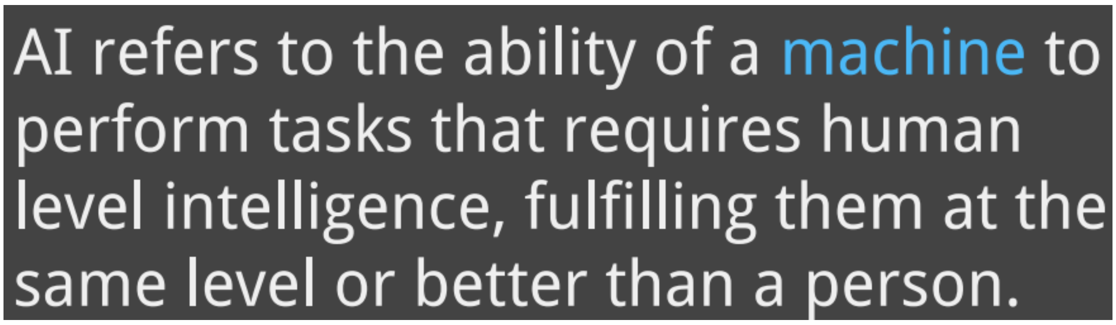
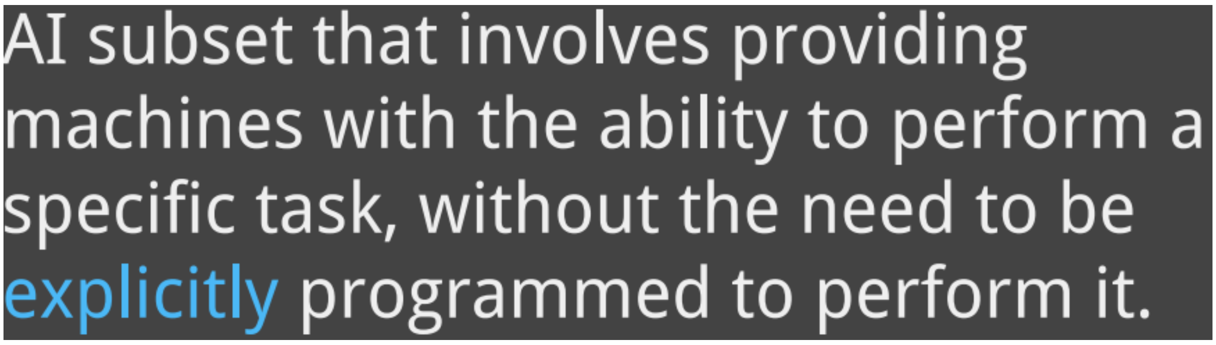
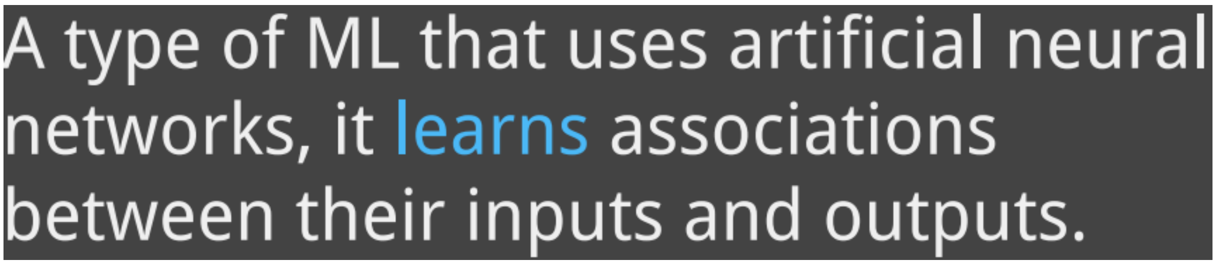


AI refers to the ability of a machine to perform tasks that requires human level intelligence, fulfilling them at the same level or better than a person.



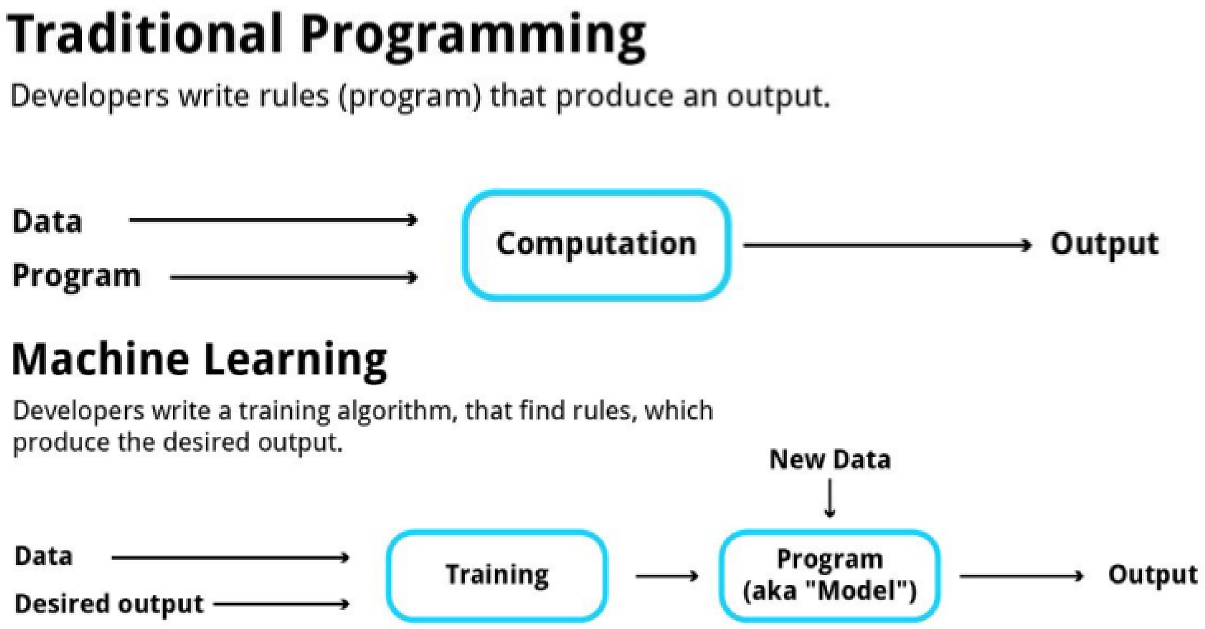
AI subset that involves providing machines with the ability to perform a specific task, without the need to be explicitly programmed to perform it.

