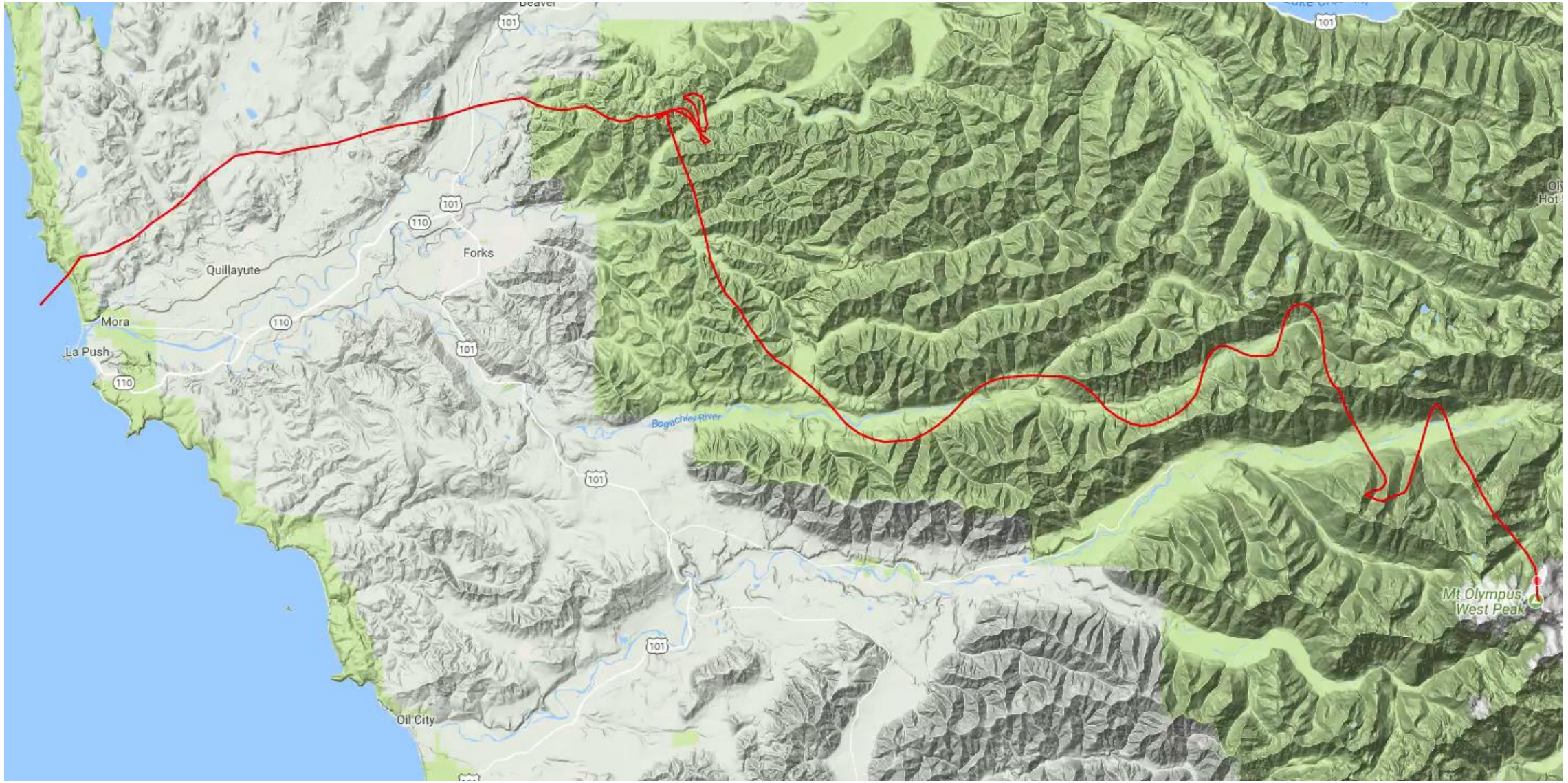
Li et al., “Visualizing the Loss Landscape of Neural Nets”, NeurIPS 18

