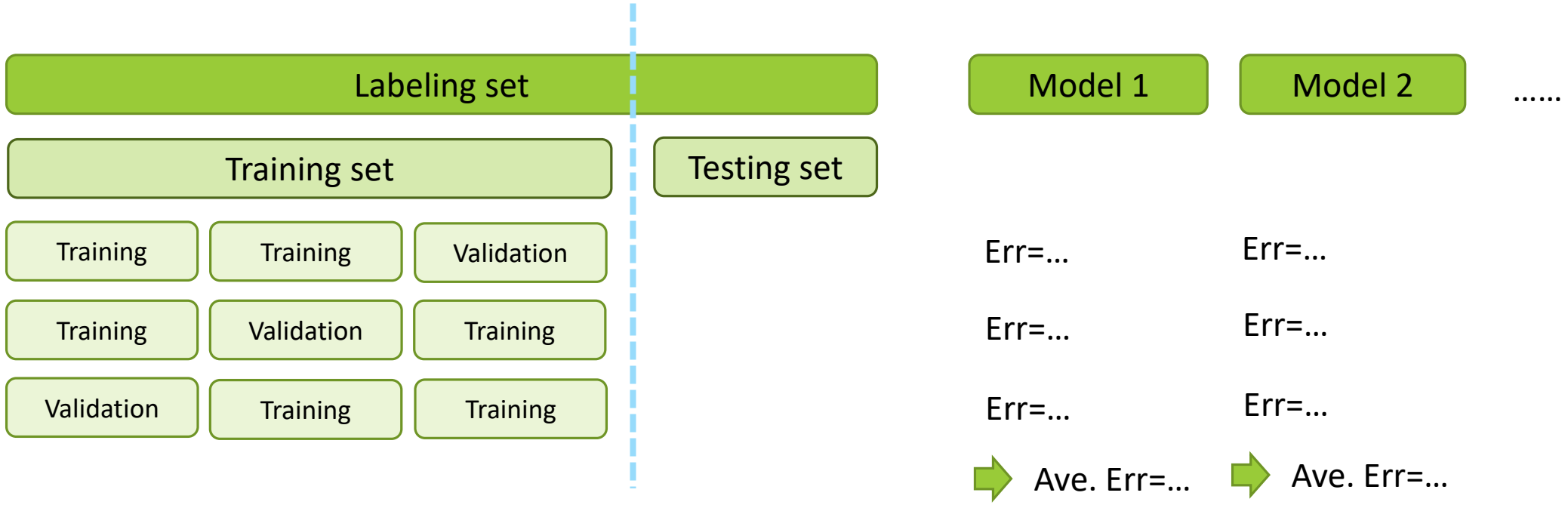
機器學習於材料資訊的應用
Machine Learning on Material Informatics

陳南佑(NAN-YOW CHEN)

楊安正(AN-CHENG YANG)

Cross validation – N fold



- Pick the model which has the smallest average error.
- Use all training set to train machine again.

- Apply machine (neuron network) to the testing set to see the accuracy.