Deep Learning



A type of ML that uses artificial neural networks, it learns associations between their inputs and outputs.

Traditional programming vs ML programming



Traditional Programming

Developers write rules (program) that produce an output.

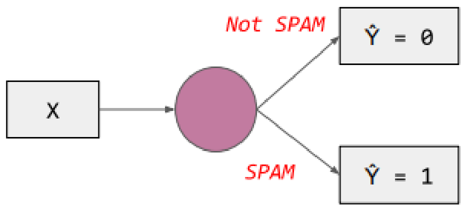
Machine Learning

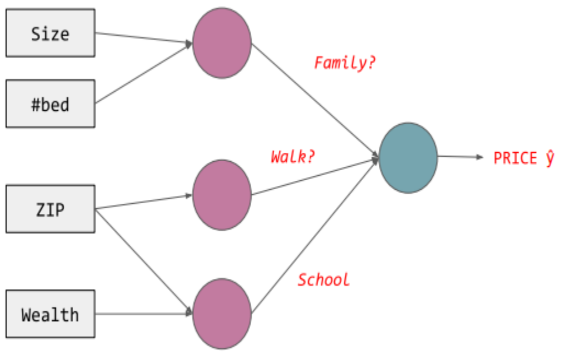
Developers write a training algorithm, that find rules, which produce the desired output.

Supervised Learning

CLASSIFICATION

REGRESSION

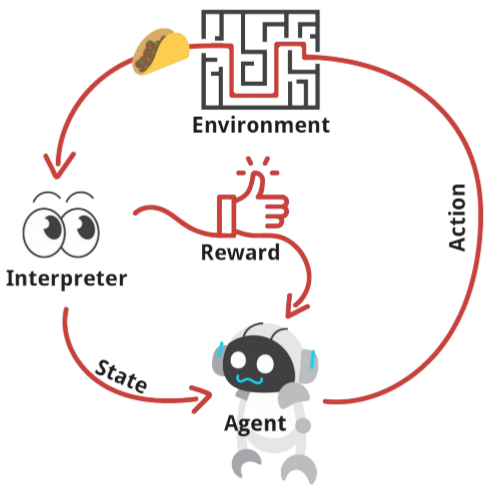




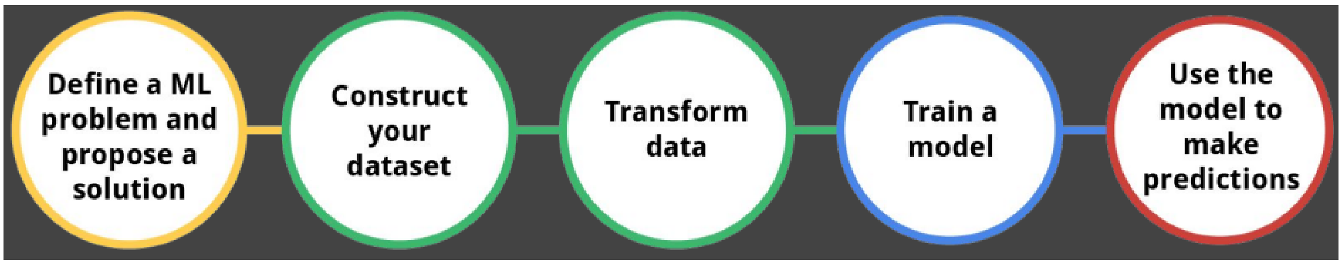
Unsupervised Learning

Cómo comer tacos.
Cabeza. 45° (aprox.)
Agarre experto (opcional)
90° (aprox.)
Plato /pizarra SIEMPRE bajo el taco (evita escurrimientos)
marcianosmx.com

Reinforcement Learning



Algoritmo para implementar ML


Define a ML problem and propose a solution
Construct your dataset
Transform data
Train a model
Use the model to make predictions

Features vs Label

A feature is one column of the data in your input set. For instance, if you're trying to predict the type of pet someone will choose, your input features might include age, home region, family income, etc. The label is the final choice, such as dog, fish, iguana, rock, etc.

Model:

A machine learning model is a file that has been trained to recognize certain types of patterns. You train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data.

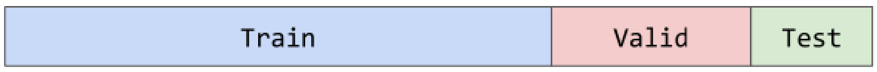
Loss vs Cost:

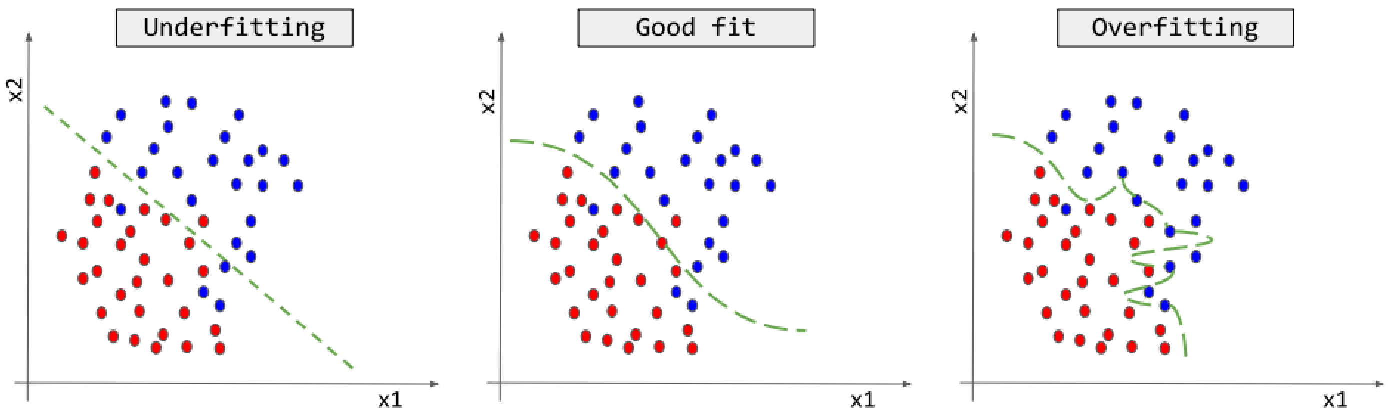
The terms cost and loss functions almost refer to the same meaning. But, loss function mainly applies for a single training set as compared to the cost function which deals with a penalty for a number of training sets or the complete batch.

