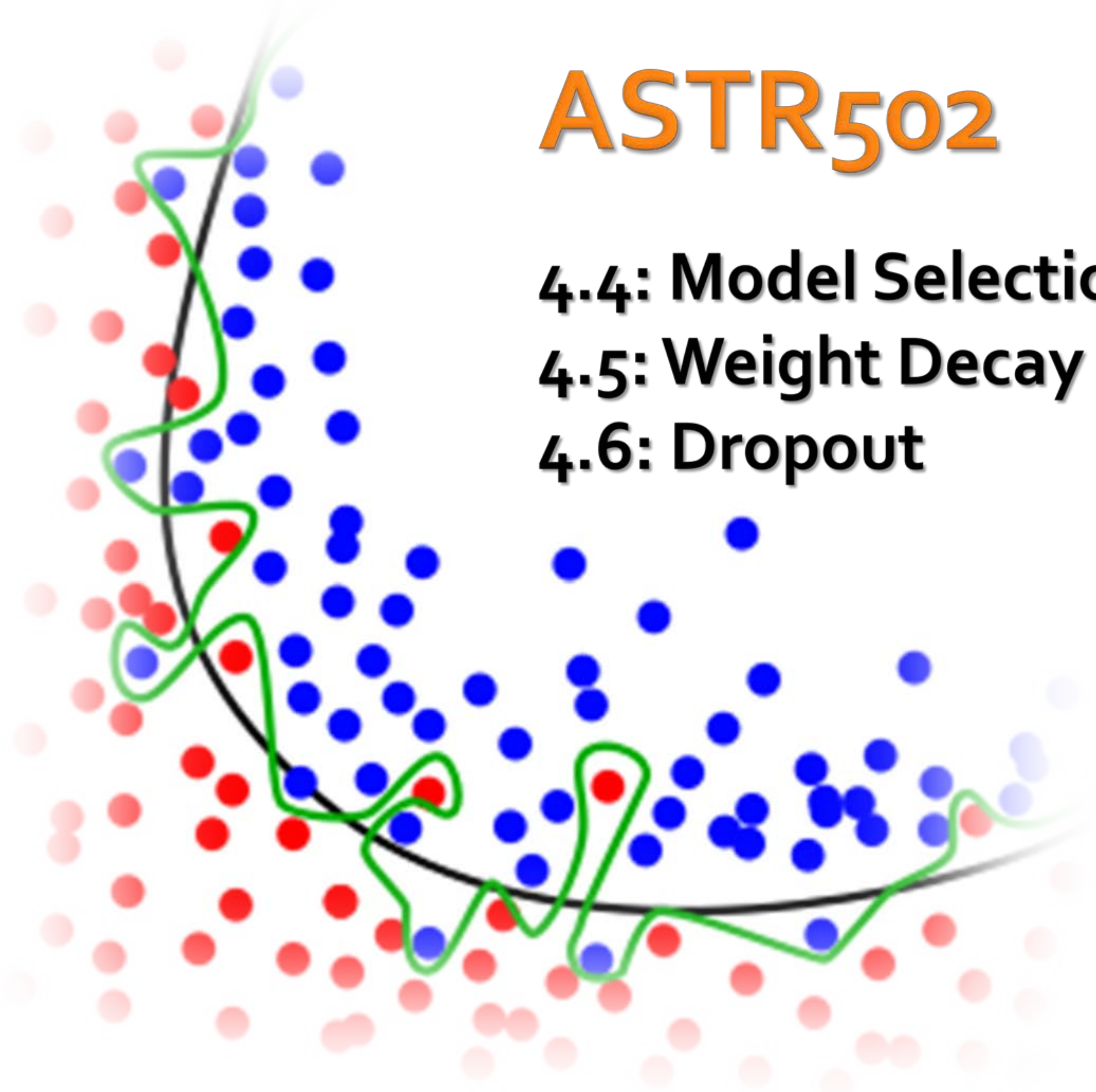
ASTR502

4.4: Model Selection, Underfitting, Overfitting
4.5: Weight Decay
4.6: Dropout



Donghyeon Jeff Khim

Image credit: https://upload.wikimedia.org/wikipedia/commons/1/19/Overfitting.svg

The fundamental problem (and goal) of ML

- Discovering general patterns
  - General pattern vs. (Simply) memorized data
  - Our predictions will only be useful if our model has truly discovered a **general pattern**

Training data

DB
ML

Test (Trained)

DB - Result: Spiral
ML - Result: Spiral

Test (New)

DB - Result: Not found
ML - Result: Spiral

Image credit: https://wp-assets.futurism.com/2013/11/suuer1.jpg

Image credit: https://www.eso.org/public/images/eso9845d/

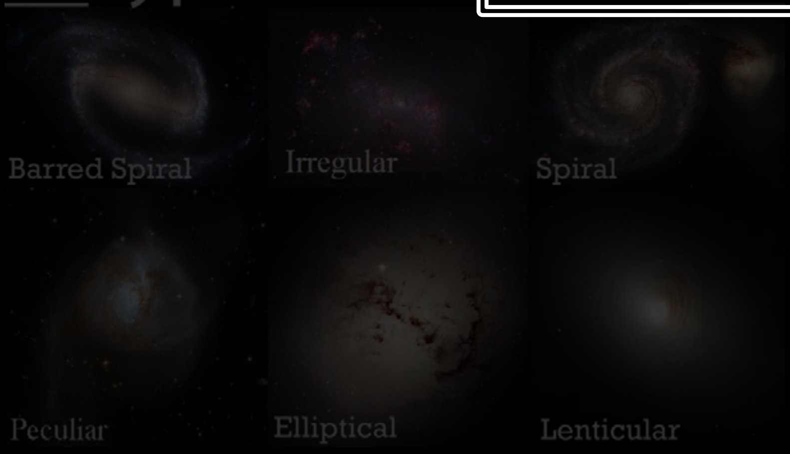
The fundamental problem (and goal) of ML

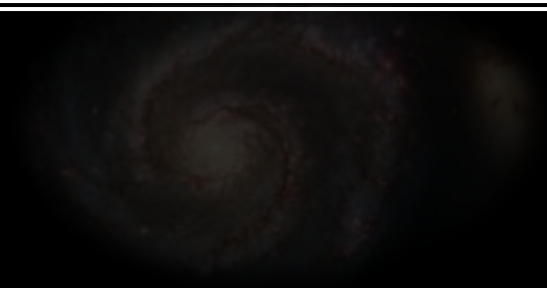
- Discovering general patterns
  - General pattern vs. (Simply) memorized data
  - Our predictions will only be useful if our model has truly discovered a **general pattern**

How much we can trust the results from the machine learning?

