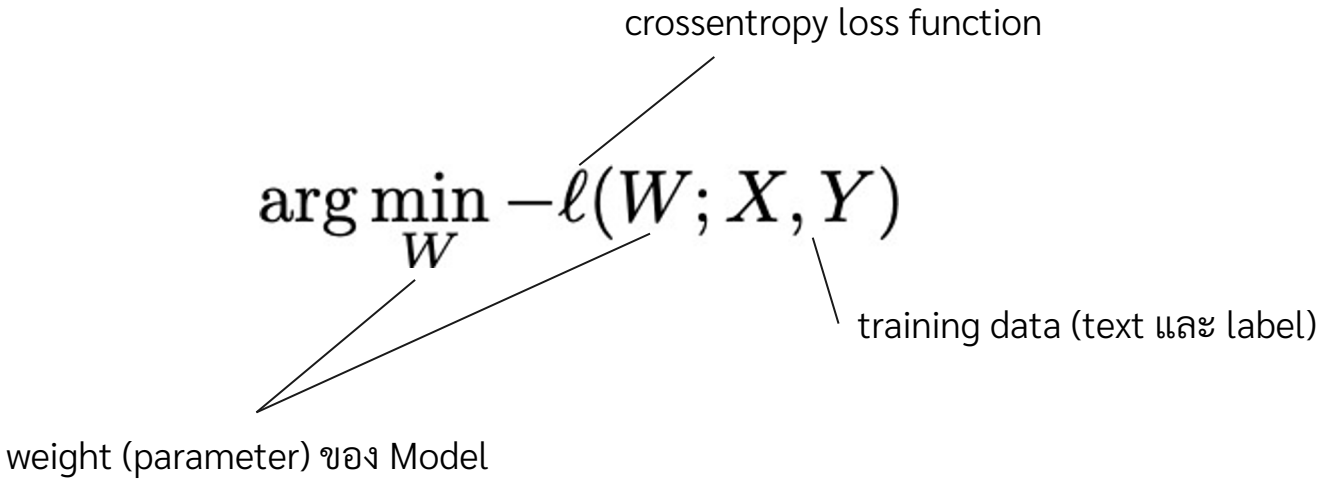
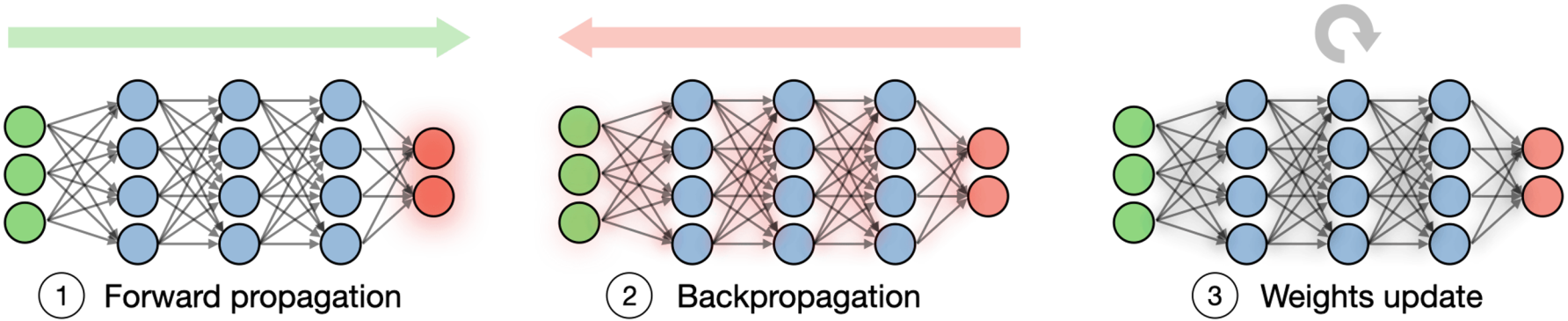
เทรน Deep Learning Model ด้วย Adaptive Learning Rate