Optimizer:

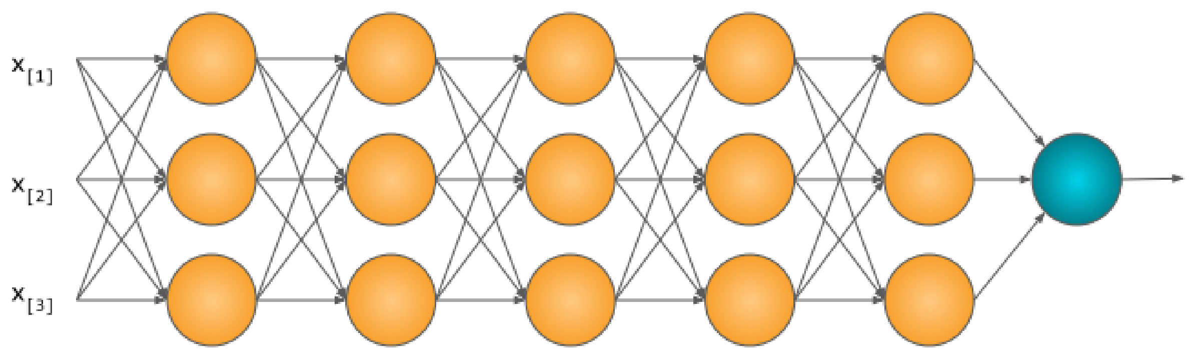
Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses. Optimizers help to get results faster.

