Extras:

Bash Normalization, Dropout y Cross Entropy
Aprendizaje profundo

Dr. Juan Bekios Calfa

Magíster en *Data Science*
Facultad de Ingeniería y Ciencias

UNIVERSIDAD ADOLFO IBÁÑEZ

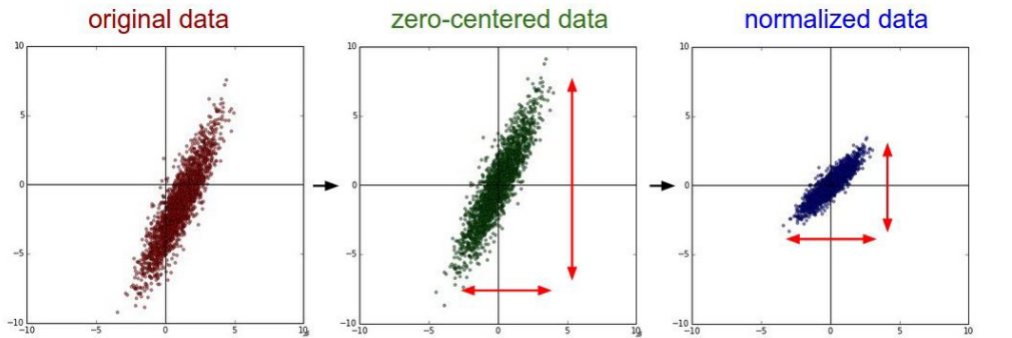
Información de Contacto

- Juan Bekios Calfa
  - email: juan.bekios@edu.uai.cl
  - Web page: http://jbekios.ucn.cl
  - Teléfono: 235(5162) - 235(5125)

UNIVERSIDAD ADOLFO IBÁÑEZ

Contenidos

Preprocesamiento de datos



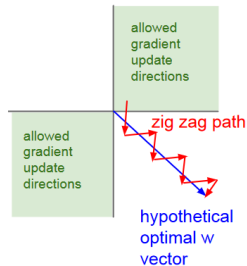
(Asuma que X $[N \times D]$ es una matriz de datos, cada fila es un ejemplo)

Preprocesamiento de datos

Consideremos lo que sucede cuando la entrada a una neurona es siempre positiva

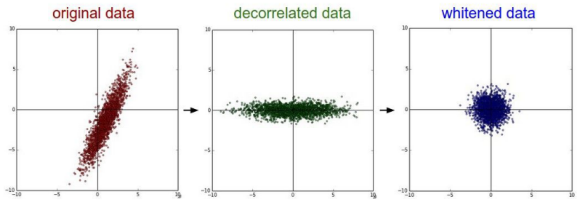
$$f(\sum_i w_i x_i + b)$$

¿Que podemos decir de los gradientes en **w**?
¿Siempre positivos o negativos?



Preprocesamiento de datos

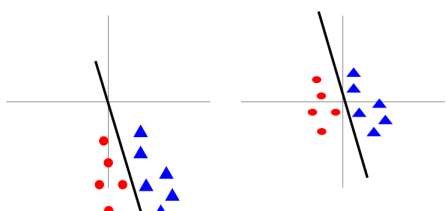
En la práctica, es posible ver el uso de ACP (**PCA**) y blanqueamiento (**Whitening**) de los datos.



Preprocesamiento de datos

Antes de la normalización: pérdida de clasificación muy sensible a cambios en la matriz de peso, difícil de optimizar

Después de la normalización: menos sensible a pequeños cambios de peso, más fácil de optimizar



Resumen: En la práctica para imágenes

Por ejemplo: Consideremos CIFAR-10 con imágenes de [32,32,3]

- Restar la imagen promedio (arreglo de [32,32,3])
- Restar el promedio por canal (3 números)
- Restar el promedio por canal y dividirlo por la desviacion estandar por canal (3 números)

Batch normalization

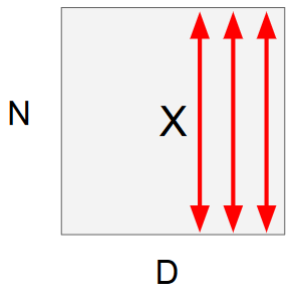
¿Quieres activaciones de varianza unitaria de media cero?