En la práctica, mecanismos con momentum y learning rate adaptativo son suficientes en la mayoría de los casos

- Momentum permite aminorar riesgo de quedar pegado en puntos críticos, al conservar "velocidad" de descenso: se pondera nuevo gradiente con dirección de descenso previa.
- Learning rate adaptativo se calcula antes de cada descenso en base a estadísticos acumulados en el trayecto, y se aplica uno distinto por cada parámetro.
- Recomendación: usar algoritmo Adam con los parámetros por defecto



Un ejemplo muy “bonito” puede construirse utilizando datos geográficos de elevación



[1]https://fosterelli.co/executing-gradient-descent-on-the-earth

No olvidemos para qué estamos optimizando

Para obtener los parámetros de una red a través de un proceso de optimización, debemos considerar bastantes elementos:

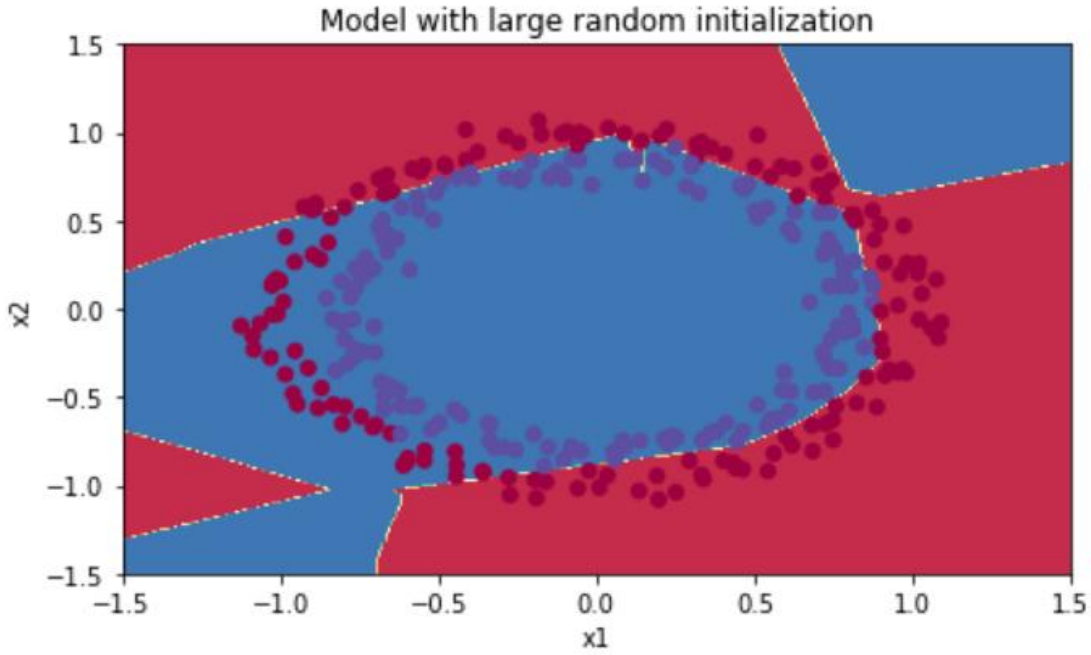
- ¿Cómo desciendo en la función objetivo?
- ¿Dónde me paro inicialmente en la función a minimizar?
- ¿Debo normalizar las entradas y features?

Inicialización de parámetros

- ¿Sirve inicializar todo con 0s?