Training data | Test (New)

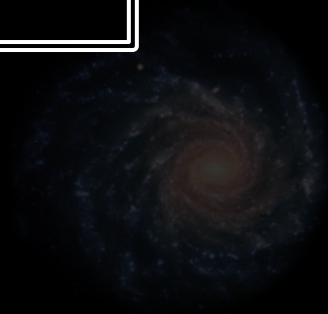
Types of Galaxies





DB - Result: Spiral

ML - Result: Spiral



DB - Result: Not found

ML - Result: Spiral

Image credit: https://wp-assets.futurism.com/2013/11/suuer1.jpg

Image credit: https://www.eso.org/public/images/eso9845d/

Training Error & Generalization Error

- **Training Error**: the error calculated on the training dataset
- **Generalization Error**: model's error for an infinite amount of data
  - We can never calculate the generalization error exactly → **Expectation**

Training data

Training Error
Accuracy : 99%
(Measured by training sets)

Real data

Generalization Error
Accuracy : 99%
(Estimated)

ML - Result: Spiral

Image credit: https://wp-assets.futurism.com/2013/11/suuer1.jpg

Image credit: https://www.eso.org/public/images/eso9845d/

Training Error & Generalization Error

- **Training Error**: the error calculated on the training dataset
- **Generalization Error**: model's error for an infinite amount of data
  - We can never calculate the generalization error exactly → **Expectation**

Training data

Training Error
Accuracy : 99%
(Measured by training sets)

Real data

Generalization Error
Accuracy : 80%
(Estimated)

Overfitting!

ML - Result: Spiral

Image credit: https://wp-assets.futurism.com/2013/11/suuer1.jpg

Image credit: https://www.eso.org/public/images/eso9845d/

Underfitting and Overfitting (Regression)

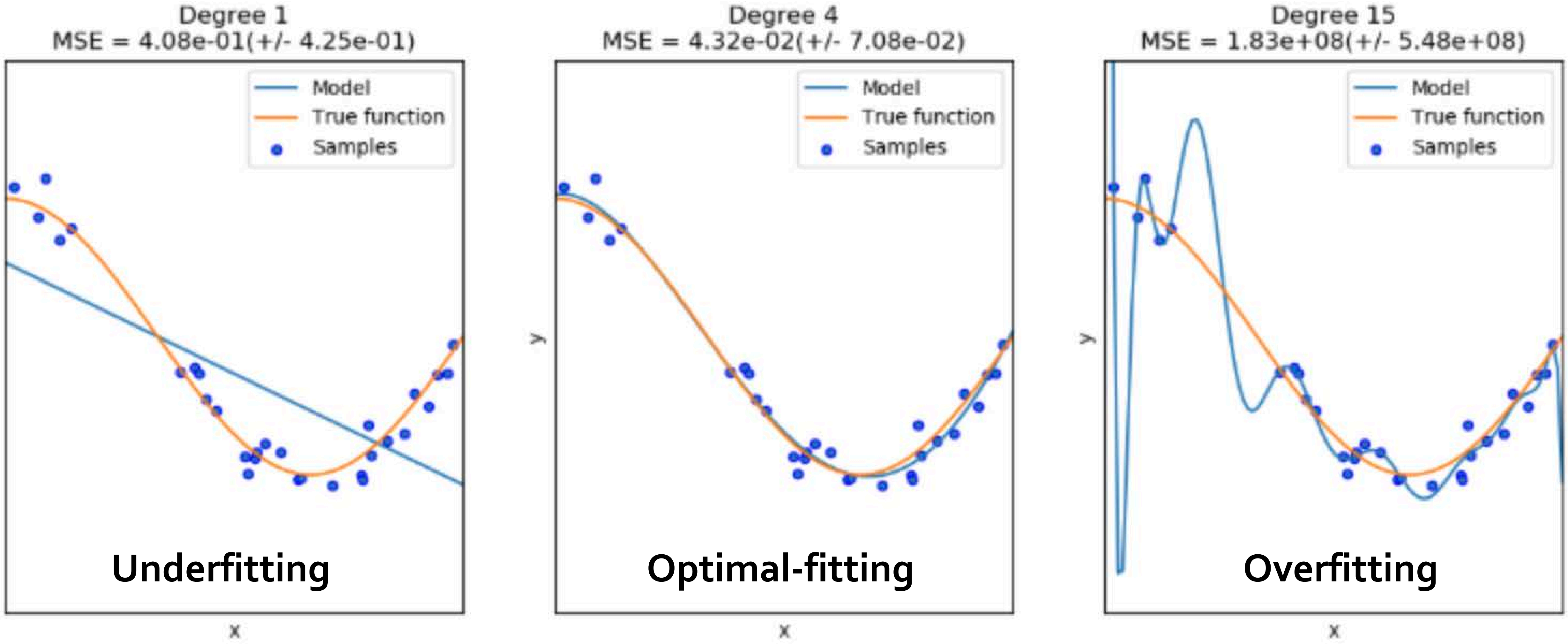
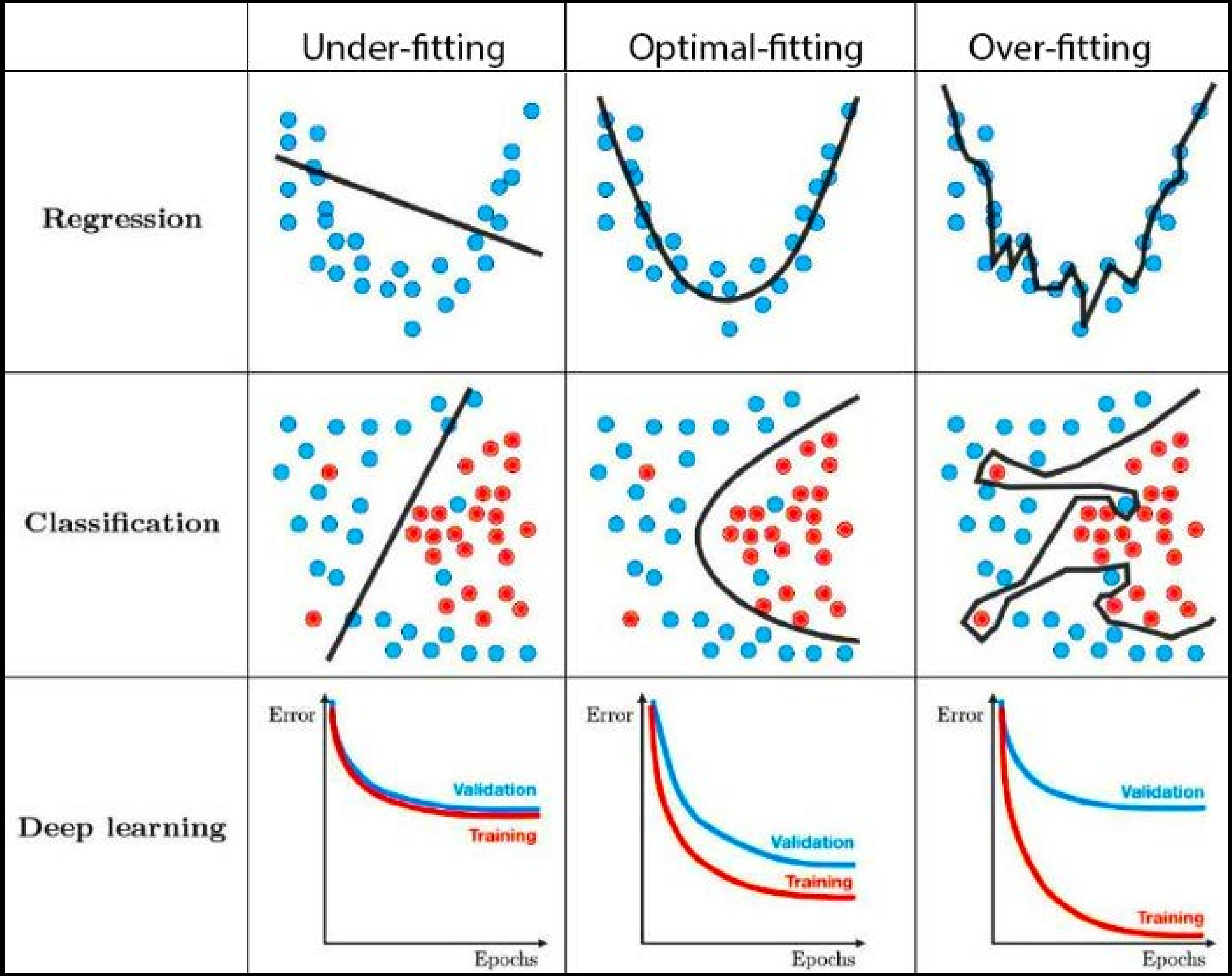