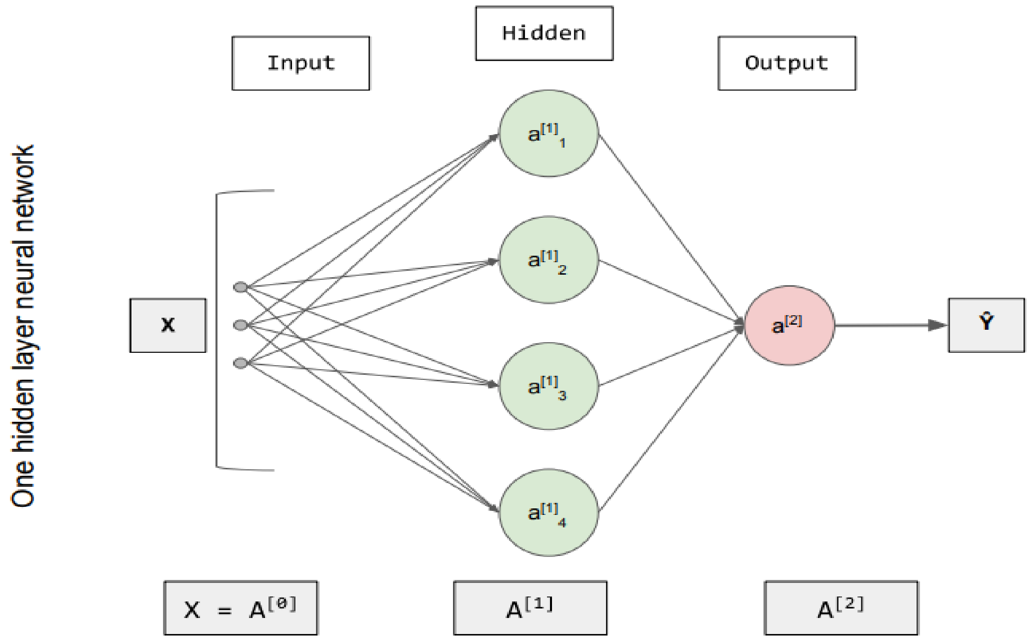
Dataset's

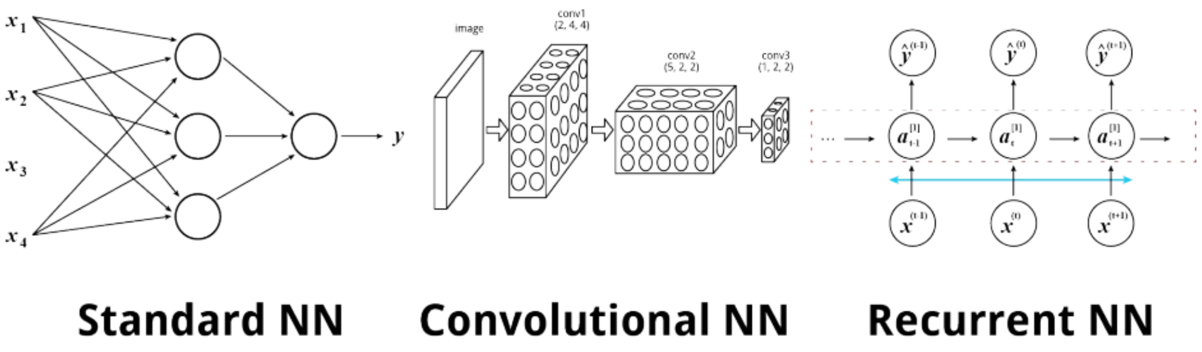


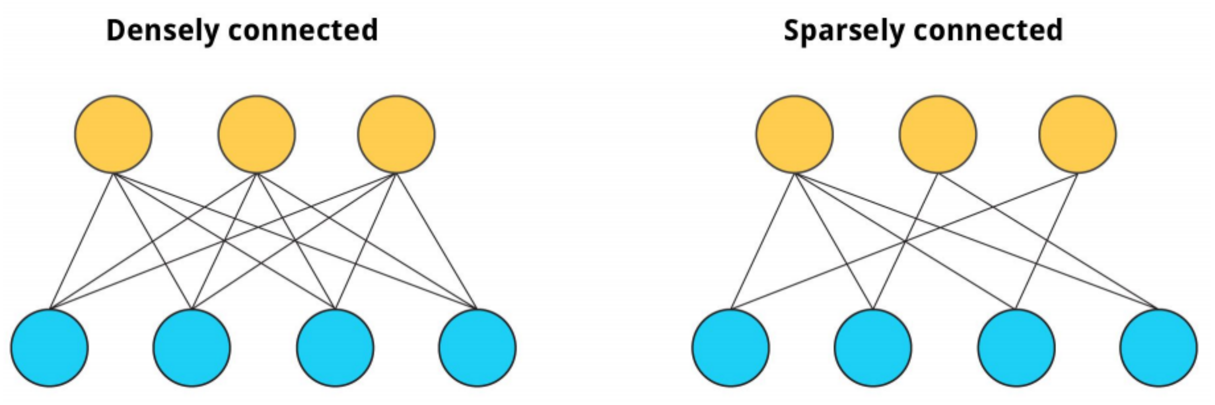


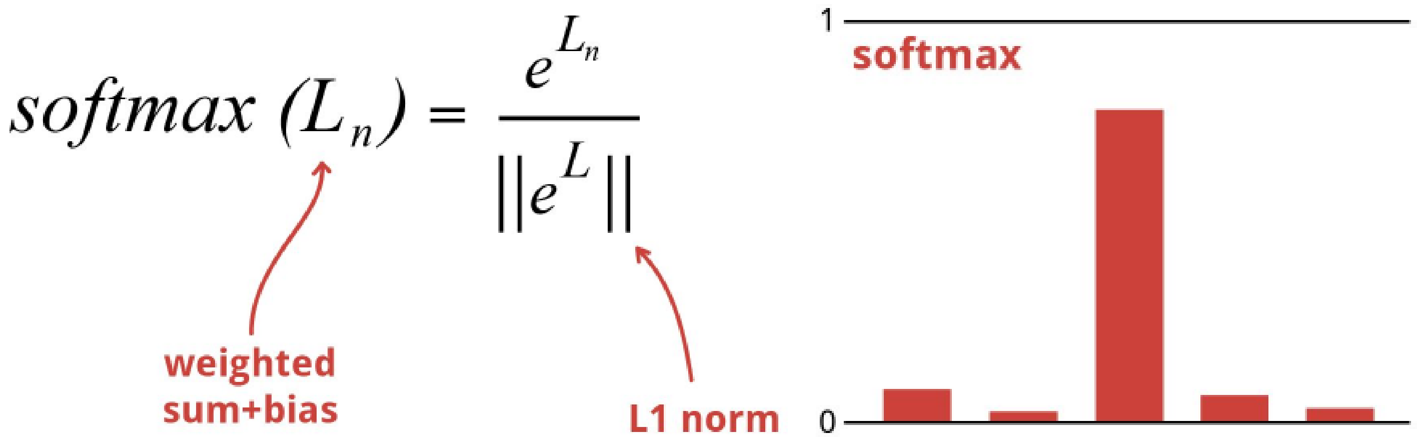
Fully connected neural network





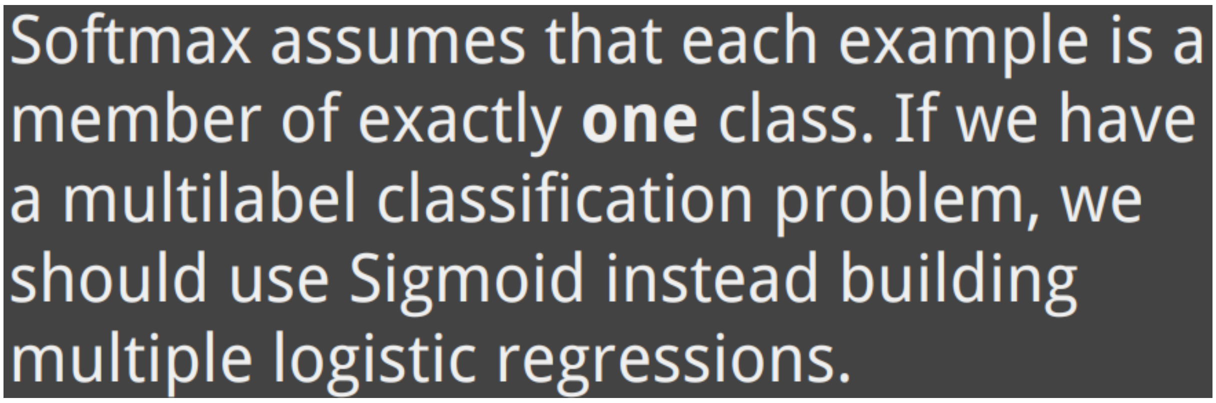






$$softmax\ (L_n) = \frac{e^{L_n}}{\|e^{L}\|}$$

Multilabel Clasification



Softmax assumes that each example is a member of exactly **one** class. If we have a multilabel classification problem, we should use Sigmoid instead building multiple logistic regressions.

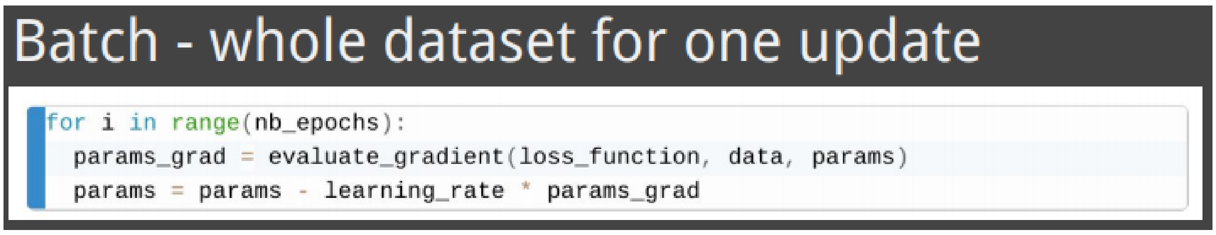
Backpropagation

La propagación hacia atrás de errores o retropropagación (del inglés *backpropagation*) es un método de cálculo del gradiente utilizado en algoritmos de aprendizaje supervisado utilizados para entrenar redes neuronales artificiales. El método emplea un ciclo de propagación – adaptación de dos fases. Una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas siguientes de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas.

Logistic Regression

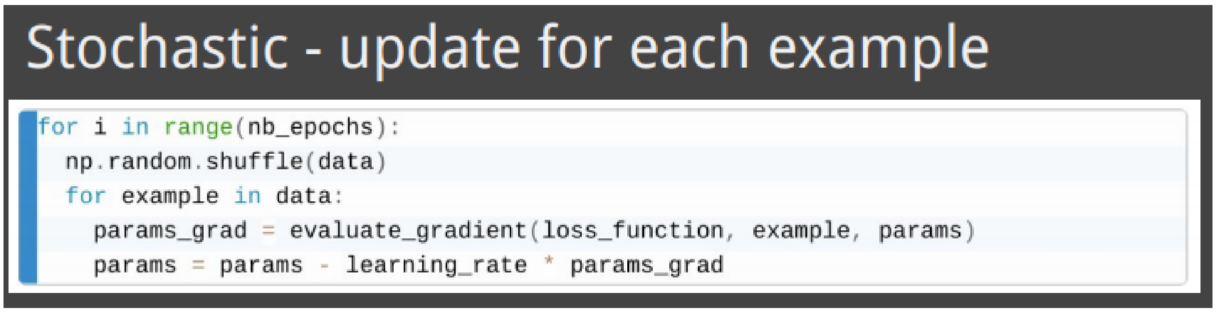
Es un tipo de análisis de regresión utilizado para predecir el resultado de una variable categórica (una variable que puede adoptar un número limitado de categorías) en función de las variables independientes o predictoras.