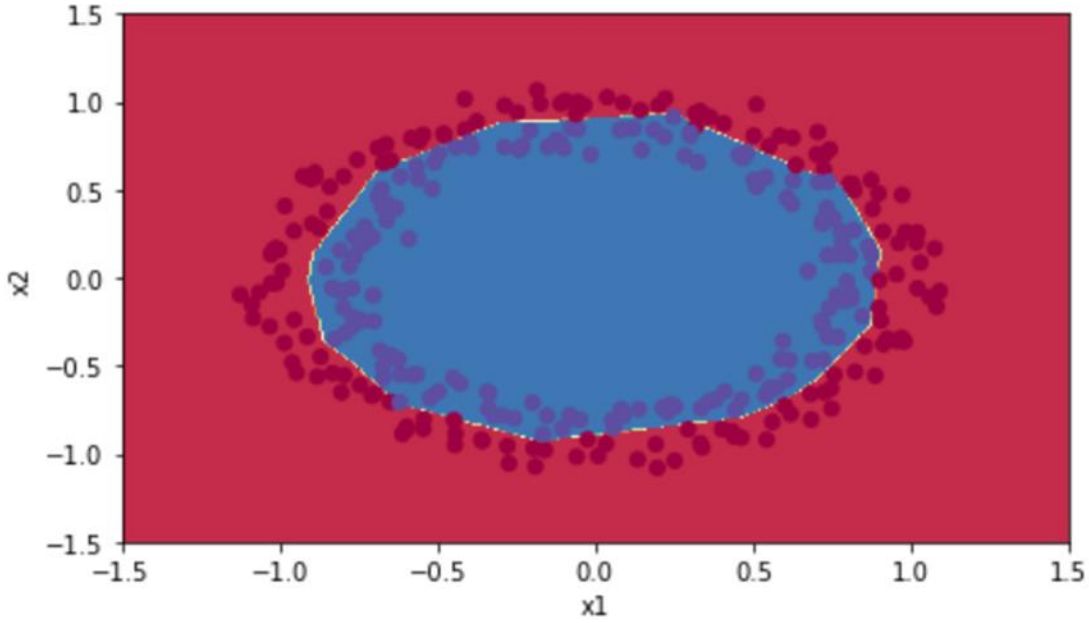
Inicialización de parámetros

- ¿Sirve inicializar todo con 0s?
- Lo más básico que se puede hacer entonces, es inicializar aleatoriamente.



Inicialización de parámetros

- Existen dos problemas con el esquema aleatorio:
  1. Tiempo de convergencia
  2. Desvanecimiento/explosión del gradiente
- Para evitar esto, se corrige la inicialización aleatoria, considerando el tamaño de las capas:



$\sqrt{\frac{2}{size^{[l-1]}}}$	$\sqrt{\frac{1}{size^{[l-1]}}}$	$\sqrt{\frac{2}{size^{[l-1]} + size^{[l]}}}$

No olvidemos para qué estamos optimizando

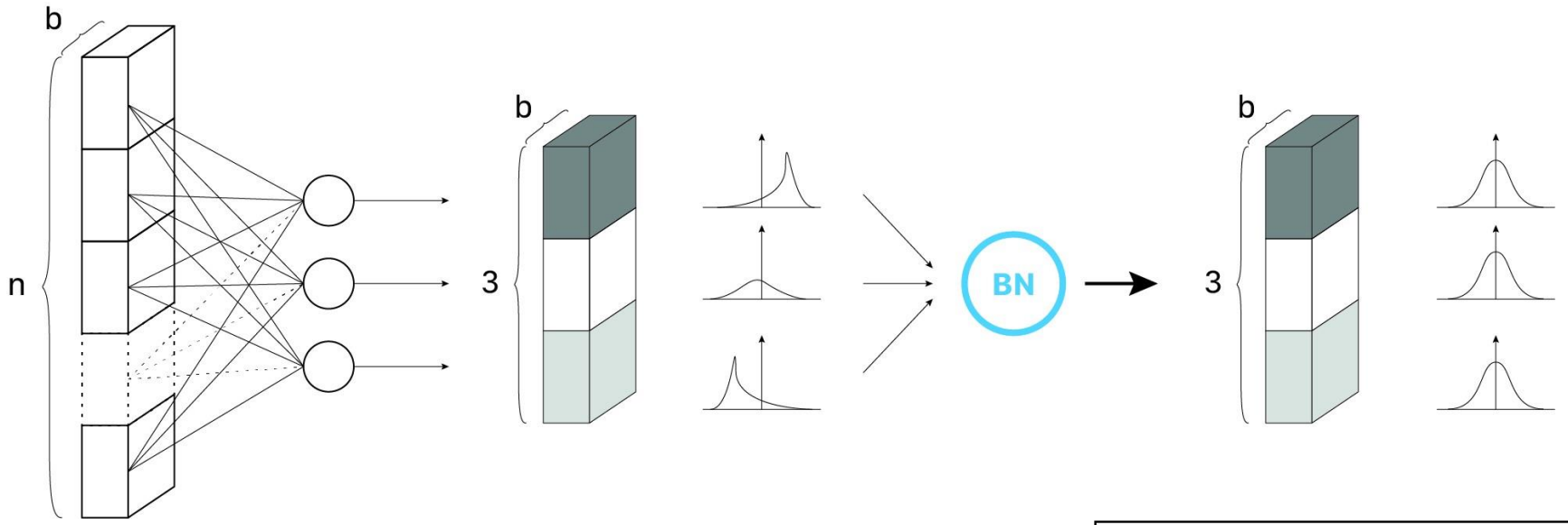
Para obtener los parámetros de una red a través de un proceso de optimización, debemos considerar bastantes elementos:

- ¿Cómo desciendo en la función objetivo?
- ¿Dónde me paro inicialmente en la función a minimizar?
- ¿Debo normalizar las entradas y features?

Capas de normalización

- En general, mientras más similares sean las distribuciones de las features, más rápida la convergencia de los pesos.
- En base a esto, normalizar las entradas a la red es una buena práctica y siempre debe considerarse.
- Pero, ¿cómo inducimos esto dentro de la red?

Fiel al mantra del curso, dejamos que la red aprenda a normalizar las activaciones de sus neuronas



- Una vez normalizadas las activaciones, las escalamos y trasladamos (parámetros aprendidos) para adecuarlas a la siguiente capa.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

En teoría, no hay diferencia entre la teoría y la práctica.
En la práctica, sí la hay.

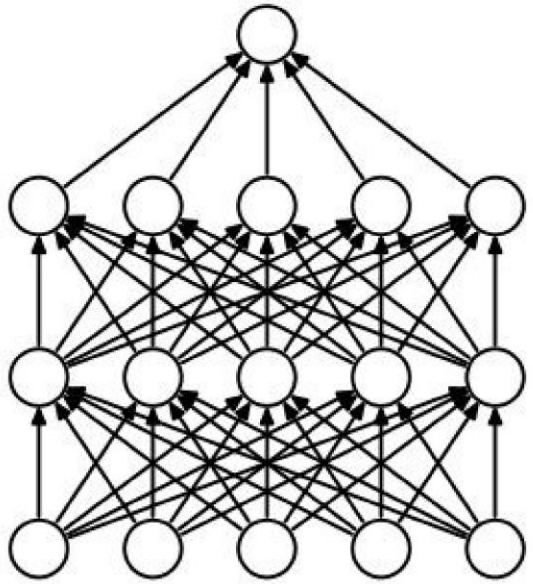
Tras bambalinas, hacer que los modelos de DL funcionen adecuadamente, y sean prácticos, requiere una gran cantidad de detalles:

- ¿Cómo optimizar?
- ¿Cómo controlar el *overfitting*?

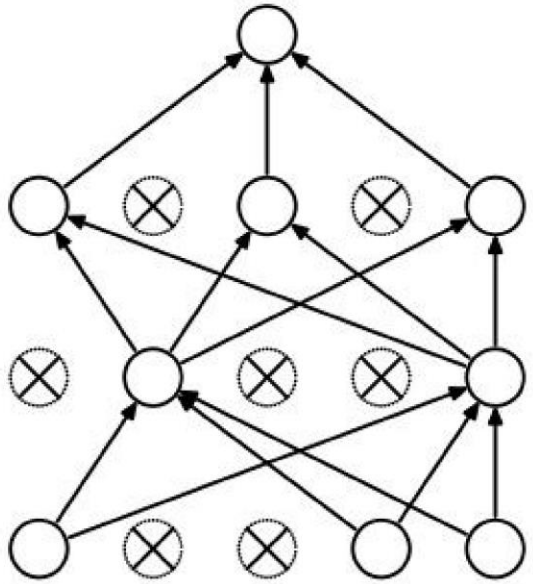
Dropout evita memorización de la activación conjunta de neuronas