Image credit: https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting

Underfitting and Overfitting



Real data

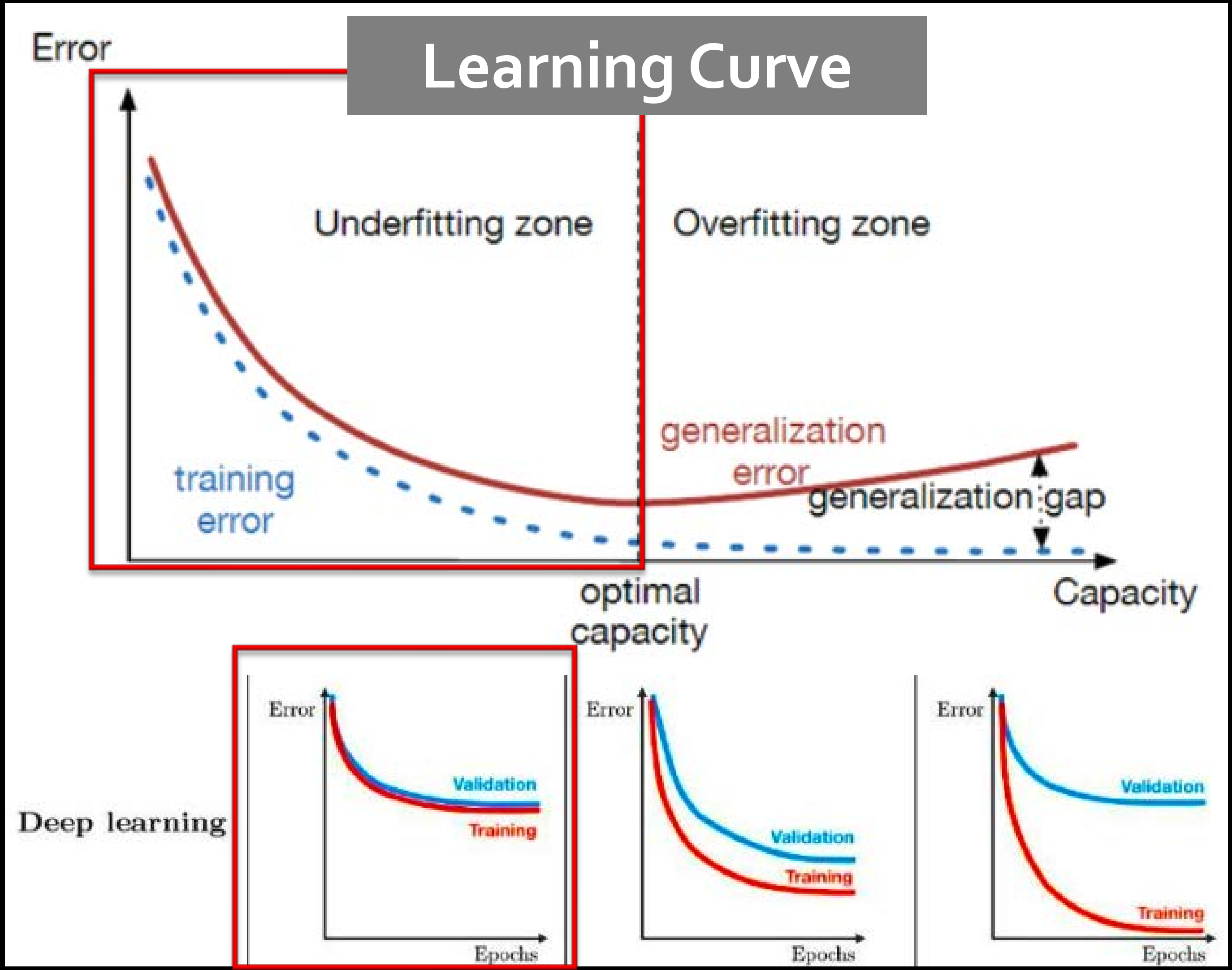
Generalization Error
Accuracy : 80%
(Estimated)

Training data

Training Error
Accuracy : 99%
(Measured by training sets)

Image credit: https://i.pinimg.com/originals/72/e2/22/72e222c1542539754df1d914cb671bd7.png