Consideremos un *batch* de activaciones en alguna capa. Para hacer que cada dimensión varíe la unidad de media cero, aplique:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch normalization

Input: $x : N \times D$



$\mu_j = \frac{1}{N} \sum_{i=1}^{N} x_{i,j}$ Promedio por canal, [D]

$\sigma_j^2 = \frac{1}{N} \sum_{i=1}^{N} (x_{i,j} - \mu_j)^2$ Varianza por canal [D]

$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$ x normalizado, [N,D]

¿Qué pasaría si el uso de media cero y variancia unitaria es una restricción demasiada fuerte?

Batch normalization

Input: $x : N \times D$ Parámetros de cambio y escala que se pueden aprender: $\gamma, \beta : D$
Aprendiendo los parámetros $\gamma = \sigma, \beta = \mu$ permitirá recuperar la función identidad

$\mu_j = \frac{1}{N} \sum_{i=1}^{N} x_{i,j}$ Promedio por canal, [D]
$\sigma_j^2 = \frac{1}{N} \sum_{i=1}^{N} (x_{i,j} - \mu_j)^2$ Var por canal [D]
$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$ x normalizado, [N,D]
$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$ Salida, [N,D]

Batch normalization - durante evaluación

Input: $x : N \times D$ Parámetros de cambio y escala que se pueden aprender: $\gamma, \beta : D$
Durante la evaluación de la red *batchnorm* se transforma en un operador lineal μ_j =Promedio de los valores vistos durante el entrenamiento Promedio por canal, [D]
σ_j^2 =Promedio de los valores vistos durante el entrenamiento Var por canal [D]
$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$ x normalizado, [N,D]
$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$ Salida, [N,D]

Batch Normalization: Pytorch

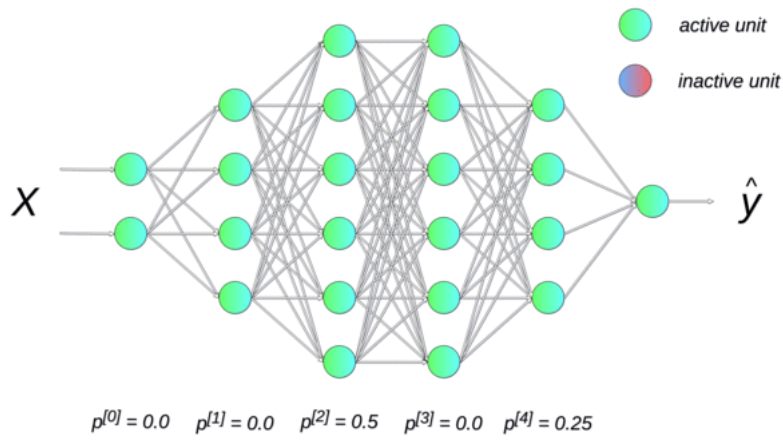
```
class Multilayerpercepron(nn.Module):

  def __init__(self):
    super().__init__()
    self.layers = nn.Sequential(
      nn.Flatten(),
      nn.Linear(32 * 32 * 3, 64),
      nn.BatchNorm1d(64),
      nn.ReLU(),
      nn.Linear(64, 32),
      nn.BatchNorm1d(32),
      nn.ReLU(),
      nn.Linear(32, 10)
    )
```