https://www.youtube.com/watch?v=xki61j7z-30

Overfitting

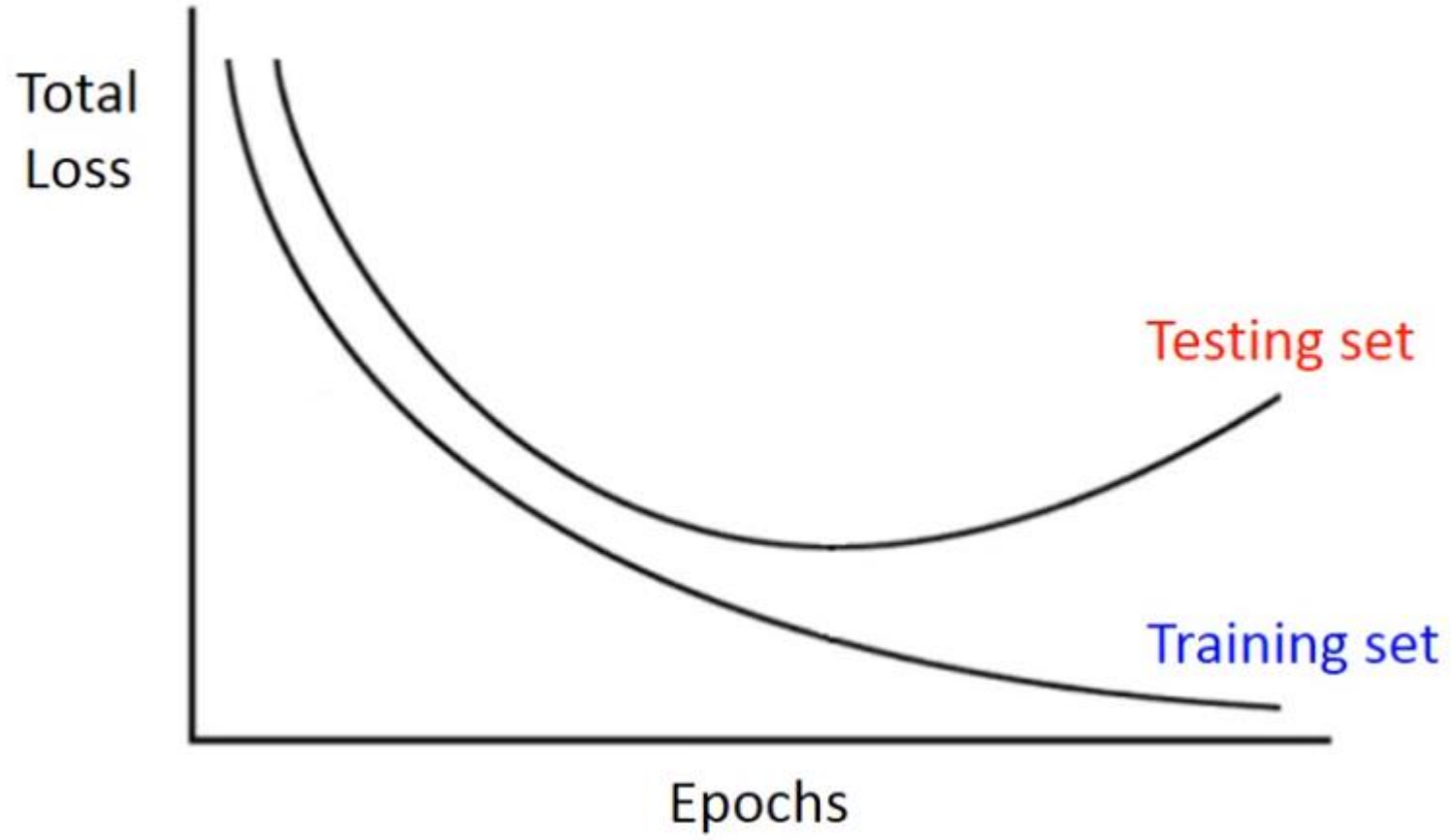
https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

How to detect overfitting?

	Low *Training* Error	High *Training* Error
Low *Testing* Error	The model is learning!	Probably some error in your code. Or you've created a *psychic* AI.
High *Testing* Error	OVERFITTING	The model is not learning.

