

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Posgrado en Ciencia e Ingeniería de la Computación

APRENDIZAJE PROFUNDO

Redes densas con PyTorch

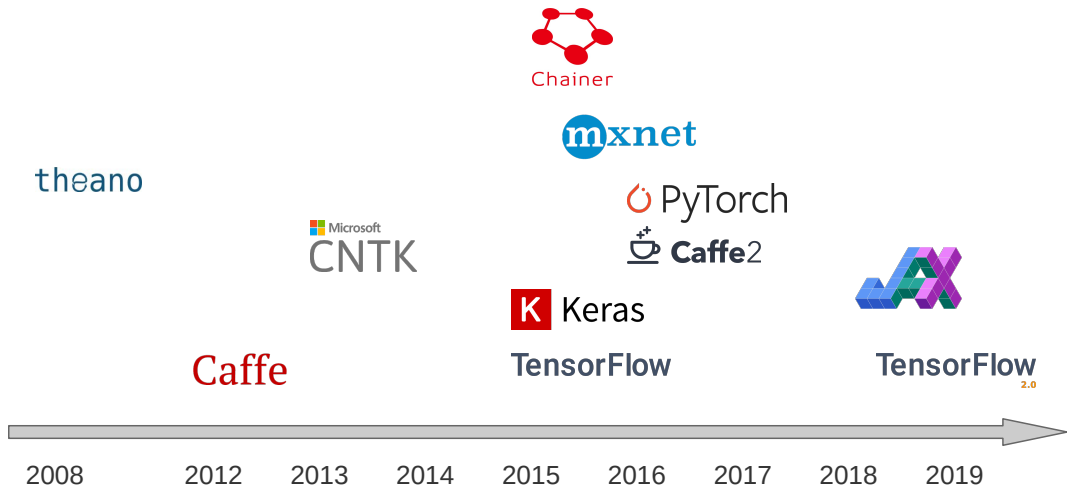
Bere & Ricardo Montalvo Lezama

Octubre 2020

Bibliotecas para aprendizaje profundo

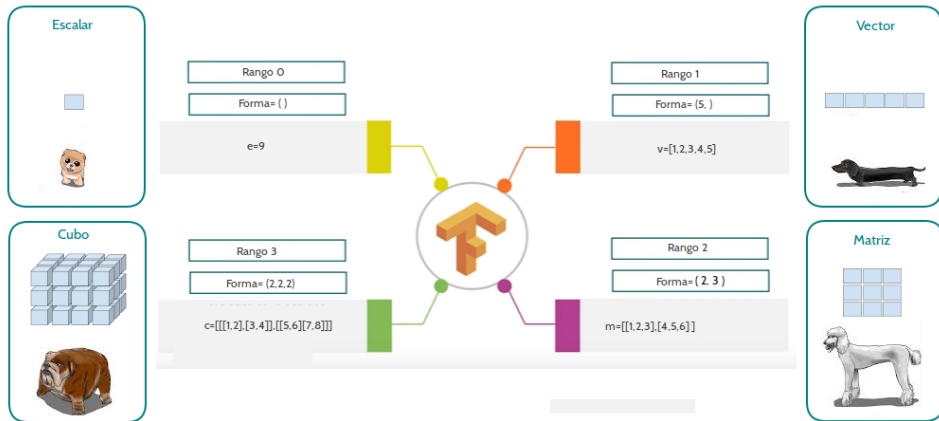
- ▶ Modelo de programación sencillo de álgebra lineal.
- ▶ Abstracciones comunes de redes neuronales.
- ▶ Ejecución en CPU y GPU.
- ▶ Diferenciación automática y optimizadores.
- ▶ Visualización, serialización, trazas, distribución.