Dropout

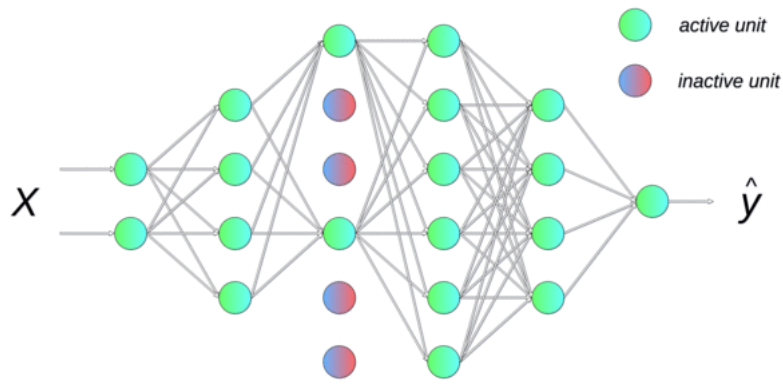
Dropout es un algoritmo para entrenar redes neuronales mediante la eliminación aleatoria de unidades durante el entrenamiento para evitar su coadaptación.

UNIVERSIDAD ADOLFO IBÁÑEZ

Dropout (I)

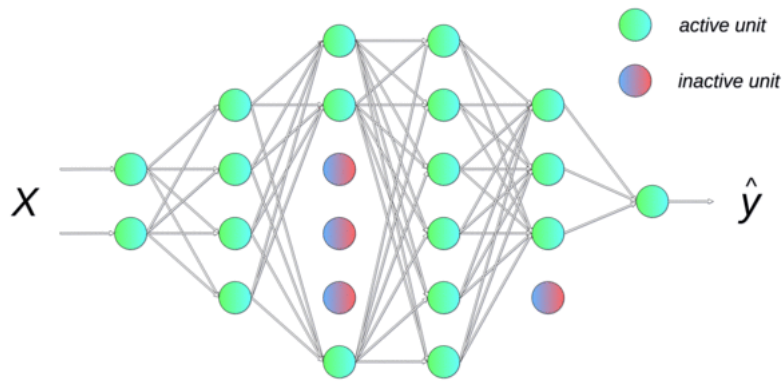


Dropout (II)



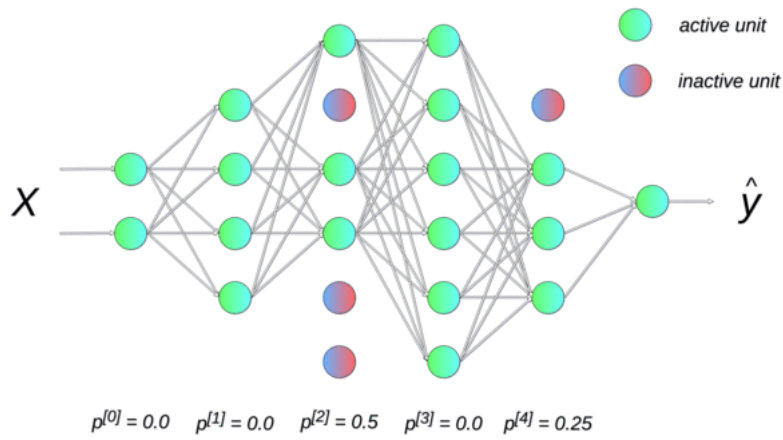
UNIVERSIDAD ADOLFO IBÁÑEZ

Dropout (III)

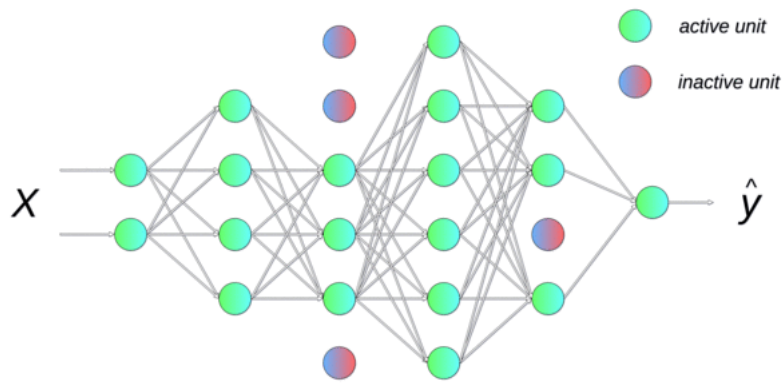


UNIVERSIDAD ADOLFO IBÁÑEZ

Dropout (IV)