https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

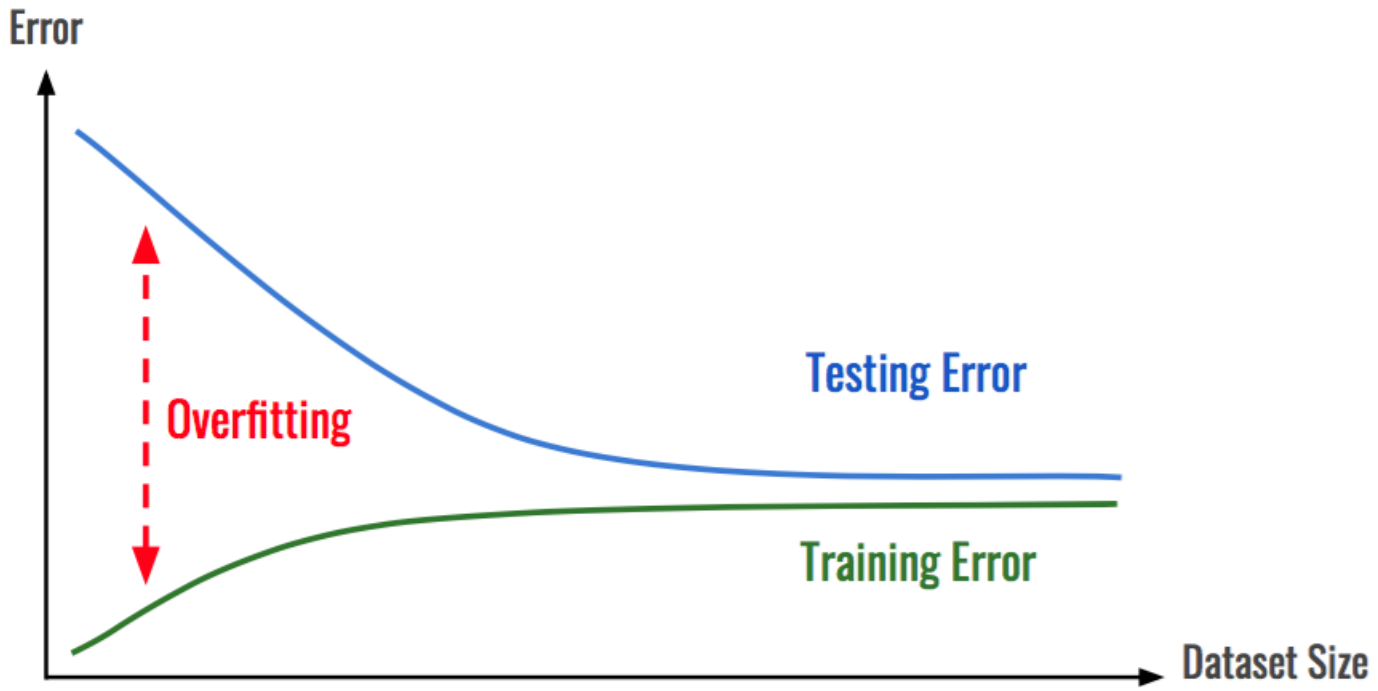
How to detect overfitting?



https://www.youtube.com/watch?v=xki61j7z-30

How to prevent overfitting?