Historia



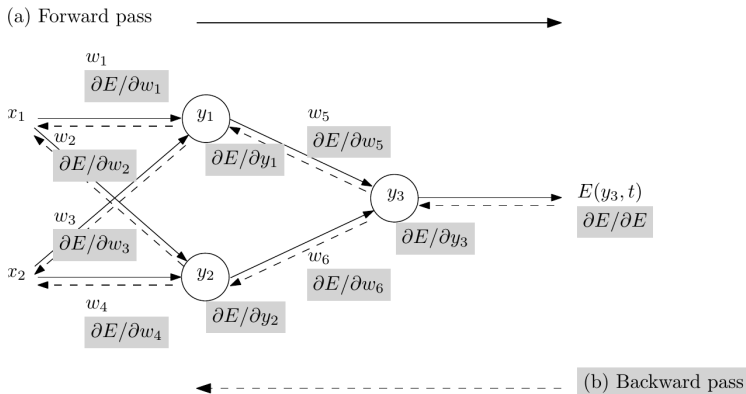
Tensores

- Un tensor es un arreglo multidimensional.



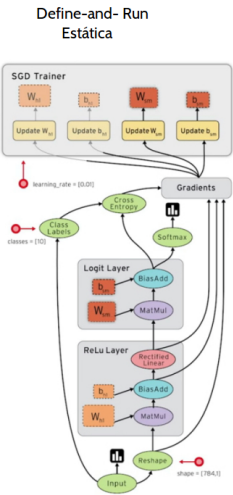
Gráficas de cómputo

- Representación de una composición de funciones donde las variables son tensores (datos y parámetros).



Fuente: Automatic Differentiation in Machine Learning: a Survey

Estáticas vs dinámicas

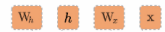


Define-by-Run
Dinámica

A graph is created on the fly

```
from torch.autograd import Variable

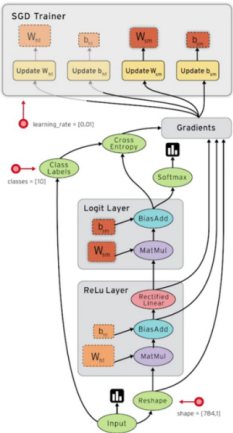
x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))
```



Fuente: tensorflow.org

Estáticas vs dinámicas

Define-and- Run
Estática



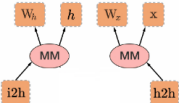
Define-by-Run
Dinámica

A graph is created on the fly

```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))

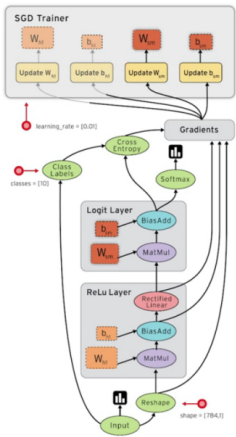
i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
```



Fuente: tensorflow.org

Estáticas vs dinámicas

Define-and- Run
Estática



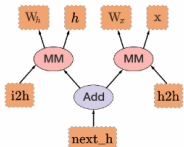
Define-by-Run
Dinámica

A graph is created on the fly

```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))

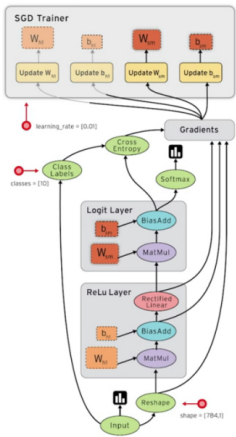
i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
next_h = i2h + h2h
```



Fuente: tensorflow.org

Estáticas vs dinámicas

Define-and- Run
Estática



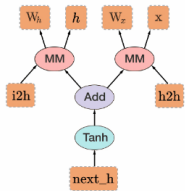
Define-by-Run
Dinámica

A graph is created on the fly

```
from torch.autograd import Variable

x = Variable(torch.randn(1, 10))
prev_h = Variable(torch.randn(1, 20))
W_h = Variable(torch.randn(20, 20))
W_x = Variable(torch.randn(20, 10))

i2h = torch.mm(W_x, x.t())
h2h = torch.mm(W_h, prev_h.t())
next_h = i2h + h2h
next_h = next_h.tanh()
```