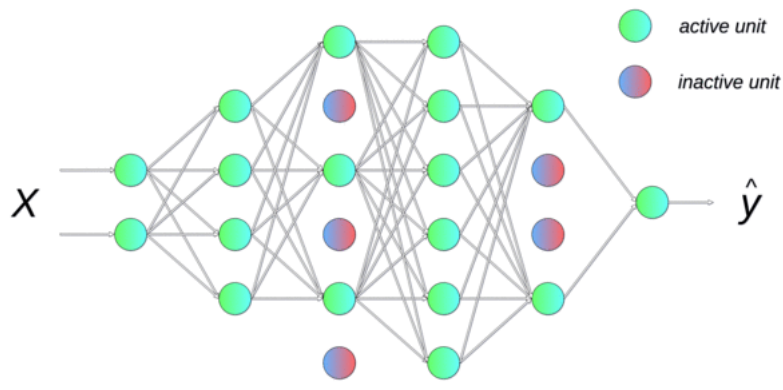


Dropout (V)



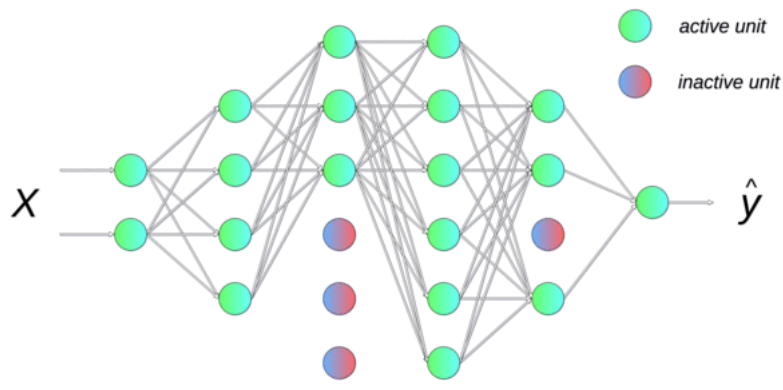
UNIVERSIDAD ADOLFO IBÁÑEZ

Dropout (VI)



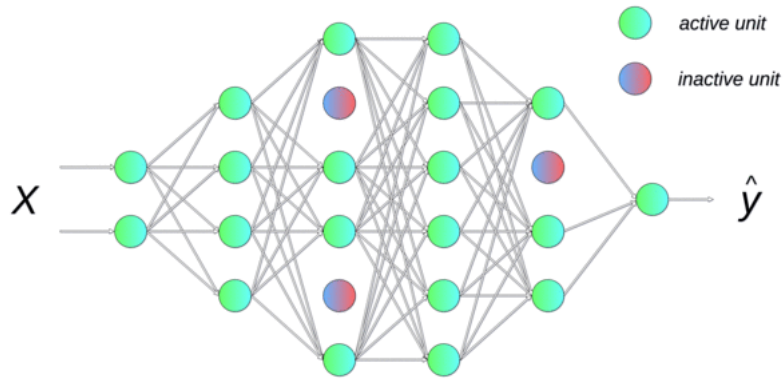
UNIVERSIDAD ADOLFO IBÁÑEZ

Dropout (VII)

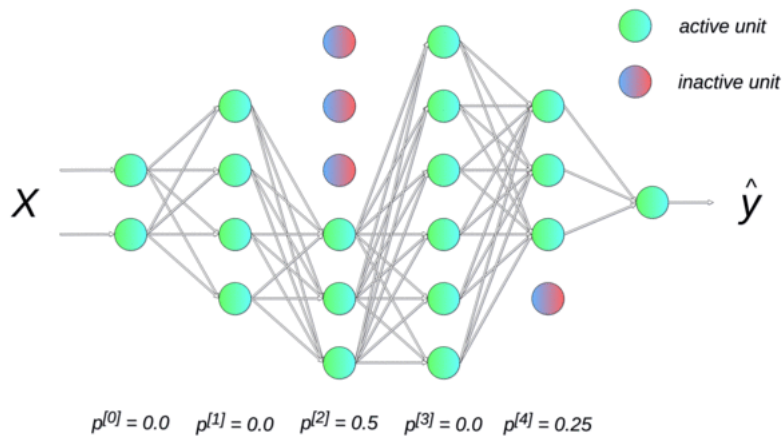


UNIVERSIDAD ADOLFO IBÁÑEZ

Dropout (VIII)

