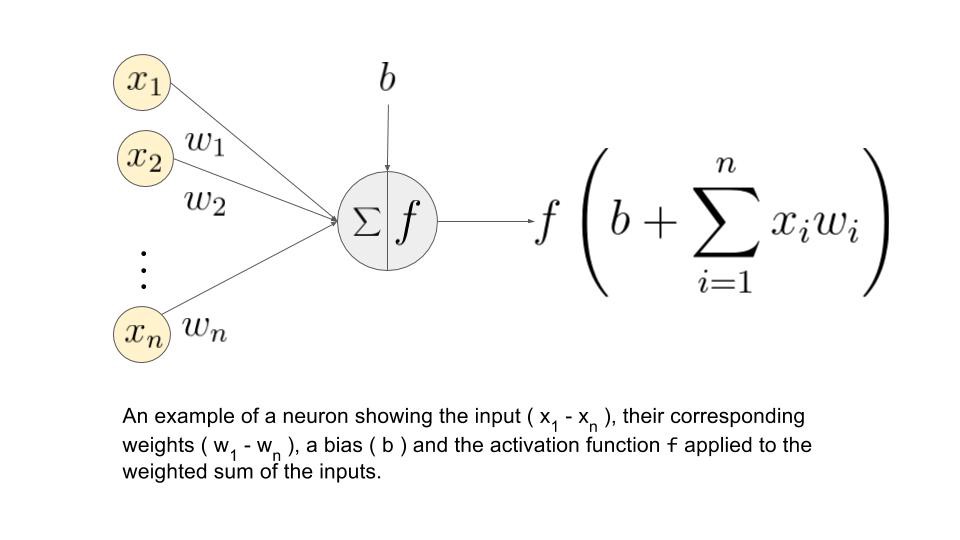
***Artificial Neural Network***

1. Jelaskan bagaimana forward propagation pada ANN!

*Forward Propagation* adalah proses mempropagasikan input data yang ada dari *input layer* sampai menuju ke *output layer*. Proses ini akan memerlukan inisialisasi *weight* atau bobot dan *bias* untuk setiap neuron yang ada pada ANN. *Weight* mengindikasikan seberapa penting suatu fitur tertentu terhadap penentuan *output*, sedangkan *bias* merupakan cara untuk membantu model dalam melakukan *fitting* terhadap dataset, mirip seperti *intercept* pada regresi biasa.

Untuk mendapatkan suatu *output*, maka data input harus diberikan hanya pada arah depan (menuju ke *output layer*). Propagasi yang demikian dimaksudkan untuk menghindari terjadinya siklus pada graf neuron yang ada. Konfigurasi model yang dimaksud disebut dengan *feed-forward network*.

Secara umum, pada proses ini, model hanya melakukan dua buah tahap pemrosesan input saja. Tahap pertama merupakan tahap *pre-activation*. Pada tahap ini, model akan melakukan pemrosesan input yang didapat dari neuron pada *layer* sebelumnya. Model akan melakukan transformasi linear untuk mengkalkulasi *weighted sum* atau agregasi bobot terhadap masukan yang diberikan. Ilustrasi kalkulasi *weighted input sum* ini untuk sebuah *layer* dapat terlihat pada gambar berikut.



Agregasi bobot yang telah didapat dari hasil tersebut kemudian akan ditentukan untuk diteruskan atau tidak informasinya ke neuron pada *layer* selanjutnya. Penentuan ini dilakukan pada tahap kedua, yaitu *activation*. Pada tahap ini, nilai yang didapat pada tahap sebelumnya akan dilewatkan ke dalam *activation function*. *Activation Function* merupakan fungsi matematis yang digunakan untuk menambah non-linearitas kepada *network* yang dibangun.