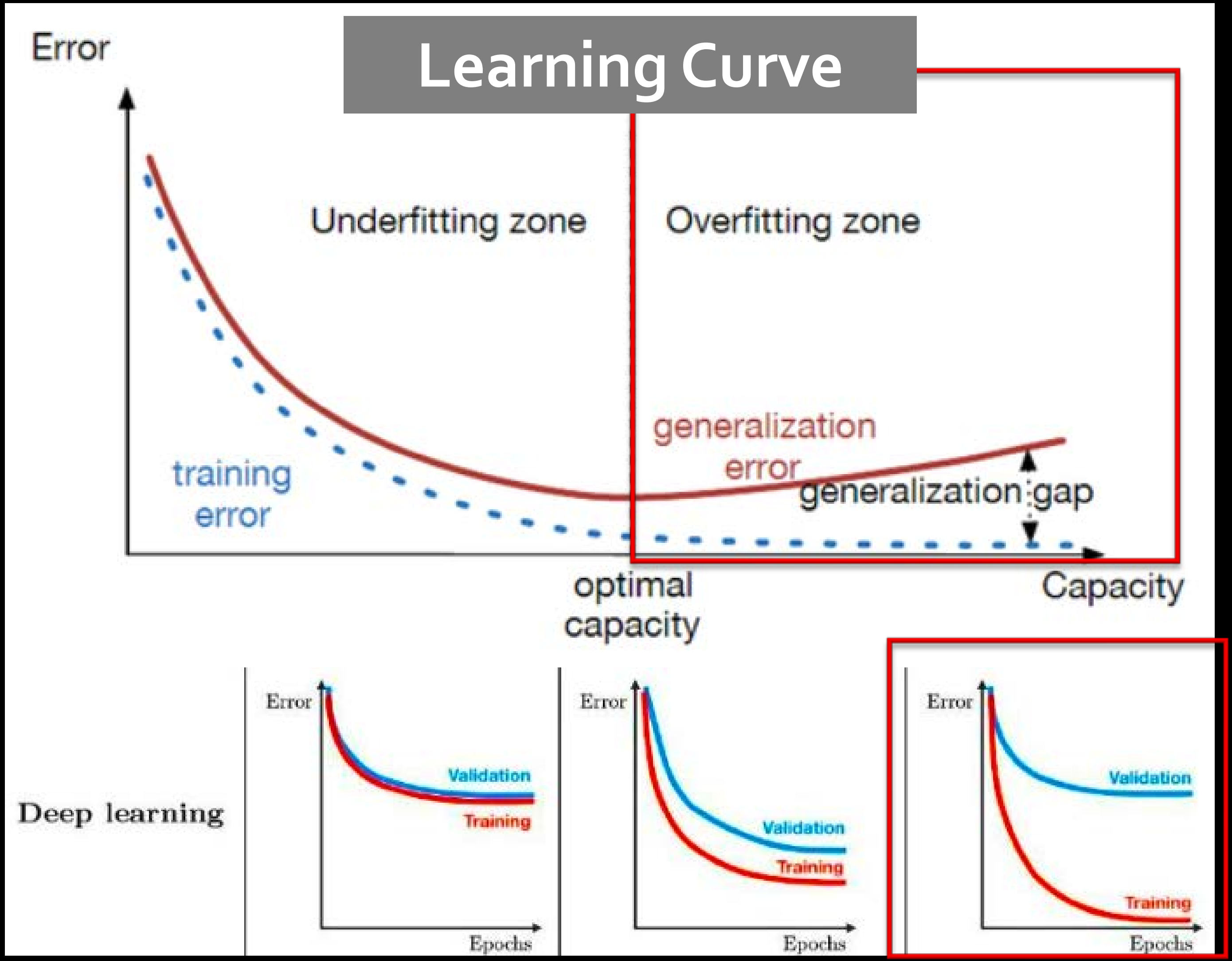
Underfitting and Overfitting



- Underfitting
  - Training & validation errors are both substantial
    - The model is too simple to capture the pattern
  - Small generalization gap
    - We may use more complex modes
  - High bias
    - Not be able to fit data well

Image credit: https://srdas.github.io/DLBook2/ImprovingModelGeneralization.html

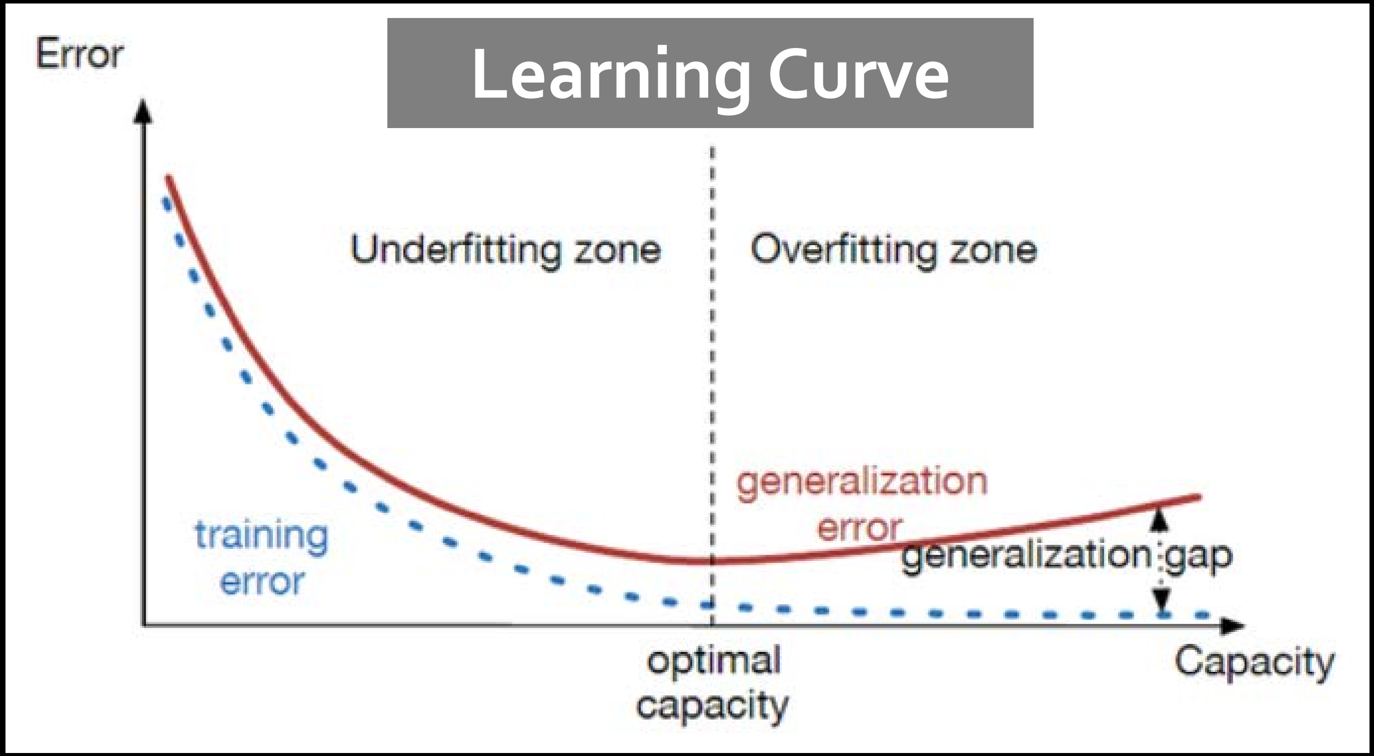
Underfitting and Overfitting



- Overfitting
  - Training error is significantly lower than the validation error
    - Huge generalization gap
  - High variance
    - Performs specifically well under certain noise realization of data