หา Weight ที่ทำให้ Loss ต่ำที่สุด = Optimization



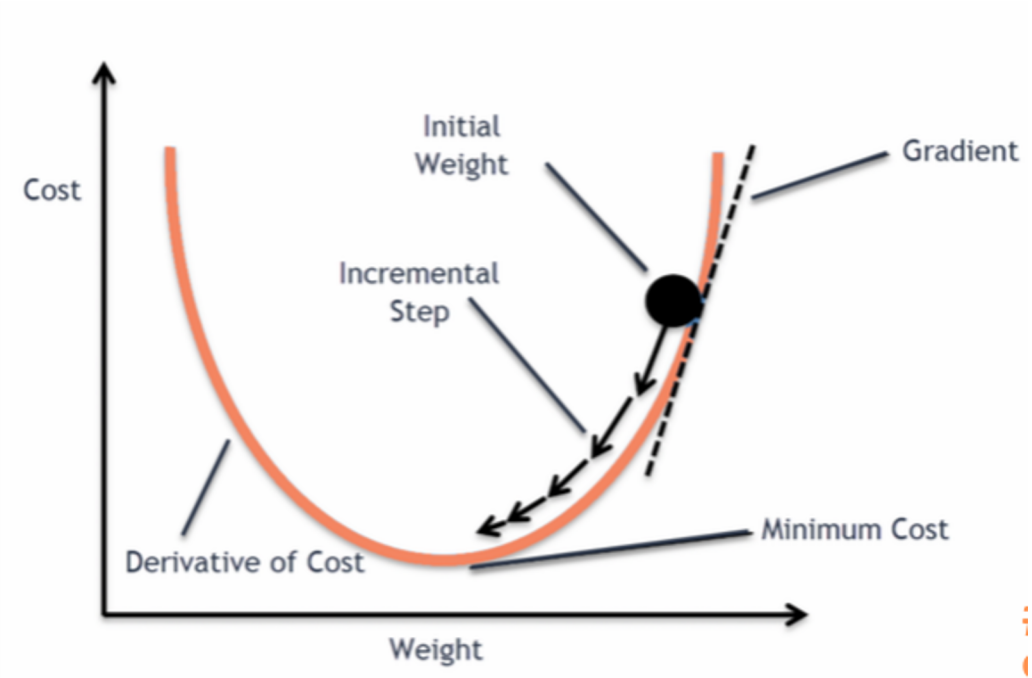
Training Process



https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-deep-learning-tips-and-tricks

Gradient Descent

การใช้ Error (ซึ่งมาจากการ diff objective/loss function) นำมาปรับ parameter เพื่อให้ loss function ต่ำลง



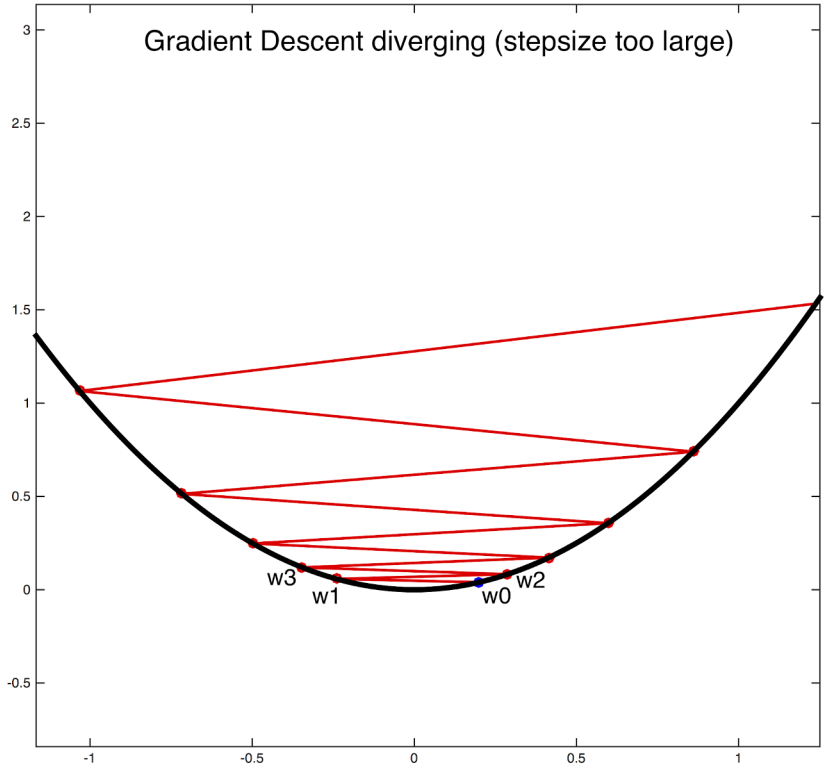
https://blog.clairvoyantsoft.com/the-ascent-of-gradient-descent-23356390836f

Problems

- Learning rate = step size = ควรจะก้าวสั้น ก้าวยาว
  - ก้าวช้า Parameter ขยับช้าเกินไป
  - ก้าวเร็ว Parameter ขยับเร็วเกินไป มัวแต่เลยไปเลยมา