- From data:
  - Gather more data



https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

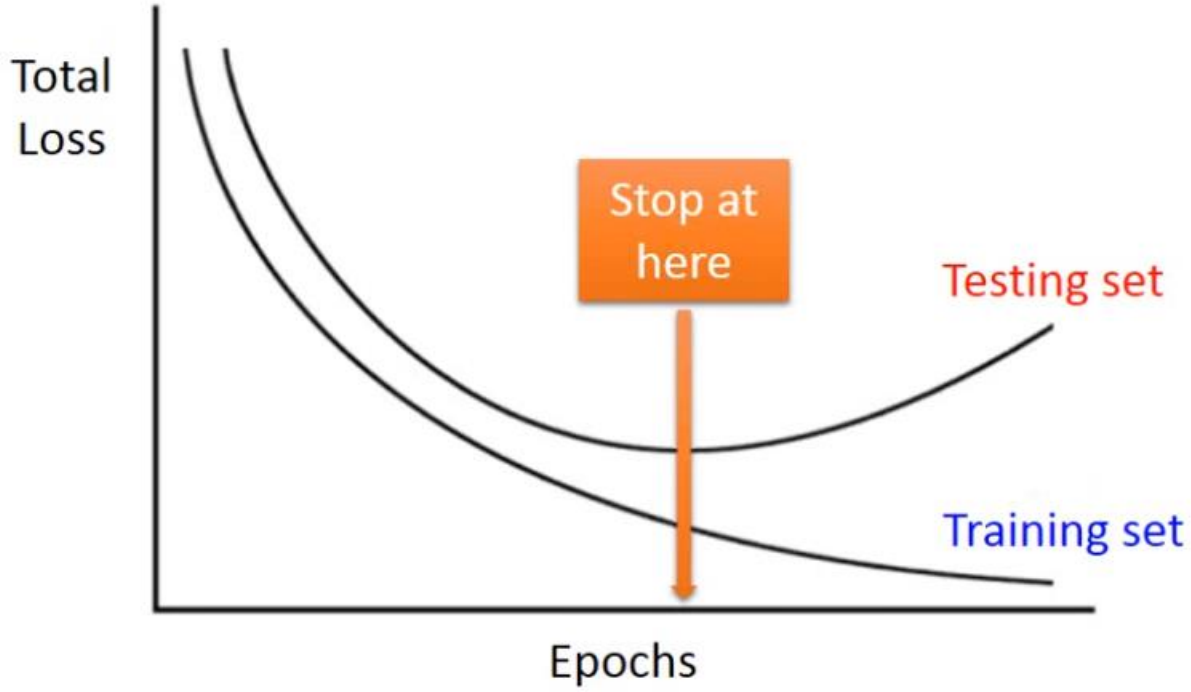
How to prevent overfitting?

- From data:
  - Gather more data
  - Data augmentation & Noise

https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

How to prevent overfitting?

- From model:
  - Early termination (early stop)



https://www.youtube.com/watch?v=xki61j7z-30

How to prevent overfitting?

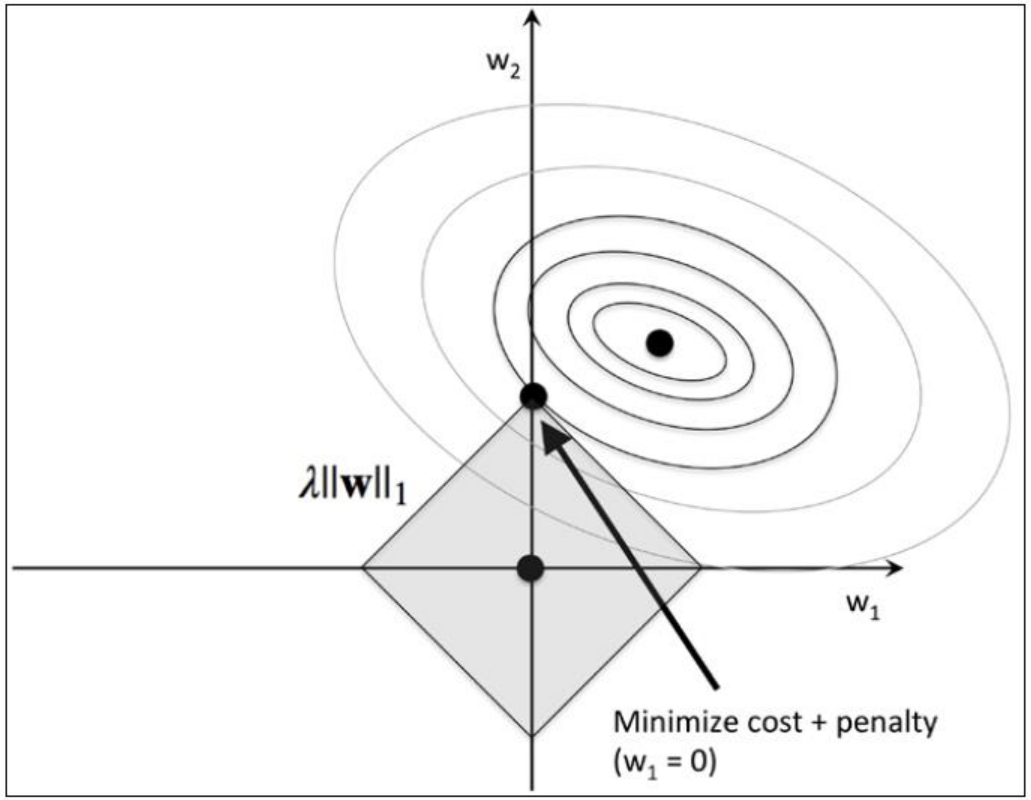
- From model:
  - Early Termination (early stop)
  - Regularization (weight decay): L1 & L2