Gradient Descent Examples



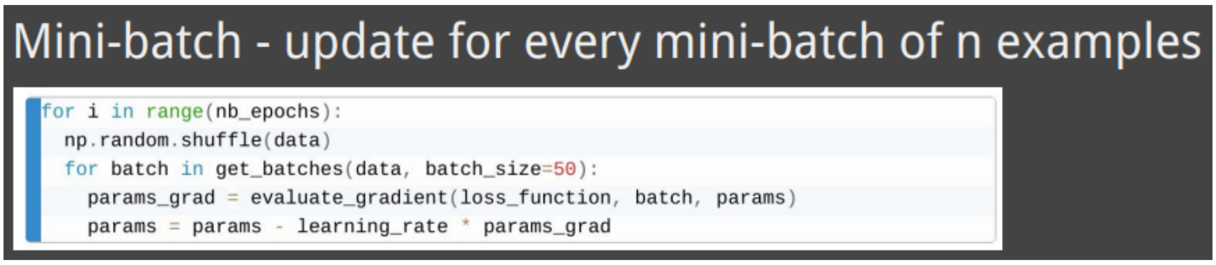
Batch - whole dataset for one update

```
for i in range(nb_epochs):
  params_grad = evaluate_gradient(loss_function, data, params)
  params = params - learning_rate * params_grad
```



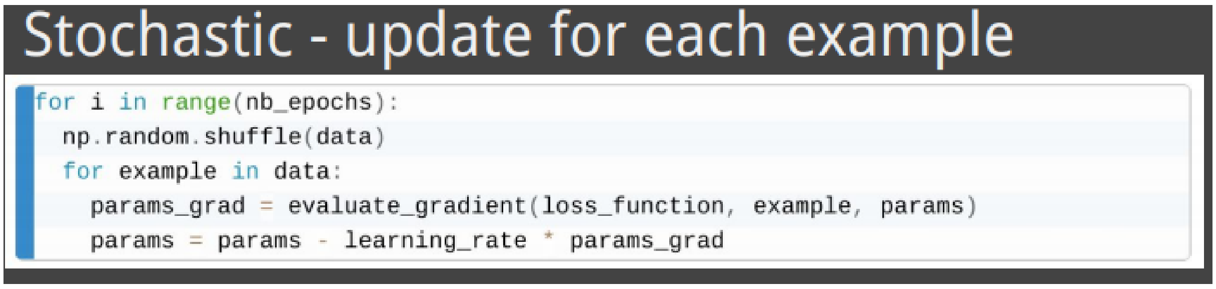
Stochastic - update for each example

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for example in data:
    params_grad = evaluate_gradient(loss_function, example, params)
    params = params - learning_rate * params_grad
```



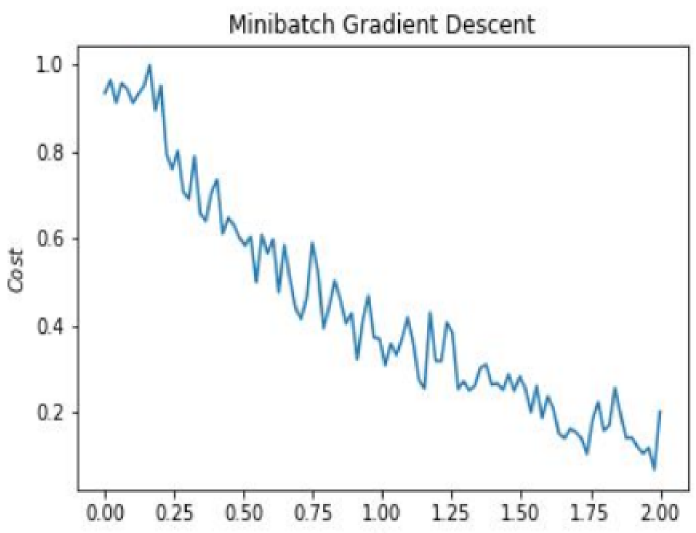
Mini-batch - update for every mini-batch of n examples

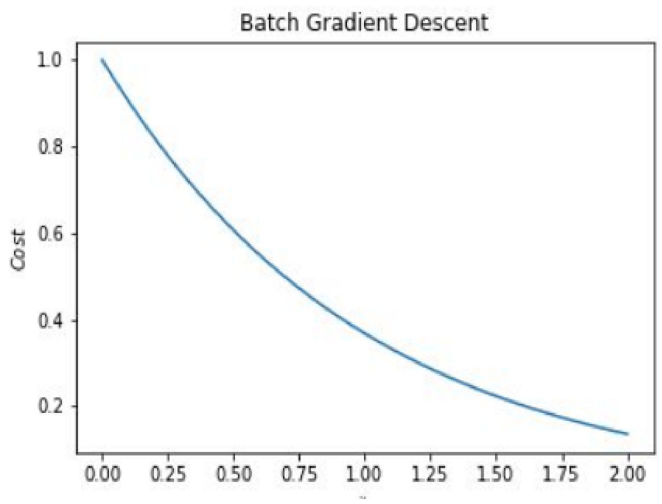
```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for batch in get_batches(data, batch_size=50):
    params_grad = evaluate_gradient(loss_function, batch, params)
    params = params - learning_rate * params_grad
```



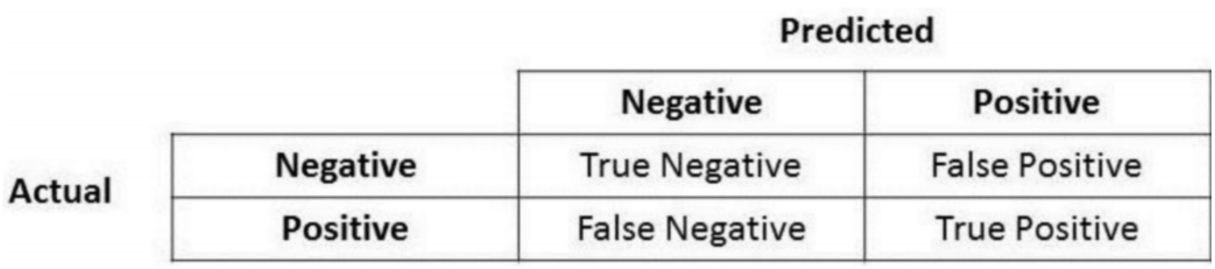
Stochastic - update for each example

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for example in data:
    params_grad = evaluate_gradient(loss_function, example, params)
    params = params - learning_rate * params_grad
```