ก้าวใหญ่ไป

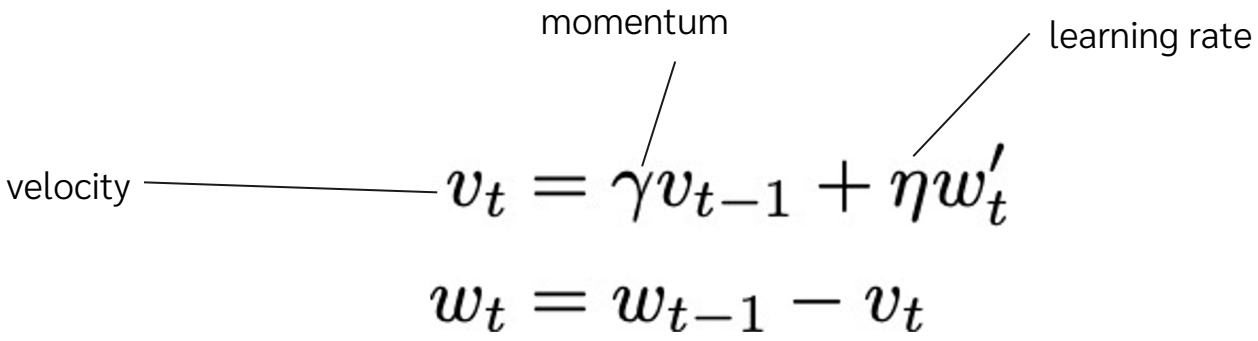
กระโดดไป กระโดดมา



Adaptive Learning Rate Optimizer

- Momentum
- RMSProp
- AdaGrad

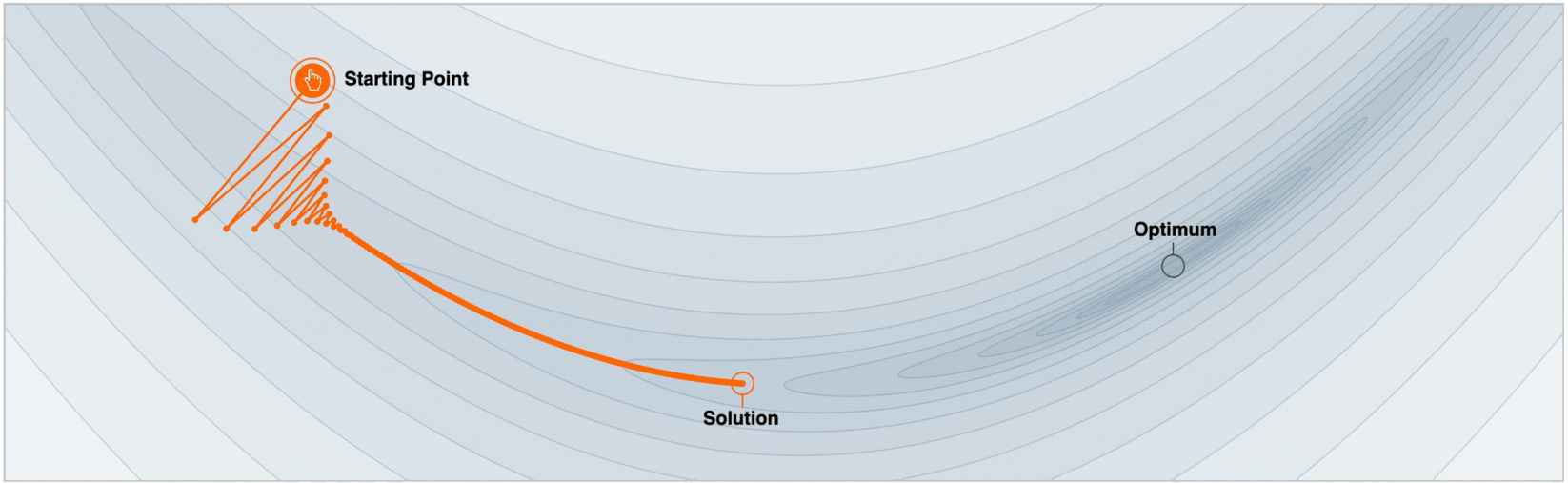
Momentum


momentum
learning rate
velocity
$v_t = \gamma v_{t-1} + \eta w'_t$
$w_t = w_{t-1} - v_t$