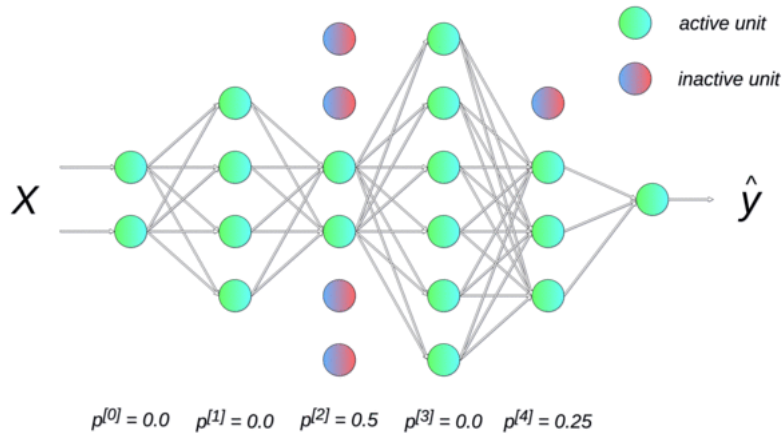


Dropout (IX)



UNIVERSIDAD ADOLFO IBÁÑEZ

Dropout (X)



Dropout: Pytorch

Regularization 1: Dropout

Dropout: in pytorch is implemented as `torch.nn.Dropout`

If we have a network:

```
model = torch.nn.Sequential(
        torch.nn.Linear(1,100), torch.nn.ReLU(),
        torch.nn.Linear(100,50), torch.nn.ReLU(),
        torch.nn.Linear(50,2))
```

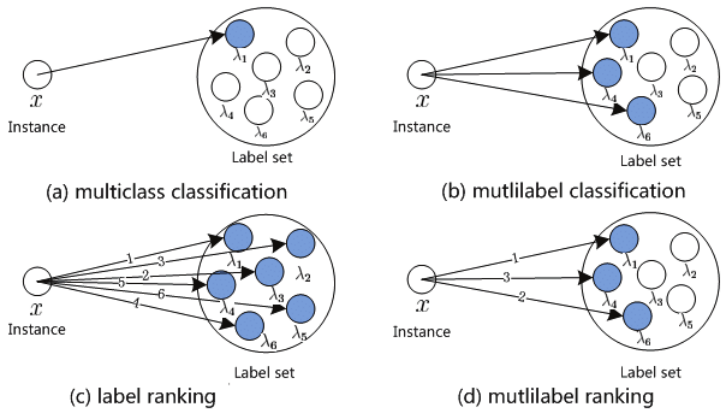
We can simply add dropout layers:

```
model = torch.nn.Sequential(
        torch.nn.Linear(1,100), torch.nn.ReLU(),
        torch.nn.Dropout()
        torch.nn.Linear(100,50), torch.nn.ReLU(),
        torch.nn.Dropout()
        torch.nn.Linear(50,2))
```

Note: A model using dropout has to be set in train or eval model.

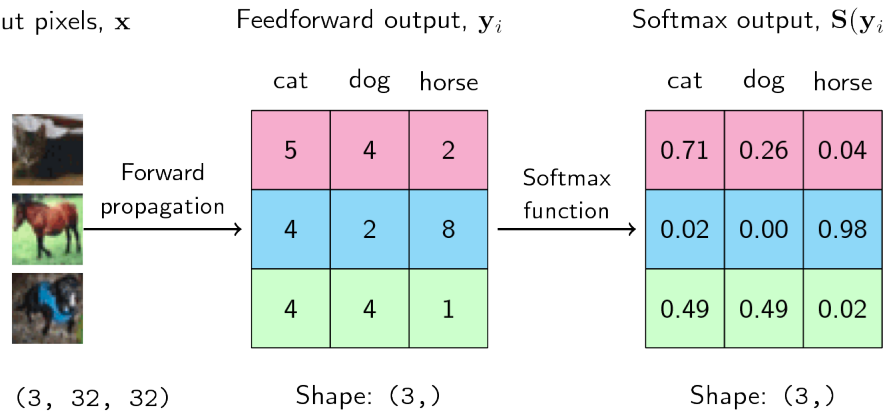
https://speakerdeck.com/sambaiga/foundation-of-deep-learning?slide=43