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$



http://laid.delanover.com/difference-between-l1-and-l2-regularization-implementation-and-visualization-in-tensorflow/

How to prevent overfitting?

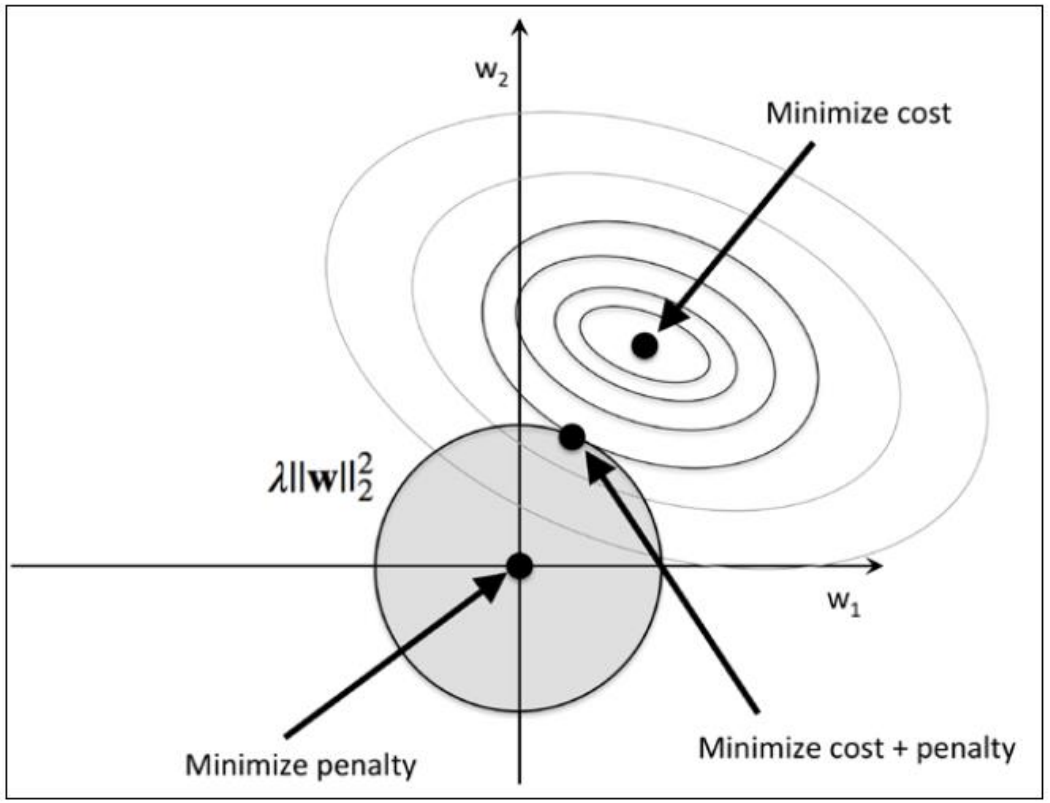
- From model:
  - Early Termination (early stop)
  - Regularization (weight decay): L1 & L2