Momentum

$$v_t = \gamma v_{t-1} + \eta w'_t$$

$$w_t = w_{t-1} - v_t$$



Step-size α = 0.0030

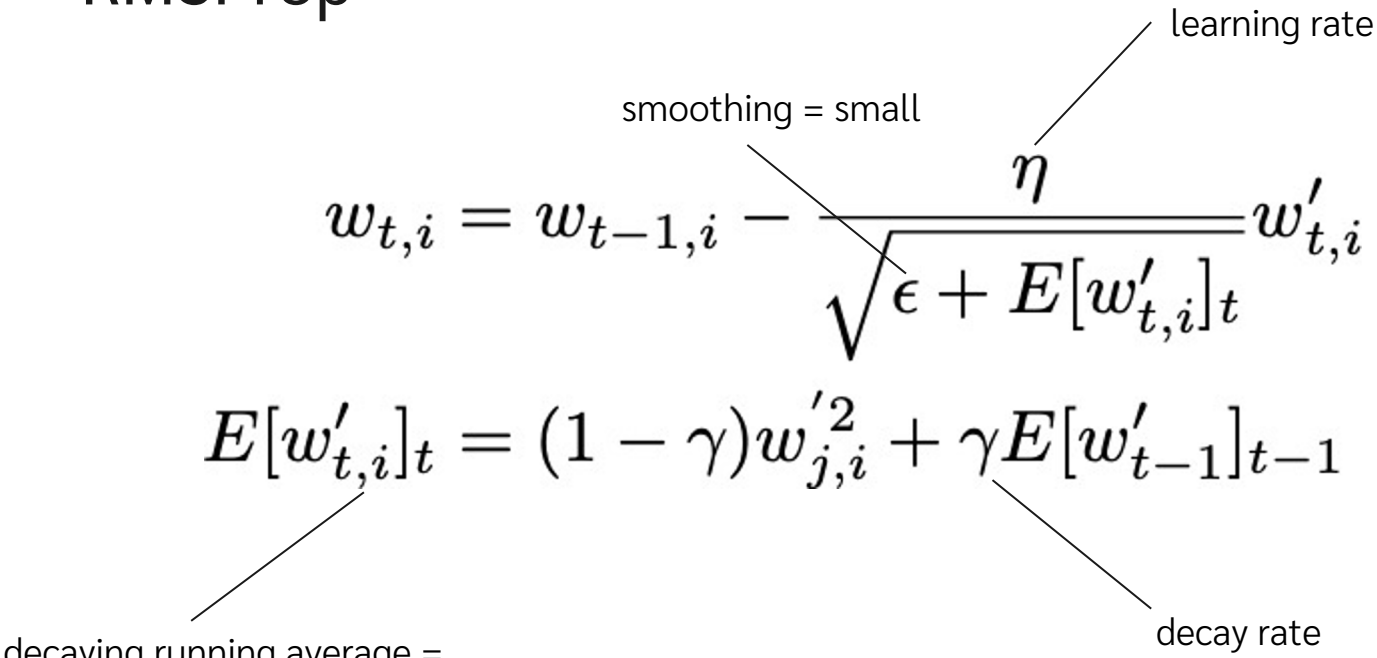
0 0.003 0.006

Momentum β = 0.0

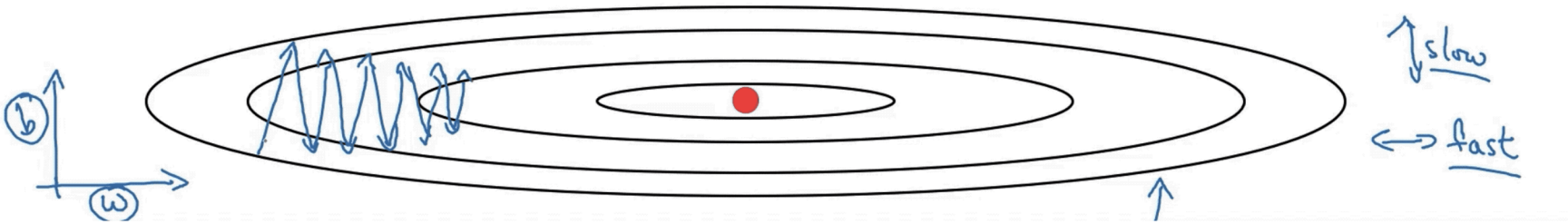
0.00 0.500 0.990

We often think of Momentum as a means of dampening oscillations and speeding up the iterations, leading to faster convergence. But it has other interesting behavior. It allows a larger range of step-sizes to be used, and creates its own oscillations. What is going on?

RMSProp


learning rate
smoothing = small
$w_{t,i} = w_{t-1,i} - \frac{\eta}{\sqrt{\epsilon + E[w'_{t,i}]_t}} w'_{t,i}$
$E[w'_{t,i}]_t = (1-\gamma) w'^{2}_{j,i} + \gamma E[w'_{t-1}]_{t-1}$
decaying running average =
ค่าเฉลี่ยที่ให้ค่าน้ำหนักของของใหม่มากกว่า
decay rate

RMSprop



Momentum (blue) and RMSprop (green)

AdaGrad

learning rate

$$w_{t,i} = w_{t-1,i} - \frac{\eta}{\sqrt{\epsilon + \sum_{j=1}^{t-1} w'^{2}_{j,i}}} w'_{t,i}$$