Image credit: https://srdas.github.io/DLBook2/ImprovingModelGeneralization.html

Underfitting and Overfitting



- **Training Error**: the error calculated on the training dataset
- **Generalization Error**: model's error for an infinite amount of data
  - We can never calculate the generalization error exactly → **Expectation**

How can we measure (estimate) the errors?

Image credit: https://srdas.github.io/DLBook2/ImprovingModelGeneralization.html

Training set, Test set

Data set (with labels)

Training set | **Test set**

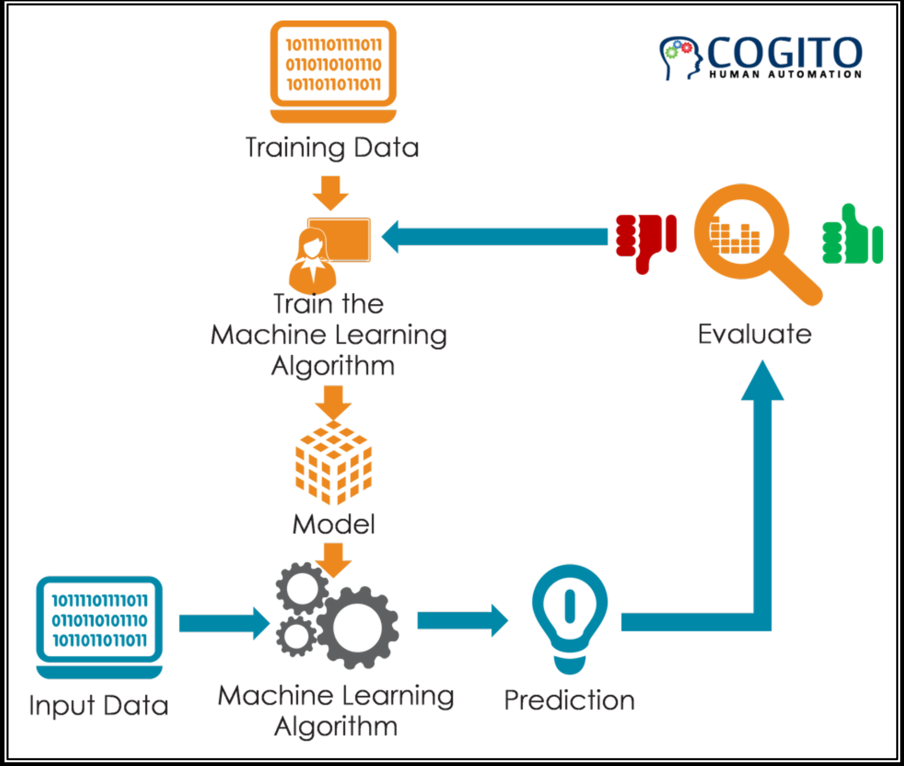


Image credit: https://machinelearningasaservice.weebly.com/

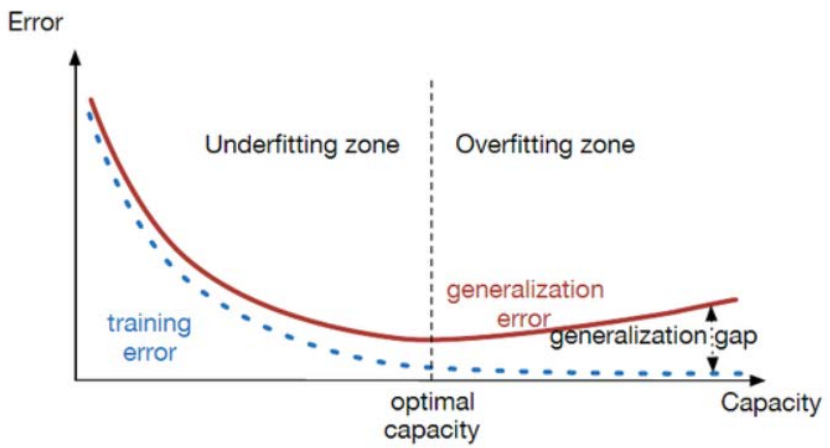
- **Training Set**
  - Dataset that we use to train the model to determine the network parameters (weights and biases)
- **Test Set**
  - Evaluate the network
  - Provide an **unbiased estimate** on the performance of the final network.
  - We ***may*** measure the validation error based on the test set

Can we say the test set did not affect the model?

Training set, Test set

Data set (with labels)

Training set | **Test set**



Check Overfitting, Underfitting

Hyperparameters - # of layers, # of hidden units per layer, Batch size, learning rate…

- **Test set**
  - We should not touch our test set until after we have chosen all our hyperparameters.
  - There is a risk that we **overfit** the test data
  - → We should never rely on the test data for model selection.

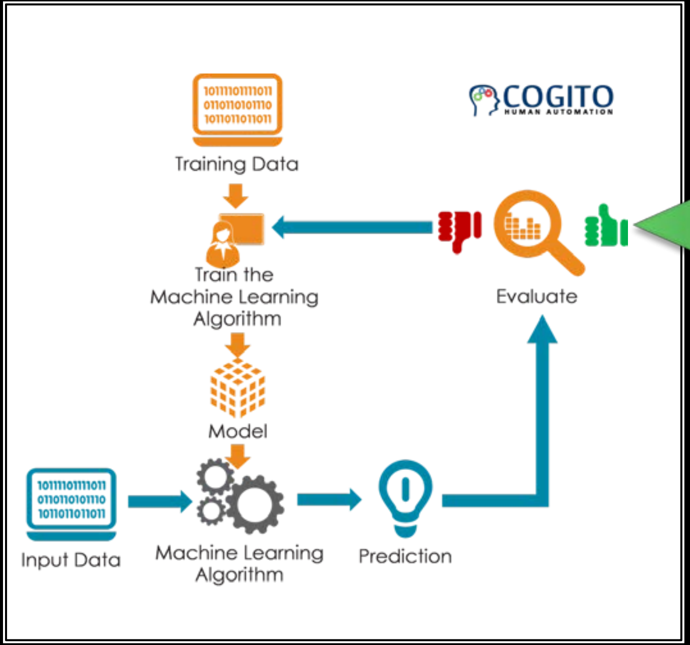
Image credit: https://srdas.github.io/DLBook2/ImprovingModelGeneralization.html