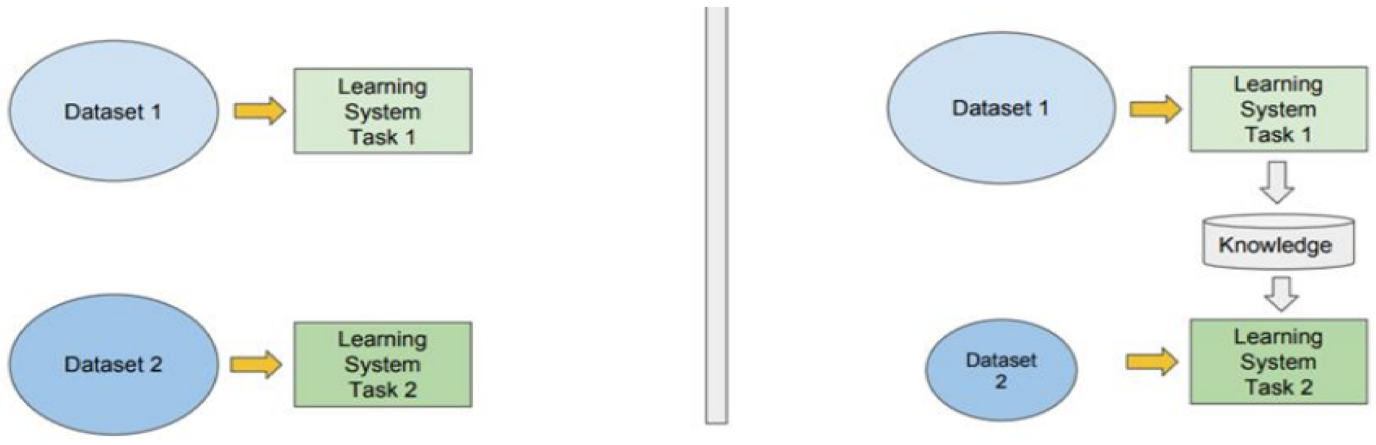
Accuracy Precision Recall



		Predicted	
		Negative	Positive
Actual	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

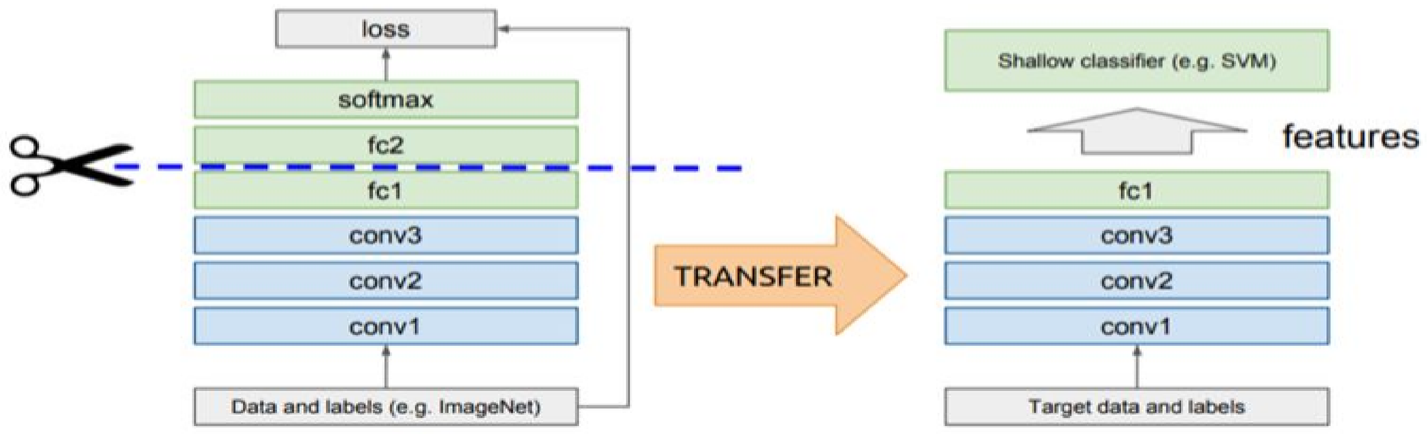
Transfer Learning

Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.[1] For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks. This area of research bears some relation to the long history of psychological literature on transfer of learning, although formal ties between the two fields are limited. From the practical standpoint, reusing or transferring information from previously learned tasks for the learning of new tasks has the potential to significantly improve the sample efficiency of a reinforcement learning agent



Feature Extraction

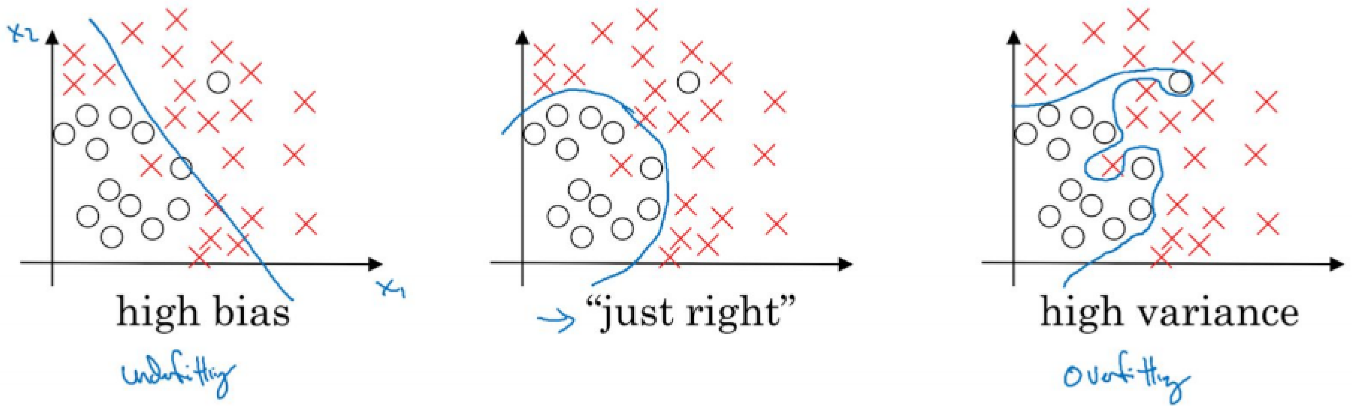
In machine learning, pattern recognition, and image processing, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction.