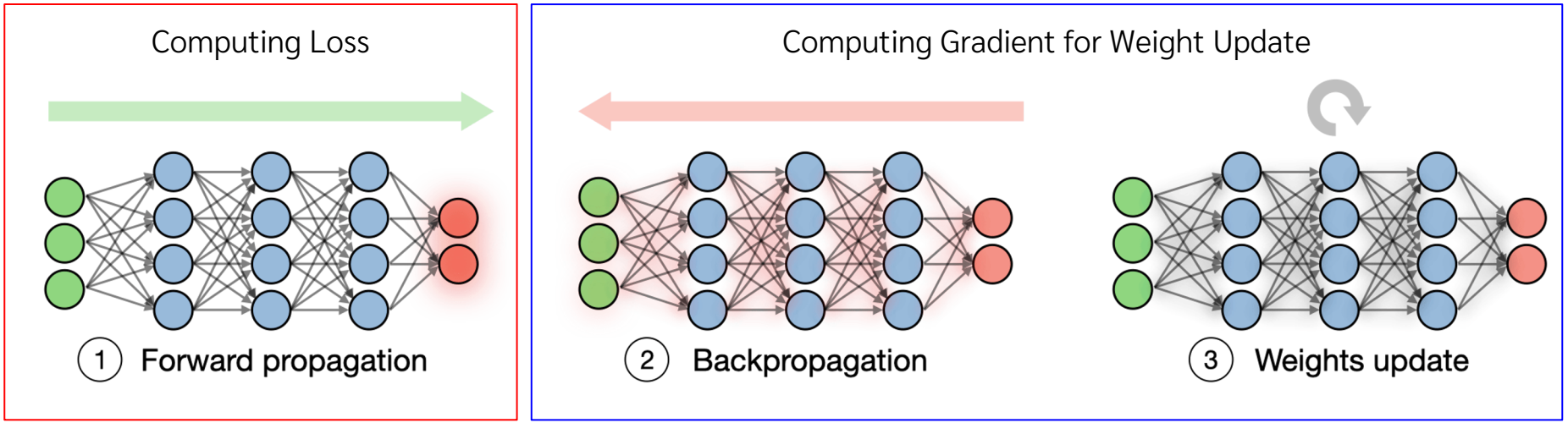
ค่านี้จะใหญ่ขึ้นเรื่อยๆ ทำให้ก้าวช้าลงเรื่อย ๆ

Optimizers

- ไม่มีข้อตกลงแน่นอนว่าอันไหนดีกว่าอันไหน ในสถานการณ์ไหน
- Optimizer ที่เป็นที่นิยมแต่ไม่ได้พูดถึง
  - Adam
  - AdaDelta

Minibatching and Batch Normalization

Training Process



https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-deep-learning-tips-and-tricks

Text	Label

Stochastic Gradient Descent

Text	Label

Batch Gradient Descent

Mini-batching

เพราะคำนวณ gradient ใช้เวลานานเกินไป ถ้าคำนวณจากทุกแถว

mini-batch size = 3
number of rows = 15
number of mini-batches = 5

1 epoch = 5 iterations

Text	Label

Batch Normalization ช่วยทำให้เทรน โดยใช้ epoch เท่าเดิม แต่ได้ผลที่ดีขึ้น

Batch Normalization คืออะไร

- ตอนเทรนแบบ minibatch ให้คำนวณ mean และ standard deviation ของ activation
- หักลบ activation ด้วย mean หารด้วย standard deviation

1	0.99	0.81	0.85	0.93		9.90	8.06	8.54	9.33
2	0.69	0.89	0.37	0.48		6.86	8.86	3.71	4.85
3	0.73	0.30	0.43	0.57		7.27	3.00	4.30	5.67
4	0.26	0.77	0.90	0.60		2.58	7.67	9.01	5.98
4	0.22	0.65	0.20	0.79		2.16	6.50	2.02	7.85
5	0.27	0.55	0.63	0.02		2.71	5.45	6.35	0.22
6	0.98	0.99	0.77	0.28		9.85	9.94	7.65	2.81
7	0.56	0.62	0.81	0.78		5.58	6.19	8.09	7.83
8	0.80	0.29	0.76	0.55		7.96	2.90	7.58	5.46
9	0.16	0.36	0.14	0.91		1.59	3.55	1.37	9.11
10	0.76	0.61	0.28	0.21		7.63	6.14	2.76	2.11
MEAN	0.58	0.62	0.56	0.56		5.83	6.21	5.58	5.57
STDEV	0.3	0.2	0.3	0.3		3.1	2.4	2.8	2.9

Unnormalized activations

1.32	0.79	1.05	1.29
0.33	1.13	-0.66	-0.25
0.47	-1.36	-0.46	0.03
-1.05	0.62	1.22	0.14
-1.19	0.12	-1.26	0.78
-1.01	-0.32	0.27	-1.82
1.30	1.58	0.74	-0.94
-0.08	-0.01	0.89	0.77
0.69	-1.40	0.71	-0.04
-1.37	-1.13	-1.50	1.21
0.58	-0.03	-1.00	-1.18
0.00	0.00	0.00	0.00
1	1	1	1

Batch-normalized activations

1	0.99	0.81	0.85	0.93		9.90	8.06	8.54	9.33
2	0.69	0.89	0.37	0.48		6.86	8.86	3.71	4.85
3	0.73	0.30	0.43	0.57		7.27	3.00	4.30	5.67
4	0.26	0.77	0.90	0.60		2.58	7.67	9.01	5.98
4	0.22	0.65	0.20	0.79		2.16	6.50	2.02	7.85
5	0.27	0.55	0.63	0.02		2.71	5.45	6.35	0.22
6	0.98	0.99	0.77	0.28		9.85	9.94	7.65	2.81
7	0.56	0.62	0.81	0.78		5.58	6.19	8.09	7.83
8	0.80	0.29	0.76	0.55		7.96	2.90	7.58	5.46
9	0.16	0.36	0.14	0.91		1.59	3.55	1.37	9.11
10	0.76	0.61	0.28	0.21		7.63	6.14	2.76	2.11
MEAN	0.58	0.62	0.56	0.56		5.83	6.21	5.58	5.57
STDEV	0.3	0.2	0.3	0.3		3.1	2.4	2.8	2.9