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

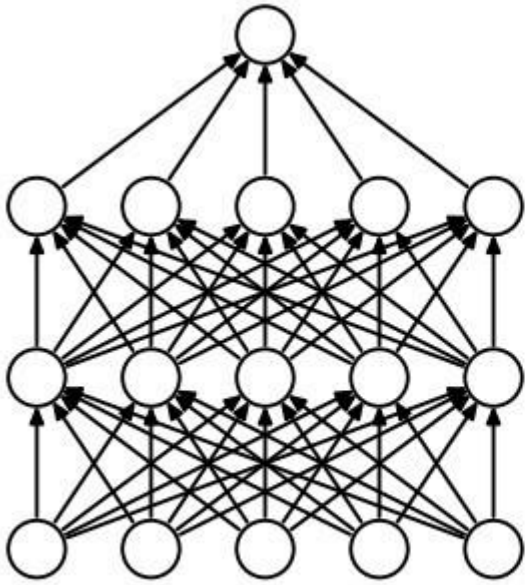
L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

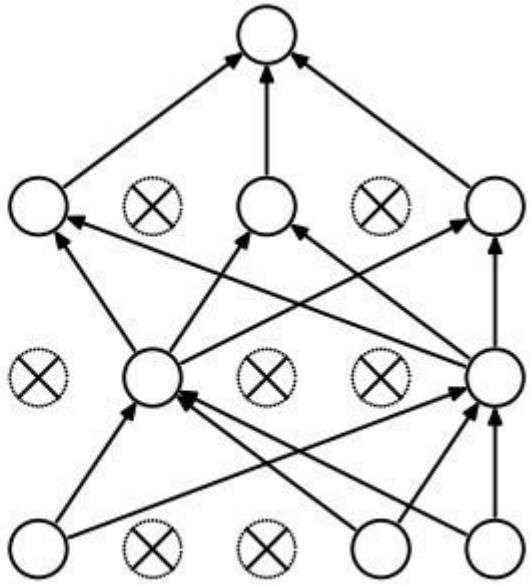


How to prevent overfitting?

- From model:
  - Early Termination (early stop)
  - Regularization (weight decay): L1 & L2
  - Dropout