Training set, Test set & Validation set

Data set (with labels)

Training set | **Test set**

Training set | **Validation set** | **Test set**



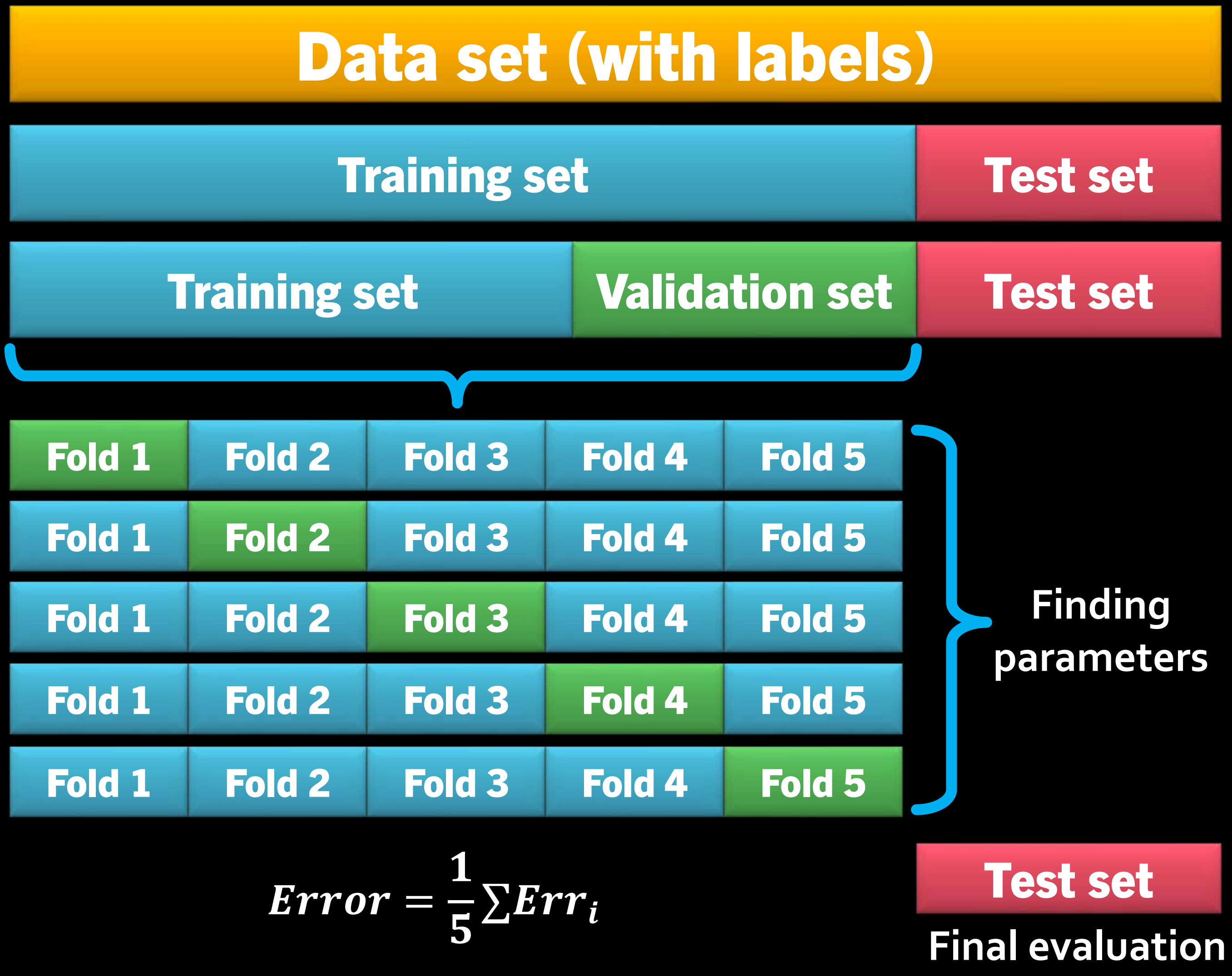
Validate model

Final evaluation
→ Unbiased estimate on the performance

- **Test set**
  - We should not touch our test set until after we have chosen all our hyperparameters.
  - There is a risk that we **overfit** the test data
  - → We should never rely on the test data for model selection.
- **Validation set**
  - Validate model performance during training.
  - Provide information on the generalizability of our model to unseen data
  - Helpful to prevent overfitting

Image credit: https://machinelearningasaservice.weebly.com/

K-Fold Cross-Validation



$$Error = \frac{1}{5}\sum Err_i$$

- When training data is **scarce**
  - We might not even be able to afford to hold out enough data to constitute a proper validation set.
- K-fold cross-validation
  - The original training data is split into K non-overlapping subsets.
  - Model training and validation are executed K times
  - Training and validation errors are estimated by averaging K errors

Model Complexity

- Complexity & Fitting
  - Underfitting
    - Simple models and abundant data
    - Generalization error ≅ training error
  - Overfitting
    - More complex models and smaller data
    - Training error ↓, generalization gap ↑