Unnormalized activations

(0.99 - 0.58) /0.3 = 1.32
(9.90 - 5.83) /3.1 = 1.32

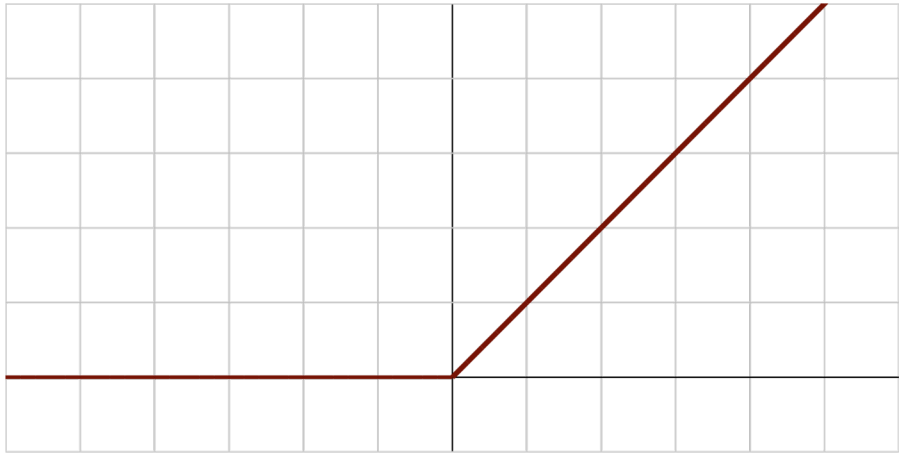
Batch-normalized activations

Batch Normalization คืออะไร

- ตอนเทรนแบบ minibatch ให้คำนวณ mean และ standard deviation ของ activation
- หักลบ activation ด้วย mean หารด้วย standard deviation
  - เพื่อให้ activation ของแต่ละ layer มีค่าใกล้ๆ กัน โดยให้ mean = 0 และ standard deviation = 1
  - เรียกอีกอย่างว่า normalization คือ standardization

Why Batch Norm works?

relu ไม่จำกัดว่า activation จะเป็นเท่าไร ถ้า parameter เราใหญ่มาก ค่า activation ก็ใหญ่มาก



$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

ข้อดีของ Batch Normalization

- ไม่ต้องแคร์ว่าค่าเริ่มต้นของ parameter ตอนเทรนเป็นเท่าไร เพราะยังไงก็ normalize ตอนหลัง
- learning rate ใหญ่ๆ ได้ จะได้ learn ได้เร็ว ๆ (ไม่ต้องใช้ epoch เยอะ)

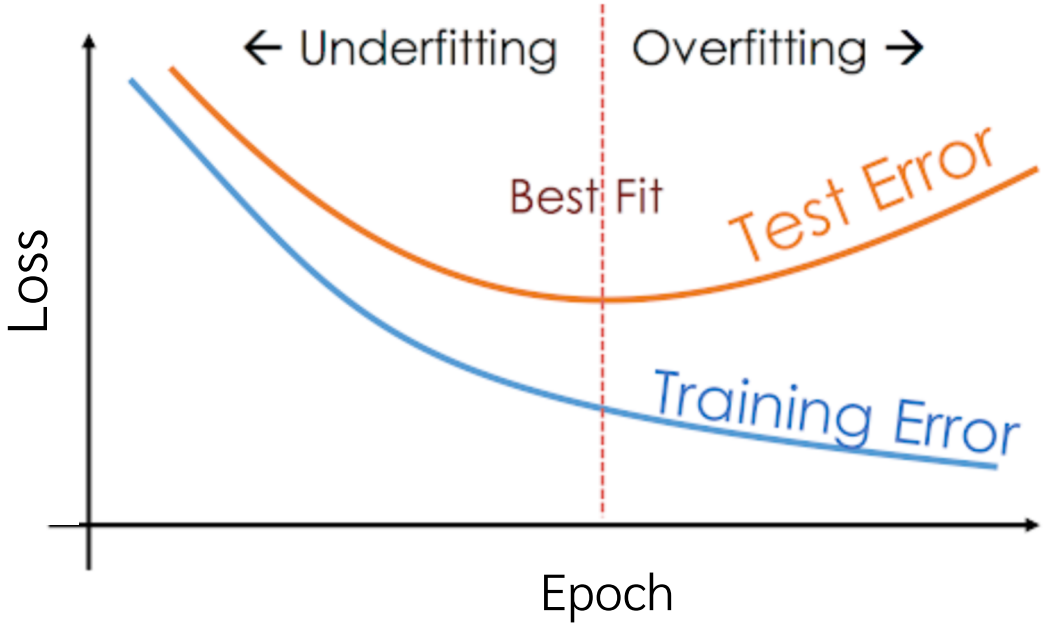
Batch Normalization in Keras

$$x_i \longleftarrow \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$$

```
tf.keras.layers.BatchNormalization(
    axis=-1,
    momentum=0.99,
    epsilon=0.001,
    center=True,
    scale=True,
    beta_initializer="zeros",
    gamma_initializer="ones",
    moving_mean_initializer="zeros",
    moving_variance_initializer="ones",
    beta_regularizer=None,
    gamma_regularizer=None,
    beta_constraint=None,
    gamma_constraint=None,
    renorm=False,
    renorm_clipping=None,
    renorm_momentum=0.99,
    fused=None,
    trainable=True,
    virtual_batch_size=None,
    adjustment=None,
    name=None,
    **kwargs
)
```