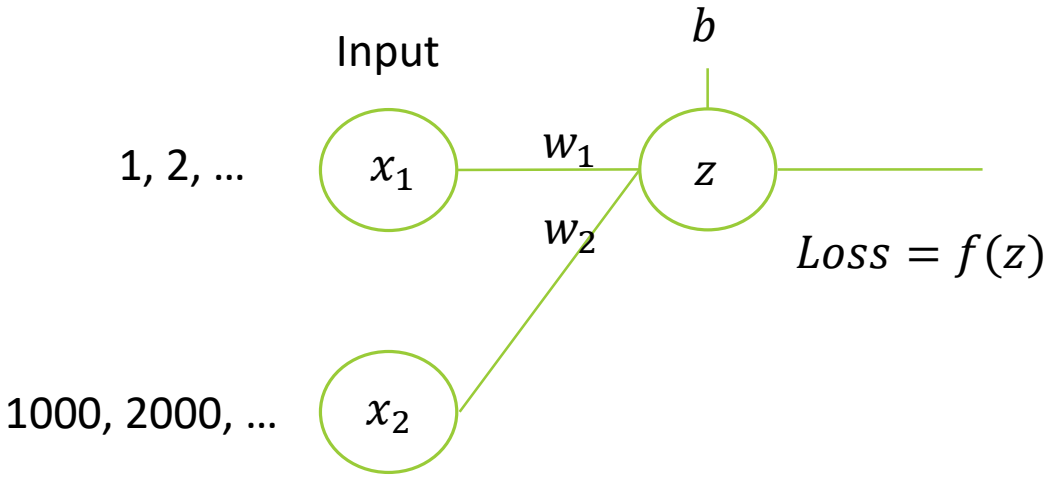
(a) Standard Neural Net

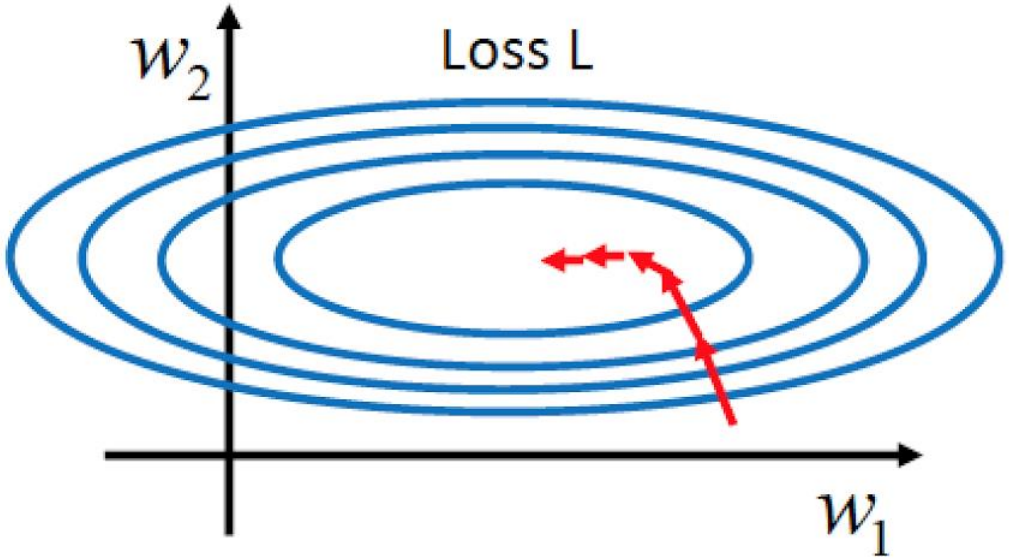


(b) After applying dropout.

http://laid.delanover.com/difference-between-l1-and-l2-regularization-implementation-and-visualization-in-tensorflow/