Clasificación multiclase



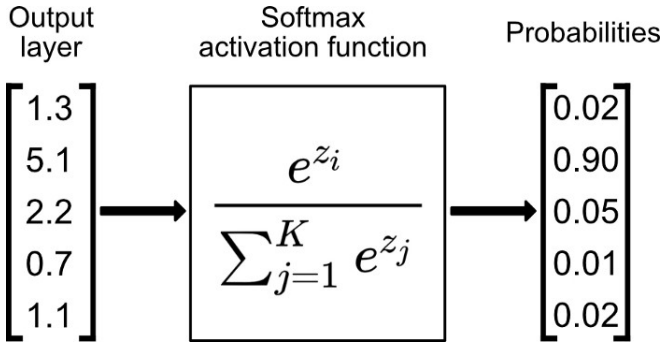
https://www.researchgate.net/publication/276240631_A_Taxonomy_of_Label_Ranking_Algorithms

Función de activación: *Softmax*



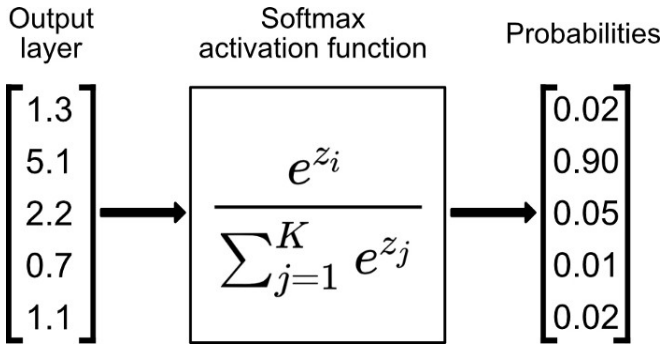
https://ogunlao.github.io/2020/04/26/you_dont_really_know_softmax.html

Función de activación: *Softmax*



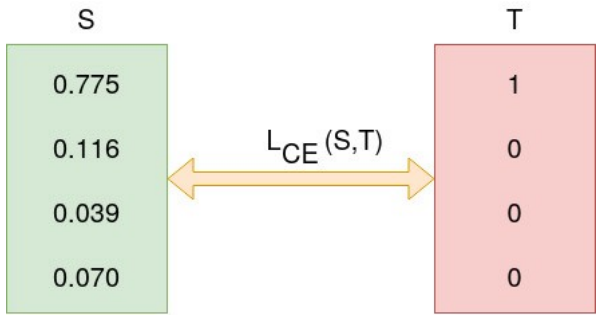
https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60

Función de pérdida: *Cross Entropy*



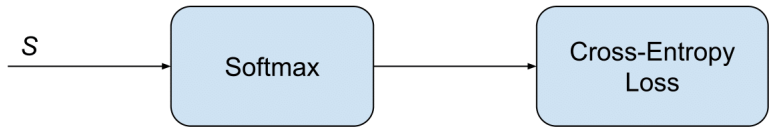
https://towardsdatascience.com/softmax-activation-function-explained-a7e1bc3ad60

Función de pérdida: *Cross Entropy*



https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e

Función de pérdida: *Cross Entropy*



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

https://gombru.github.io/2018/05/23/cross_entropy_loss/

Función de pérdida: *Binary Cross-Entropy*

https://gombru.github.io/2018/05/23/cross_entropy_loss/

UNIVERSIDAD ADOLFO IBÁÑEZ

¿Preguntas?