Regularization for Deep Learning

Typical Learning Curve



Deep Learning tends to overfit

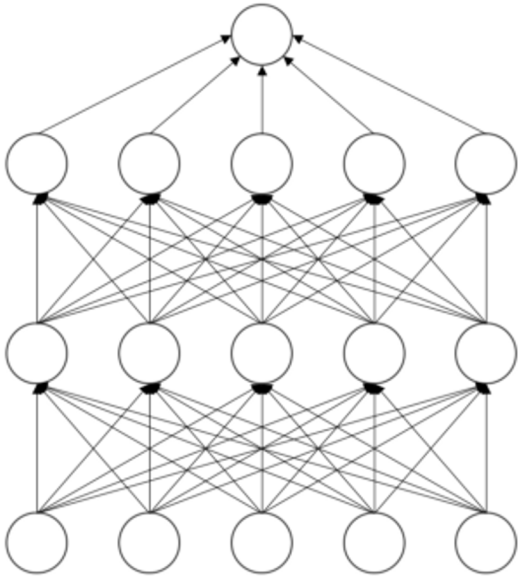
- Deep Learning model มักจะมีจำนวน parameter (weights) มากกว่าปริมาณข้อมูล ดังนั้น model มักจะเทรนจน overfit
- ถ้าเทรนไปเรื่อยๆ จะเห็นว่า accuracy บน training set จะเข้าใกล้ 100% มาก

Regularization เป็นเทคนิคที่ทำให้ overfitting น้อยลง

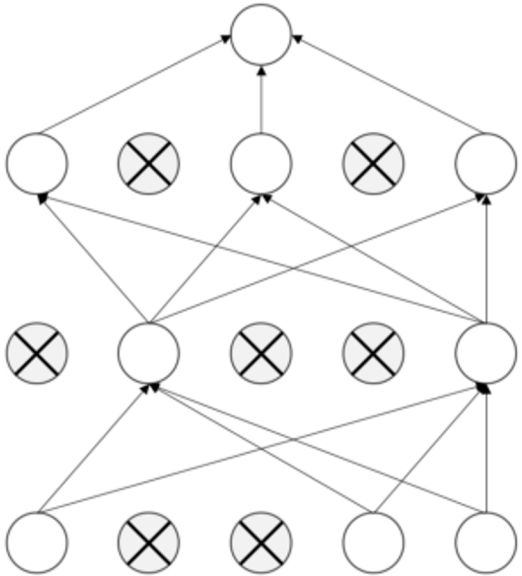
Regularization Techniques

- Dropout
- L2 Regularization

Dropout Regularization



Standard Neural Net

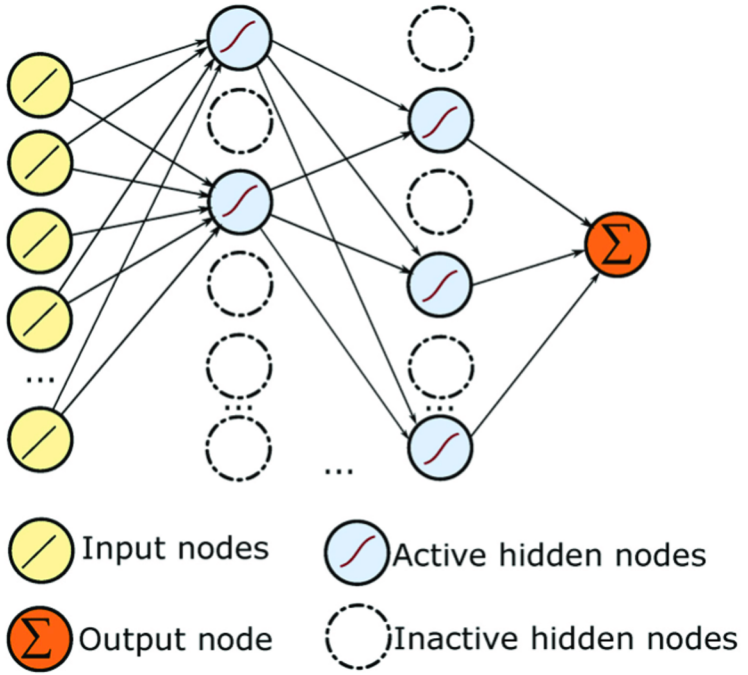


After applying dropout

https://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

Dropout

- **p = อัตราการดรอป unit**
- ช่วยให้ไม่ให้ w อันใดอันหนึ่งมีอำนาจมากจนเกินไปจน w อื่นไม่ได้เรียนรู้อะไรจาก data เลย



https://www.researchgate.net/figure/Neural-network-with-dropout_fig5_332450026

Dropout in practice

CLASS `torch.nn.Dropout(p: float = 0.5, inplace: bool = False)` [SOURCE]

During training, randomly zeroes some of the elements of the input tensor with probability `p` using samples from a Bernoulli distribution. Each channel will be zeroed out independently on every forward call.