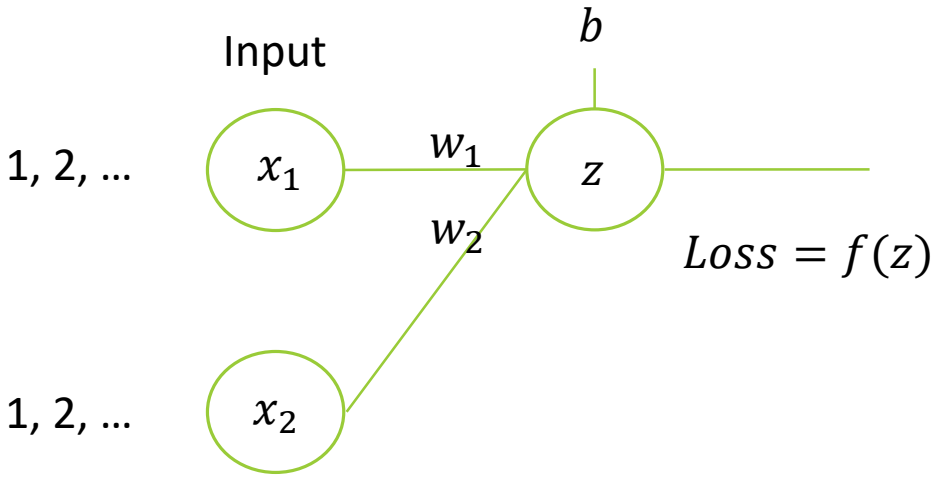
Feature scaling

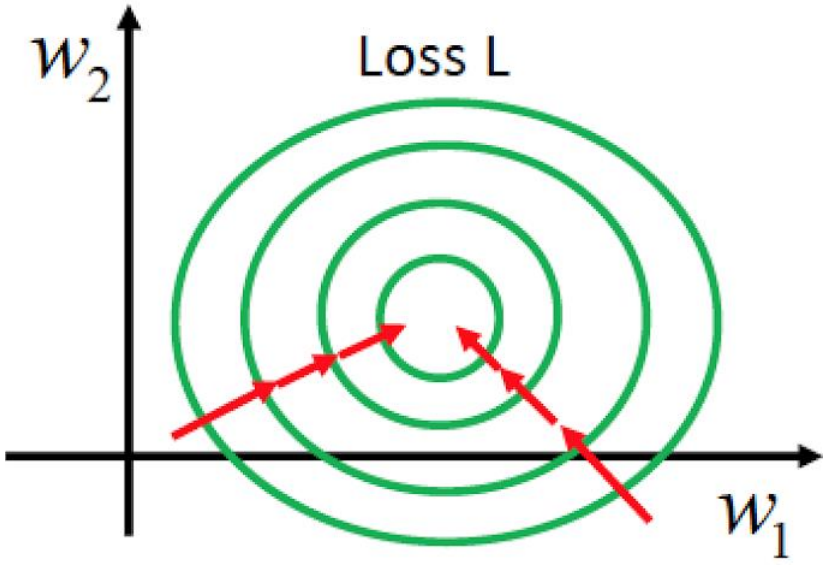




https://en.wikipedia.org/wiki/Feature_scaling & https://www.youtube.com/watch?v=BZh1ltr5Rkg

Feature scaling





https://en.wikipedia.org/wiki/Feature_scaling & https://www.youtube.com/watch?v=BZh1ltr5Rkg

Feature scaling

- Rescaling (min-max normalization)

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

- Mean normalization

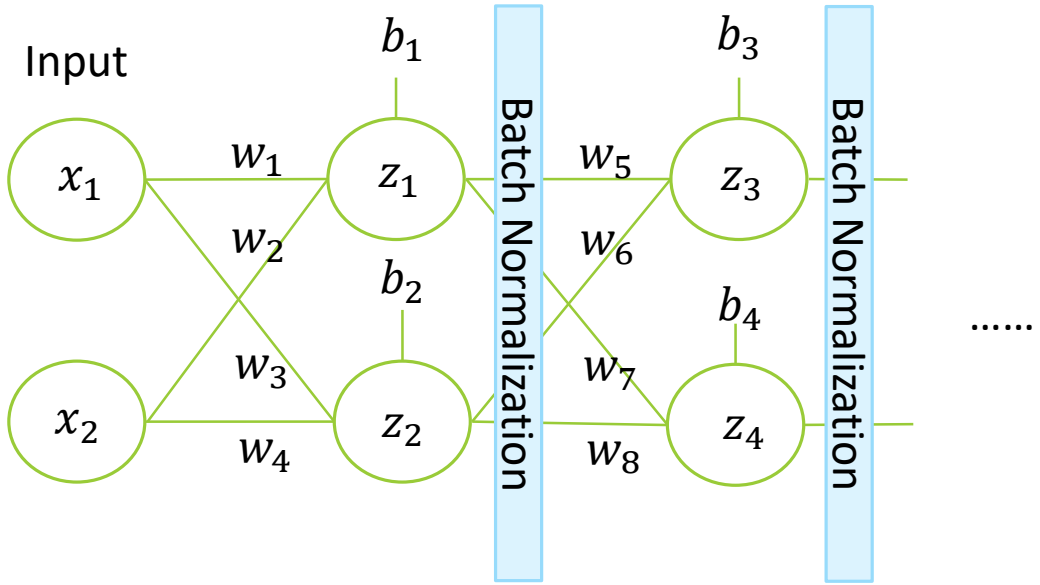
$$x' = \frac{x - \text{average}(x)}{\max(x) - \min(x)}$$

- Standardization (Z-score)

$$x' = \frac{x - \bar{x}}{\sigma}$$

where σ is its standard deviation.

How about hidden layers?



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

➔ Use batch normalization!

tf.layers.batch_normalization()
keras.layers.BatchNormalization()

https://arxiv.org/abs/1502.03167

Batch normalization

- Usually added before activation function.

- Can reduce training time.
- Can use high learning rate.
- Let learning is less affected by initialization of weights.
- Can build more deep network.
- Can reduce the demand for regularization.

https://arxiv.org/abs/1502.03167

Next class

- Case study