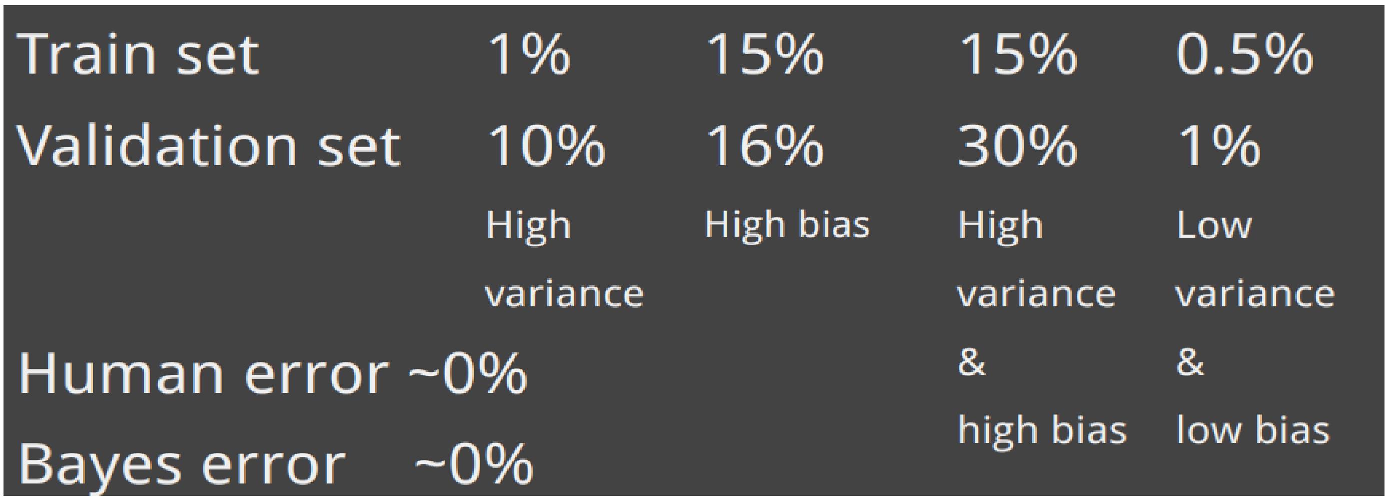


Fine Tuning

Means taking weights of a trained neural network and use it as initialization for a new model being trained on data from the same domain (often e.g. images). It is used to: speed up the training, overcome small dataset size

BIAS AND VARIANCE





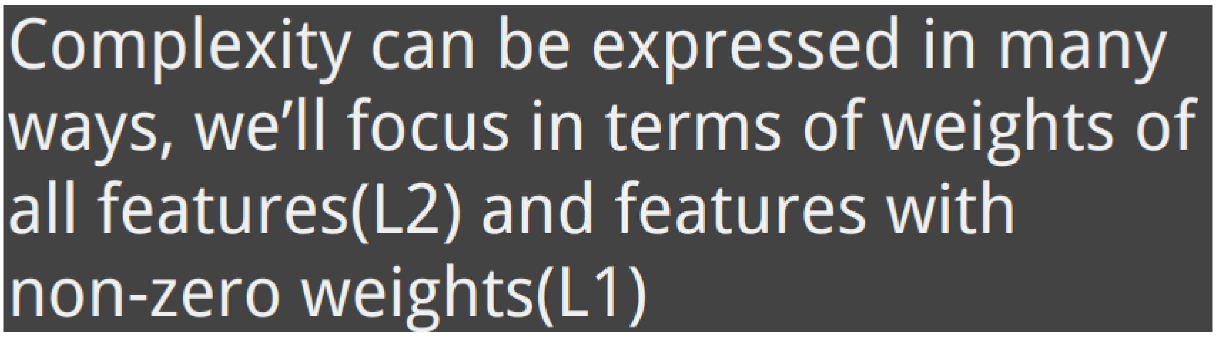
Train set	1%	15%	15%	0.5%
Validation set	10%	16%	30%	1%
	High variance	High bias	High variance & high bias	Low variance & low bias

Human error ~0%

Bayes error ~0%

Regularization

La regularización, en matemáticas y estadística y particularmente en los campos de aprendizaje automático y problemas inversos, se refiere a un proceso de introducir información adicional para solucionar un problema mal definido o para impedir el sobreajuste.



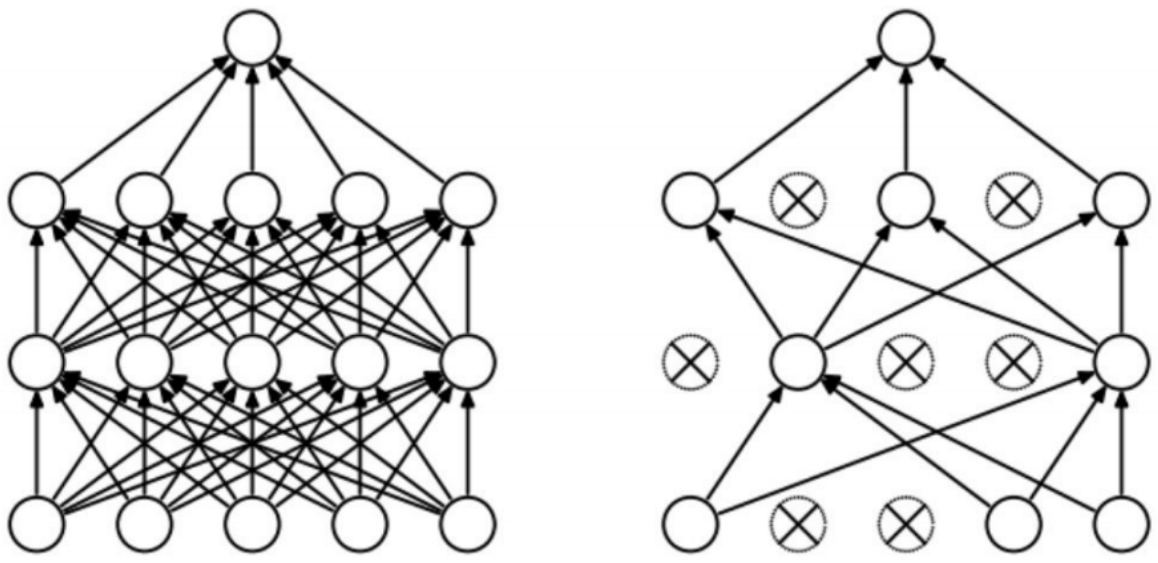
Complexity can be expressed in many ways, we'll focus in terms of weights of all features(L2) and features with non-zero weights(L1)

L1

In L1 we shrink parameters to zero, thus removing features. When regulating very few features have non-zero weights.