This has proven to be an effective technique for regularization and preventing the co-adaptation of neurons as described in the paper Improving neural networks by preventing co-adaptation of feature detectors .

Furthermore, the outputs are scaled by a factor of $\frac{1}{1-p}$ during training. This means that during evaluation the module simply computes an identity function.

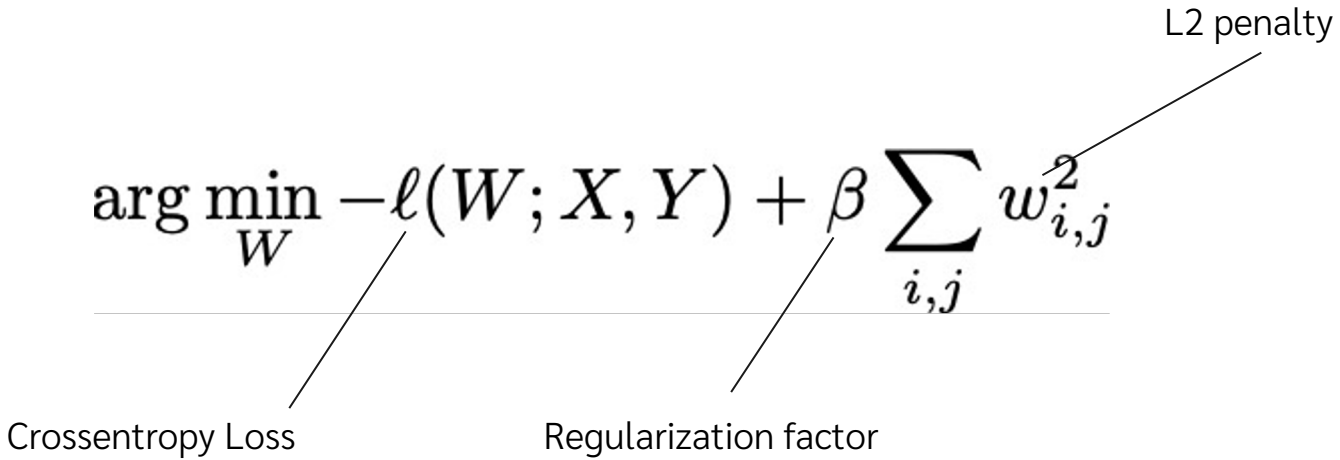
Parameters

- **p** – probability of an element to be zeroed. Default: 0.5
- **inplace** – If set to `True`, will do this operation in-place. Default: `False`

Regularization Techniques

- Dropout
- **L2 Regularization**

Optimizing loss and penalty term



L2 Regularization in Keras

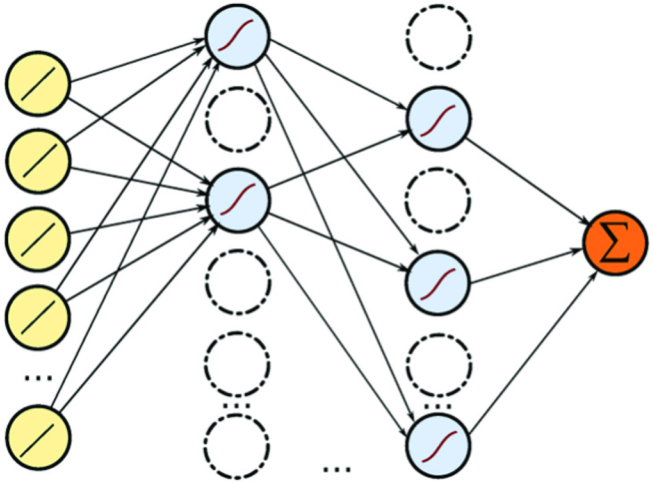
```
from tensorflow.keras import layers
from tensorflow.keras import regularizers

layer = layers.Dense(
    units=64,
    kernel_regularizer=regularizers.l2(l2=1e-4),
    bias_regularizer=regularizers.l2(l2=1e-4)
)
```

Regularization

+ ทำให้ไม่ต้องห่วงมากว่าโมเดลจะ overfit เทรนไปนานๆ เลยก็ได้
- ต้องปรับจูน dropout rate และ regularization factor

ควรใช้ทุกครั้งเพราะทำให้ผลดีขึ้นมาก



$$\arg\min_{W} -\ell(W; X, Y) + \beta \sum_{i,j} w_{i,j}^2$$