Kedua tahap ini dilakukan untuk semua lapisan neuron hingga mencapai *output layer*. Hasil yang didapat pada *output layer* dapat dikatakan sebagai sebuah prediksi oleh model pada saat itu dan dengan demikian dilabeli sebagai *ŷ*. *ŷ* ini bisa digunakan untuk dibandingkan dengan nilai yang sebenarnya pada data. Perbedaan keduanya dapat dihitung dengan menggunakan *cost* atau *loss function* sebagai bentuk evaluasi model.

1. Jelaskan bagaimana back propagation pada ANN!

*Back Propagation* adalah proses dalam *Neural Network* yang berlawanan dengan *Forward Propagation*. *Back Propagation* merupakan propagasi informasi mengenai *error* dari *layer terakhir* (*output layer*) sampai *layer* pertama (*input layer*). Informasi tersebut diperoleh dari kalkulasi *error function* terhadap *ŷ* yang telah didapat tadi. Tujuan dari propagasi ini ialah untuk memperbaiki *weight* dan *bias* yang telah diinisialisasi sebelumnya.

Teknik yang biasanya digunakan dalam *back propagation* adalah *Gradient Descent*. *Back propagation* akan mencari gradien dari *error* hasil *Forward Propagation* terhadap *weight* dan *bias* yang dimilikinya. Gradien ini nantinya akan digunakan untuk memperbaiki kedua hiperparameter bersamaan dengan parameter lain yang disebut dengan *learning rate*. Tujuan dari *gradient descent* sendiri adalah untuk mencari titik global minimum dari *error function* yang dimiliki. Dengan demikian, model yang dibangun memiliki kemampuan untuk melakukan *fitting* dengan baik terhadap data. Kalkulasi yang bisa dilakukan untuk mendapatkan gradien ini secara sederhana adalah sebagai berikut.

Shape

Description automatically generated with medium confidence





Shape

Description automatically generated with medium confidence

Shape

Description automatically generated with medium confidence

Shape

Description automatically generated with medium confidence

atau secara lebih umum:

Text, letter

Description automatically generated

Perubahan *weight* dan *bias* dapat dilakukan dengan mengurangi *weights* dan *bias* pada saat itu dengan hasil perkalian *learning rate* terhadap gradien hiperparameter yang berkaitan. Kedua (atau bahkan ketiga jika memperhitungkan perubahan *weight* dan *bias*) dilakukan secara iteratif sebanyak jumlah *epochs* yang ditentukan pada pemakaian algoritma.

Text

Description automatically generated with low confidence

*Gradient Descent* merupakan salah satu algoritma optimisasi yang bisa digunakan. Dalam implementasi tugas ini, digunakan teknik lain yang disebut dengan *Stochastic Gradient Descent*. Cara kerja teknik ini mirip dengan algoritma *Gradient Descent* yang juga biasa disebut dengan *Batch Gradient Descent*. Bedanya, pada *Batch Gradient Descent* gradien dihitung untuk semua data yang ada, sedangkan pada SGD gradien dihitung dengan menggunakan metode stokastik dan memilih data dari dataset dalam jumlah tertentu secara acak.

1. Apa itu optimizer, loss function, dan activation function?

*Optimizer* adalah algoritma atau fungsi yang digunakan untuk meminimalkan *error function. Optimizer* bekerja dengan memperbarui hiperparameter model, yaitu *weights* dan *biases*, sebagai respon terhadap *error function* yang didapat pada *forward propagation*. Secara umum, optimisasi dilakukan dengan mencari gradien dari *error function* terhadap hiperparameter tersebut. Teknik yang paling umum dan sederhana untuk digunakan adalah *Gradient Descent* yang juga menjadi cikal bakal teknik lainnya.

*Loss Function* merupakan salah satu jenis *error function* yang sering digunakan secara *interchangeably* dengan *Cost Function*. Secara sederhana, *loss function* merupakan sebuah cara untuk mengevaluasi apakah kondisi model pada saat tertentu sudah cukup baik dalam mempelajari data masukan yang diberikan. Dengan demikian, maka model diinginkan mempunyai *loss function* yang nilainya semakin rendah sehingga mengindikasikan bahwa model berhasil melakukan *fitting* terhadap data yang ada. *Loss function* berbeda dengan *Cost Function* yang bersifat lebih umum dan biasanya digunakan untuk menyatakan akumulasi *loss function* pada suatu *batch* data tertentu.