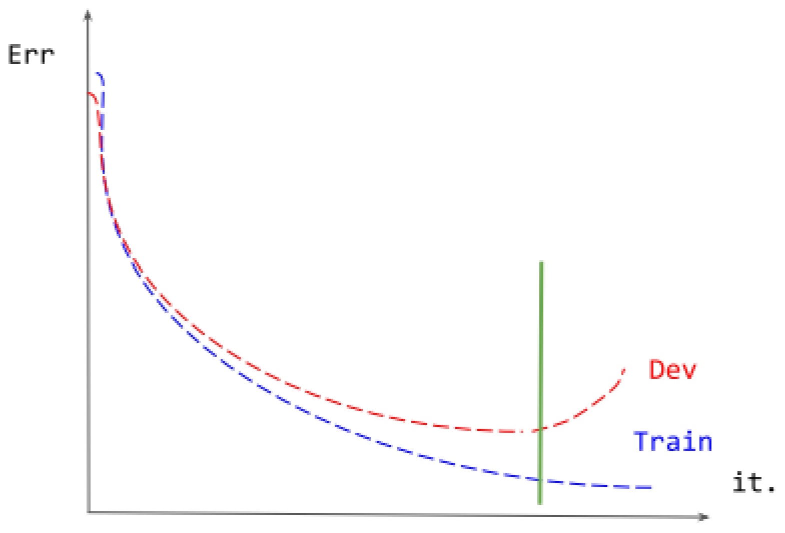
DropOUT

Dilución (también conocido como Dropout) es una técnica de regularización para reducir el sobreajuste en redes neuronales artificiales. Es una forma eficiente de realizar promedios de modelos con redes neuronales. El término dropout significa "abandonar" u omitir aleatoriamente neuronas(tanto ocultas como visibles) durante el proceso de entrenamiento de una red neuronal. [1] Tanto la reducción de los pesos como omitir unidades obtienen el mismo tipo de regularización.

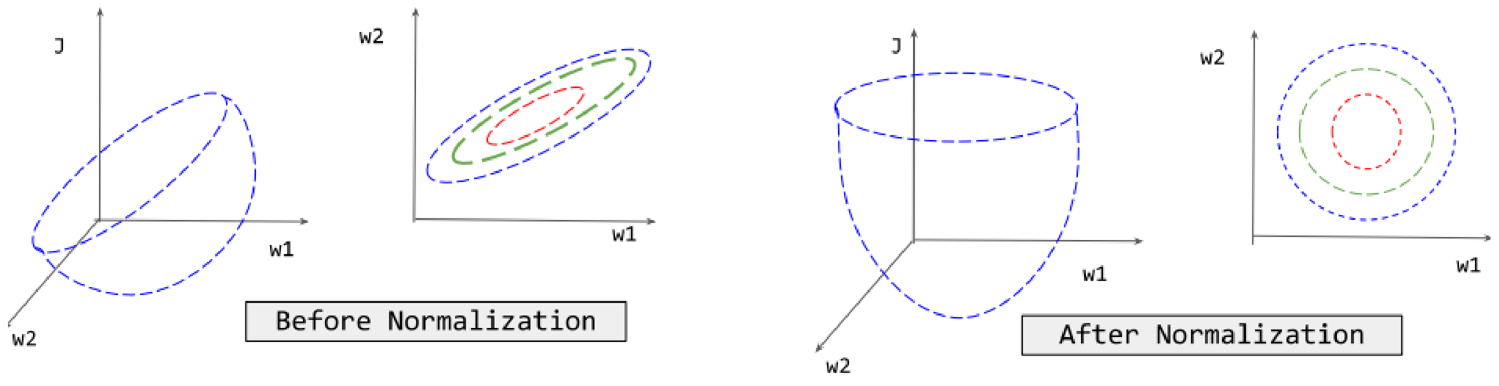

(a) Standard Neural Net (b) After applying dropout.

Early Stopping

In machine learning, early stopping is a form of regularization used to avoid overfitting when training a learner with an iterative method, such as gradient descent. Such methods update the learner so as to make it better fit the training data with each iteration. Up to a point, this improves the learner's performance on data outside of the training set. Past that point, however, improving the learner's fit to the training data comes at the expense of increased generalization error. Early stopping rules provide guidance as to how many iterations can be run before the learner begins to over-fit. Early stopping rules have been employed in many different machine learning methods, with varying amounts of theoretical foundation.



Normalization



Data augmentation

Data augmentation in data analysis are techniques used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model.[1] It is closely related to oversampling in data analysis.

Vanish gradients

En aprendizaje de máquinas, el problema de desvanecimiento de gradiente es una dificultad encontrada para entrenar redes neuronales artificiales mediante métodos de aprendizaje basados en descenso estocástico de gradientes y de retropropagación. En tales métodos, cada uno de los pesos de la red neuronal recibe una actualización proporcional a la derivada parcial de la función de error con respecto al peso actual en cada iteración de entrenamiento.

Exploiding Gradients

An error gradient is the direction and magnitude calculated during the training of a neural network that is used to update the network weights in the right direction and by the right amount.

In deep networks or recurrent neural networks, error gradients can accumulate during an update and result in very large gradients. These in turn result in large updates to the network weights, and in turn, an unstable network. At an extreme, the values of weights can become so large as to overflow and result in NaN values.

The explosion occurs through exponential growth by repeatedly multiplying gradients through the network layers that have values larger than 1.0

Dead relu units

A "dead" ReLU always outputs the same value (zero as it happens, but that is not important) for any input. Probably this is arrived at by learning a large negative bias term for its weights.
In turn, that means that it takes no role in discriminating between inputs. For classification, you could visualise this as a decision plane *outside* of all possible input data.
Once a ReLU ends up in this state, it is unlikely to recover, because the function gradient at 0 is also 0, so gradient descent learning will not alter the weights. "Leaky" ReLUs with a small positive gradient for negative inputs ($y=0.01x$ when $x < 0$ say) are one attempt to address this issue and give a chance to recover.
The sigmoid and tanh neurons can suffer from similar problems as their values saturate, but there is always at least a small gradient allowing them to recover in the long term.