Fuente: tensorflow.org

Gráficas y bibliotecas

► Estática

theano

Caffe

Microsoft
CNTK



TensorFlow 1.0

Caffe2

► Dinámica

PYTORCH

Chainer

Eager
execution



TensorFlow
2.0

Interfaces de PyTorch

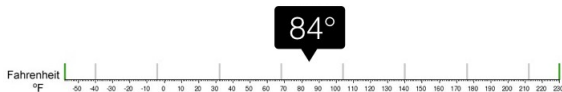
- ▶ Alto Nivel
 - ▶ Apilando capas con `nn.Sequential`.
- ▶ Medio Nivel
 - ▶ Heredando de `nn.Module`.
- ▶ Bajo Nivel
 - ▶ Usando primitivas y definiendo parámetros.

Clasificación vs Regresión



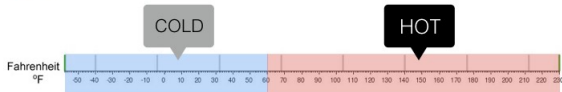
Regresión

¿Cuál será la temperatura del día de mañana ?

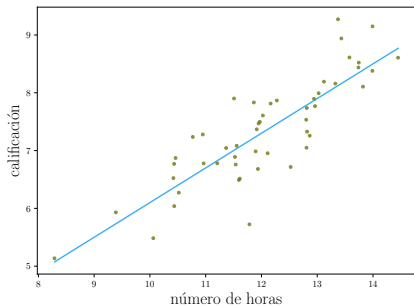


Clasificación

¿El día de mañana será caluroso o frío ?



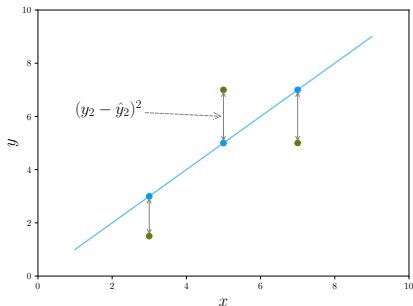
Regresión lineal simple (I)



► Modelo:

$$\hat{y} = wx + b$$

Regresión lineal simple (II)



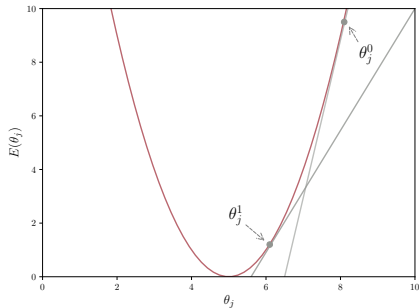
► Modelo:

$$\hat{y} = wx + b$$

► Función de error:

$$E(w, b) = \frac{1}{2} \sum_{i=1}^n (y - \hat{y})^2$$

Regresión lineal simple (III)



- Modelo:

$$\hat{y} = wx + b$$

- Función de error:

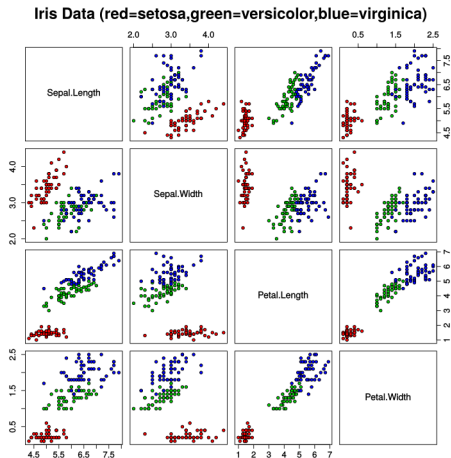
$$E(w, b) = \frac{1}{2} \sum_{i=1}^n (y - \hat{y})^2$$

- Optimización de la función de error:

$$w := w - \alpha \frac{\partial}{\partial w} E(w, b)$$

$$b := b - \alpha \frac{\partial}{\partial b} E(w, b)$$

Conjunto de datos Iris



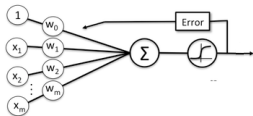
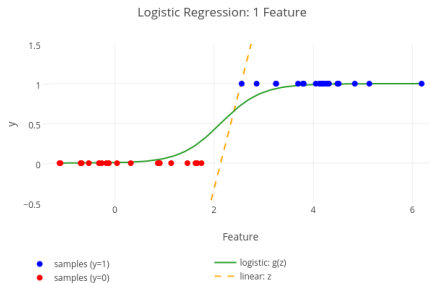
Conjunto de datos de tres especies de Iris.