Regularization: **Dropout**

"randomly set some neurons to zero in the forward pass"



(a) Standard Neural Net

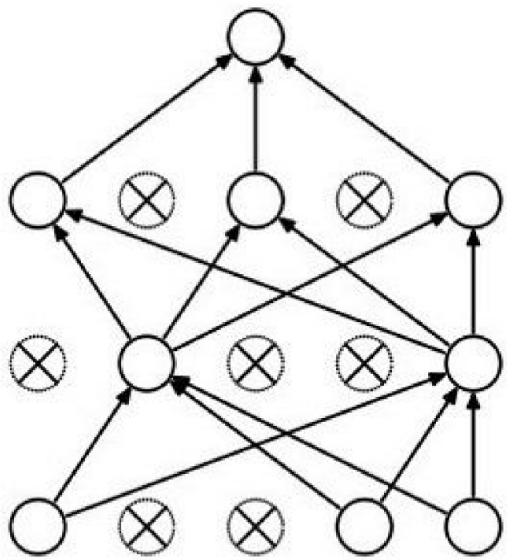


(b) After applying dropout.

[Srivastava et al., 2014]

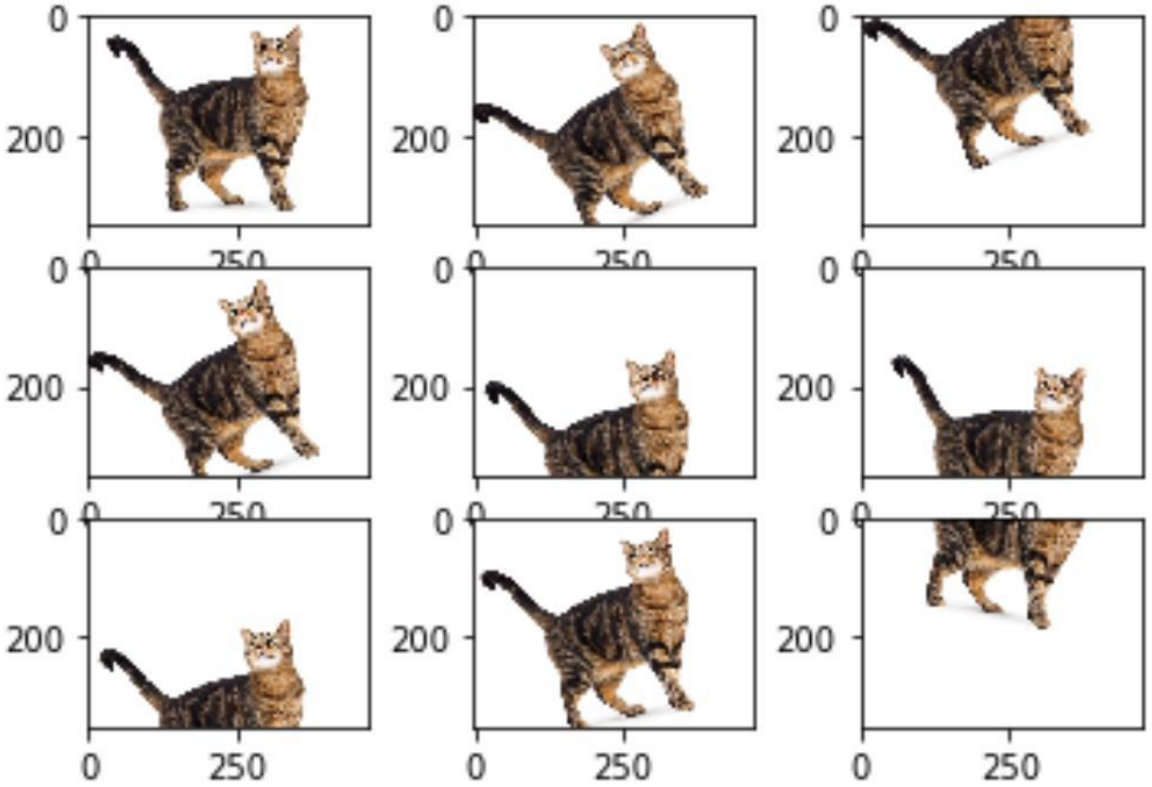
Dropout evita memorización de la activación conjunta de neuronas

Waaaait a second…
How could this possibly be a good idea?