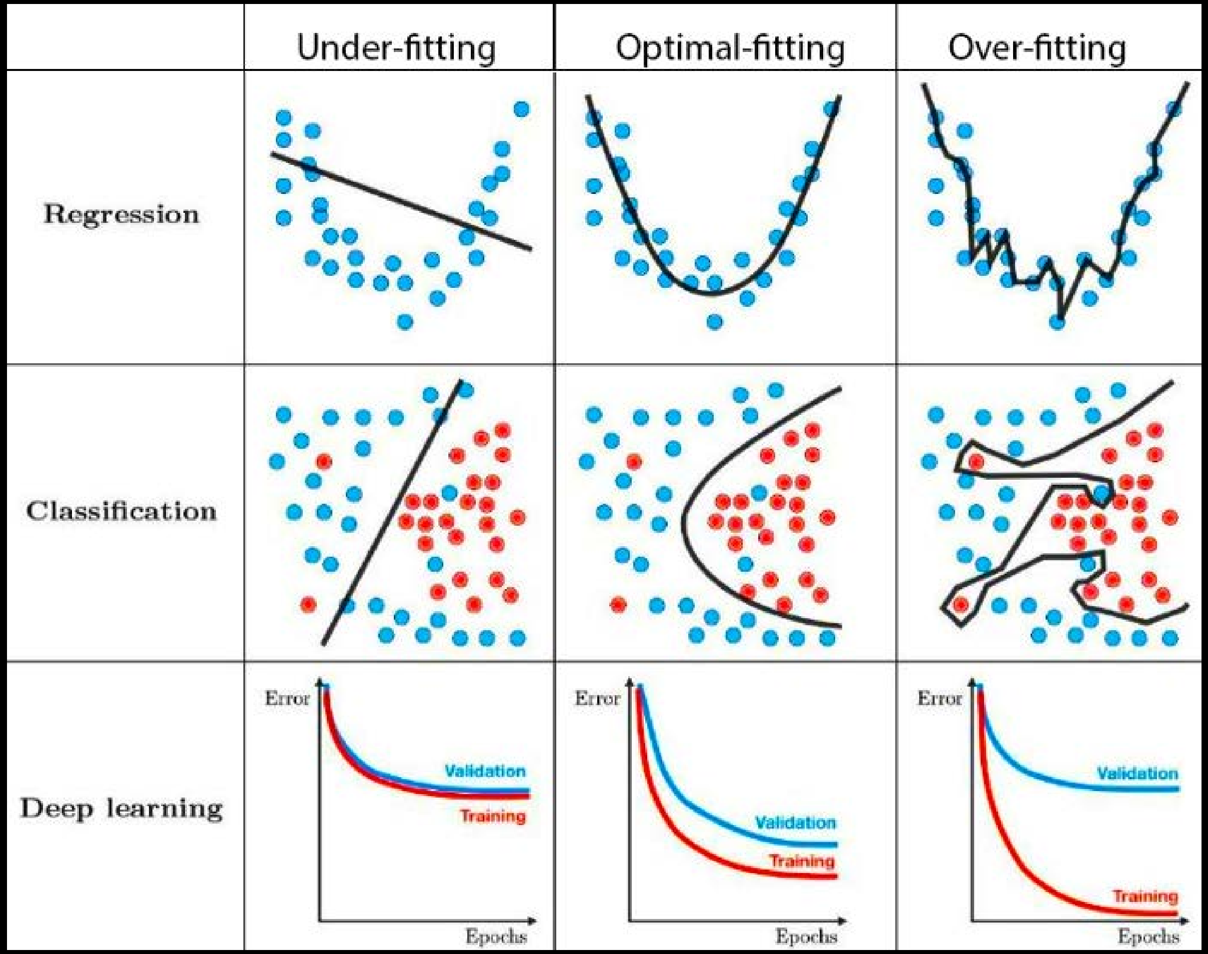


Image credit: https://i.pinimg.com/originals/72/e2/22/72e222c1542539754df1d914cb671bd7.png

Model Complexity

Complexity & Size of data

- Complexity ↑ for…
  1. A model with more parameters
     - Degree of Freedom (DoF)
  2. A model whose parameters can take a wider range of values
     - When weights can take a wider range of values
  3. A model that takes more training iterations

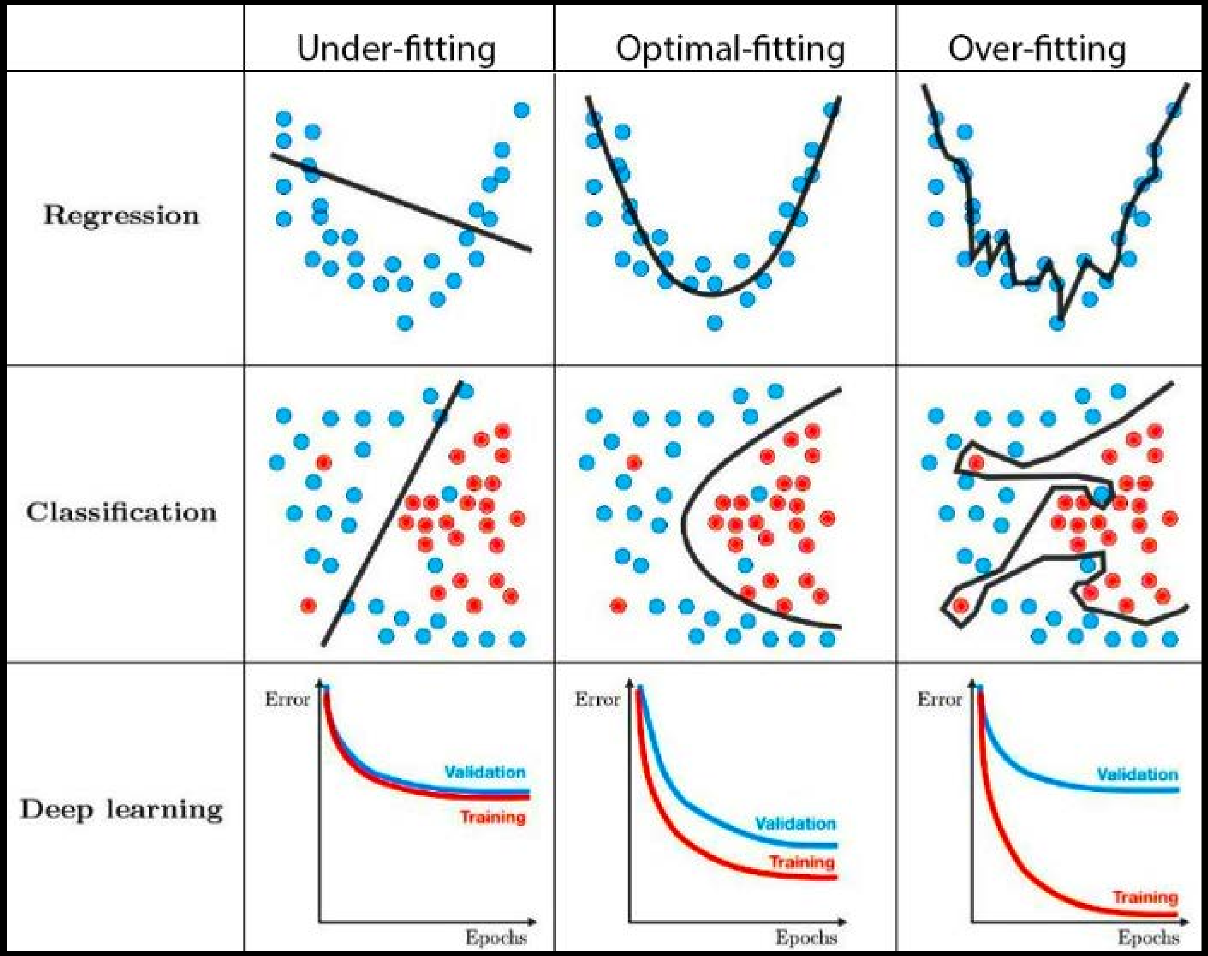
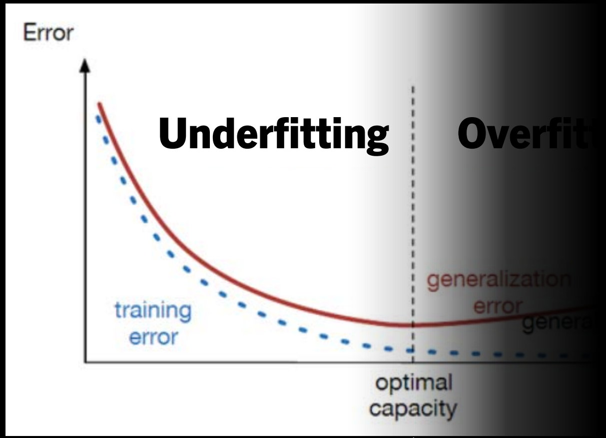