- ▶ Setosa.
- ▶ Versicolour.
- ▶ Virginica.

Atributos:

- ▶ Largo Sépalo.
- ▶ Ancho Sépalo.
- ▶ Largo Pétalo.
- ▶ Ancho Sépalo.

Regresión logística



- Modelo:

$$P(y|x_i, \theta) = \hat{y} = f \frac{1}{1 + e^{wx+b}}$$

- Función de error: entropía cruzada binaria.

$$E = -\frac{1}{m} \sum_{i=1}^m (y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$$

- Optimización de la función de error:

$$w := w - \alpha \frac{\partial}{\partial w} E(w, b)$$

$$b := b - \alpha \frac{\partial}{\partial b} E(w, b)$$

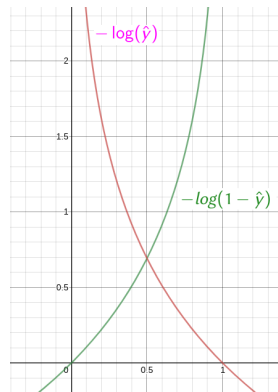
- Métrica: exactitud.

$$Ex = \frac{\# \text{ predicciones correctas}}{\# \text{ total de predicciones}}$$

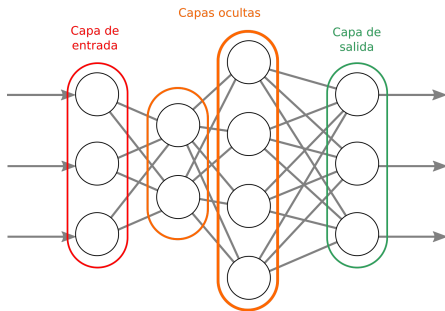
Entropía cruzada binaria.

$$E = (y)(-\log(\hat{y})) + (1 - y)(-\log(1 - \hat{y}))$$

Etiqueta y	Predicción \hat{y}	Entropía binaria	Pérdida
0	0.9	2.303	alta
0	0.1	0.105	baja
1	0.9	0.105	baja
1	0.1	2.303	alta



Perceptrón multicapa



Entrenamiento:

- ▶ Paso 1: carga el lote de entrenamiento datos.
- ▶ Paso 2: propagación hacia adelante para obtener las predicciones y calcular el error.
- ▶ Paso 3: retropropagar el error.
- ▶ Paso 4: actualizar los pesos.

- ▶ Activación de la j -ésima neurona de la i -ésima capa:

$$a_j^i = f\left(\sum_{i=1}^n x_i w_i + b\right).$$

- ▶ Función de error: entropía cruzada categórica.

$$E = -\frac{1}{m} \sum_{i=1}^m (y \log(\hat{y}))$$

- ▶ Optimización de la función de error:

$$w := w - \alpha \frac{\partial}{\partial w} E(w, b)$$

$$b := b - \alpha \frac{\partial}{\partial b} E(w, b)$$

- ▶ Métrica: exactitud.

$$Ex = \frac{\# \text{ predicciones correctas}}{\# \text{ total de predicciones}}$$

Ventajas de PyTorch

- ▶ Arquitecturas, modelos preentrenados y conjuntos de datos.
 - ▶ torchvision, torchtext, torchaudio.
- ▶ Visualización y entrenamiento.
 - ▶ Tensorboard, skorch, Ignite, Lightning.
- ▶ Dominios especializados.
 - ▶ AllenNLP (lenguaje), BoTorch (optimización bayesiana), DGL (gráficas).
- ▶ Precisión mixta e interoperabilidad.
 - ▶ Apex/AMP, ONNX.