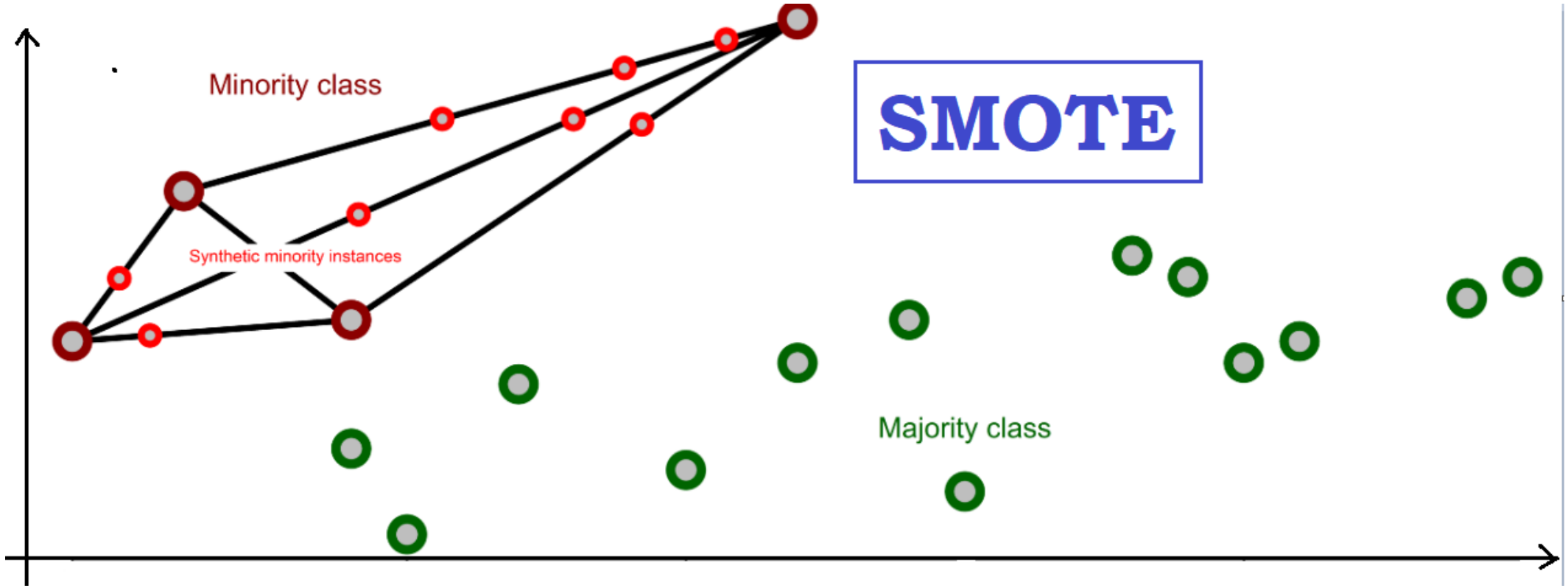


Forces the network to have a redundant representation.

Aumento de datos incrementa el set de datos “gratuitamente”, subiendo la dificultad del problema (bias-variance tradeoff)



Otras veces es necesario hacer para balancear la cantidad de ejemplos por clase



Vamos a Colab...

Cuál es EL CONCEPTO CENTRAL de esta clase

- Para entrenar redes hay que considerar muchos elementos prácticos
- Afortunadamente, no es necesario reinventar la rueda y muchas veces es suficiente basarse trabajos anteriores.

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería de Transporte y Logística

Sistemas Urbanos Inteligentes

Maquinaria de Deep Learning

Hans Löbel
Dpto. Ingeniería de Transporte y Logística
Dpto. Ciencia de la Computación