1. Sebutkan beberapa optimizer, loss function, dan activation function lainnya selain yang tercantum pada spesifikasi!

Beberapa *optimizer* lain yang ada pada saat ini adalah *Gradient Descent* dan *Mini-Batch Gradient Descent* yang menggunakan *batches of data* dalam perhitungan gradien dan merupakan kombinasi SGD dan *Batch Gradient Descent* biasa. Selain itu, SGD juga memiliki versi yang menggunakan momentum untuk mensimulasikan adanya inersia pada pergerakan gradiennya.

Beberapa *optimizer* lain yang sering digunakan pada saat ini adalah *Nesterove Accelerated Gradient* (NAG), *Adaptive Gradient Descent* (AdaGrad), *Root Mean Square Propagation* (RMS-Prop) dan AdaDelta yang merupakan perkembangan dari AdaGrad, serta Adam (*Adaptive Moment Estimation*) yang mengombinasikan RMS-Prop dengan *Momentum Based Gradient* dan dapat dikatakan merupakan *optimizer* paling popular untuk *Deep Learning* pada saat ini.

Beberapa contoh dari *loss function* adalah *Mean Absolute Error* (MAE atau L1 *Loss*), *Mean Squared Error* (MSE atau L2 *Loss*, sering juga disebut dengan *Quadratic Loss*), *Root Mean Square Error* (RMSE), *Mean Bias Error*, dan *Mean Squared Logarithmic Error* yang digunakan untuk permasalahan regresi. Adapun permasalahan klasifikasi biner biasanya menggunakan *Binary Cross-Entropy Loss*, *Hinge Loss* (atau *Multiclass* SVM *Loss*), atau *Squared Hinge Loss*, sedangkan permasalahan klasifikasi multikelas dapat menggunakan *Cross-Entropy Loss* (*Negative Log Likelihood*), *Sparse-Multiclass Entropy Loss*, dan *Kullback Leibler Divergence Loss*.

1. Pada deep learning, terdapat beberapa arsitektur seperti CNN atau RNN. Coba tuliskan hal yang kamu ketahui tentang beberapa arsitektur lain selain ANN!

Referensi Tambahan:

[*Backpropagation*](https://brilliant.org/wiki/backpropagation/)*.*

[*Common Loss functions in machine learning.*](https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23)

[*Cost, Activation, Loss Function|| Neural Network|| Deep Learning. What are these?*](https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these-91167825a4de)

[*Forward Propagation and Errors in a Neural Network*](https://www.analyticsvidhya.com/blog/2021/03/forward-propagation-and-errors-in-a-neural-netwrok/)*.*

[*Forward Propagation in Neural Networks — Simplified Math and Code Version.*](https://towardsdatascience.com/forward-propagation-in-neural-networks-simplified-math-and-code-version-bbcfef6f9250)

[*How Does Back-Propagation in Artificial Neural Networks Work?*](https://towardsdatascience.com/how-does-back-propagation-in-artificial-neural-networks-work-c7cad873ea7)

*[Introduction to loss functions.](https://algorithmia.com/blog/introduction-to-loss-functions" \l "whats-a-loss-function)*

[*Introduction to optimizers.*](https://algorithmia.com/blog/introduction-to-optimizers)

[*Keras optimizers*](https://www.kaggle.com/residentmario/keras-optimizers)*.*

[*Optimizers in Deep Learning*](https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0)*.*

[*Parameter optimization in neural networks.*](https://www.deeplearning.ai/ai-notes/optimization/)

[*Types of Optimizers in Deep Learning Every AI Engineer Should Know.*](https://www.upgrad.com/blog/types-of-optimizers-in-deep-learning/)