Image credit: https://i.pinimg.com/originals/72/e2/22/72e222c1542539754df1d914cb671bd7.png

Image credit: https://srdas.github.io/DLBook2/ImprovingModelGeneralization.html

How to deal with Underfitting/Overfitting?

Complexity & Size of data



- Complexity ↑ for...
  1. A model with more parameters
     - Degree of Freedom (DoF)
  2. A model whose parameters can take a wider range of values
     - When weights can take a wider range of values
  3. A model that takes more training iterations

1. Add more parameters
   - More hidden layers
   - More number of units per layer
3. Train longer

1. Add more data
   - ~Data augmentation
2. Weight decay, Dropout
3. Early stopping
- Regularization techniques (1-3)

+Different optimization algorithms, network architectures

More data never hurt!

- In theory
  - Fewer samples → more overfitting
    - # training data ↑ → generalization error ↓
  - Absent sufficient data, simpler models may be more difficult to beat.
- But in practice…
  - Costly, Time consuming
  - e.g.) Adding one MRI brain image of a dementia patient **~$200**

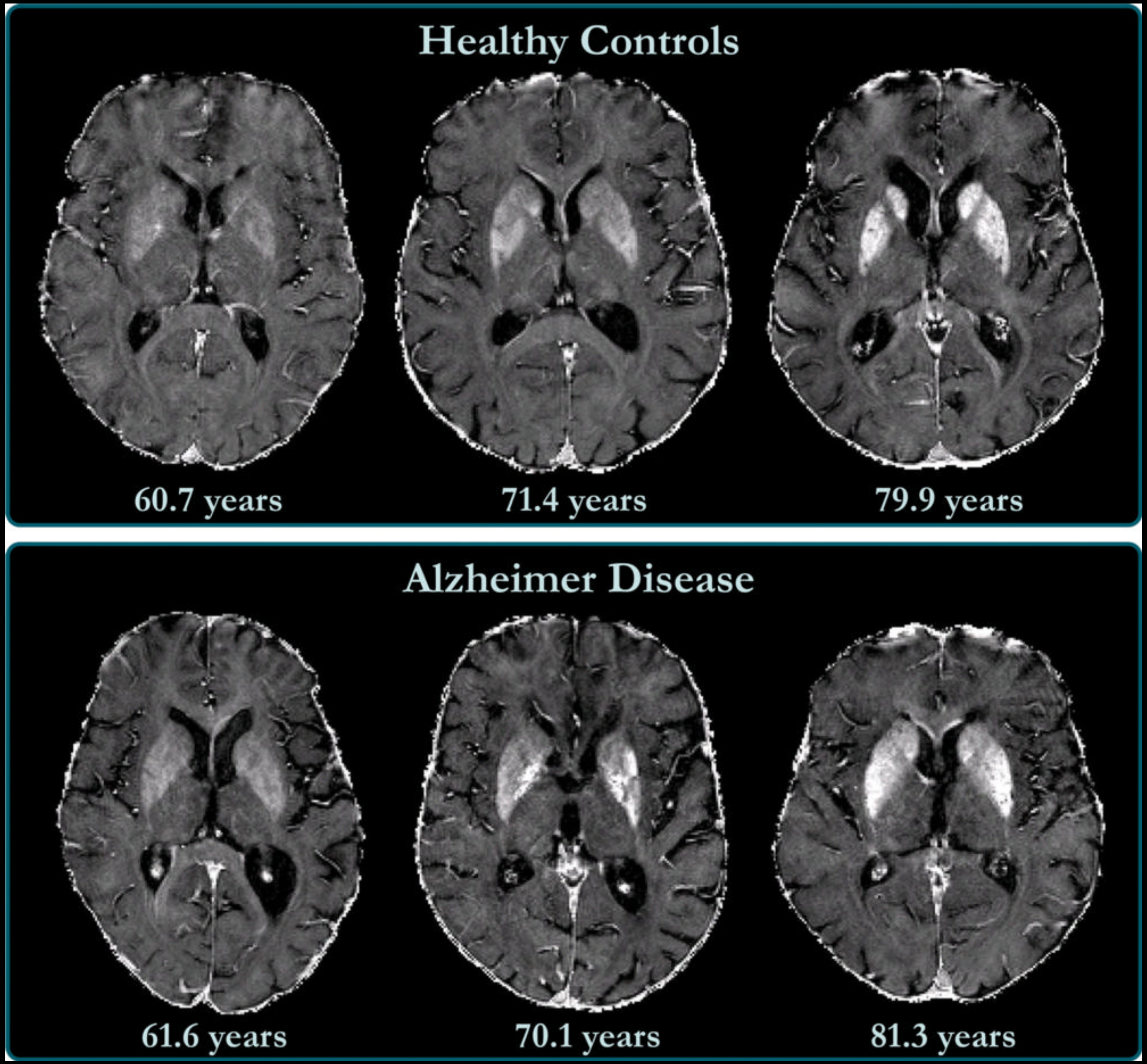


Image credit: https://www.appliedradiology.com/communities/MR-Community/mri-shows-brain-iron-accumulation-linked-to-cognitive-deterioration-in-alzheimer-s-patients

Data augmentation

- Techniques used to increase the amount of data
  1. Slightly modified copies of already existing data
  2. Newly created synthetic data from existing data.

- Acts as a regularizer and helps reduce overfitting

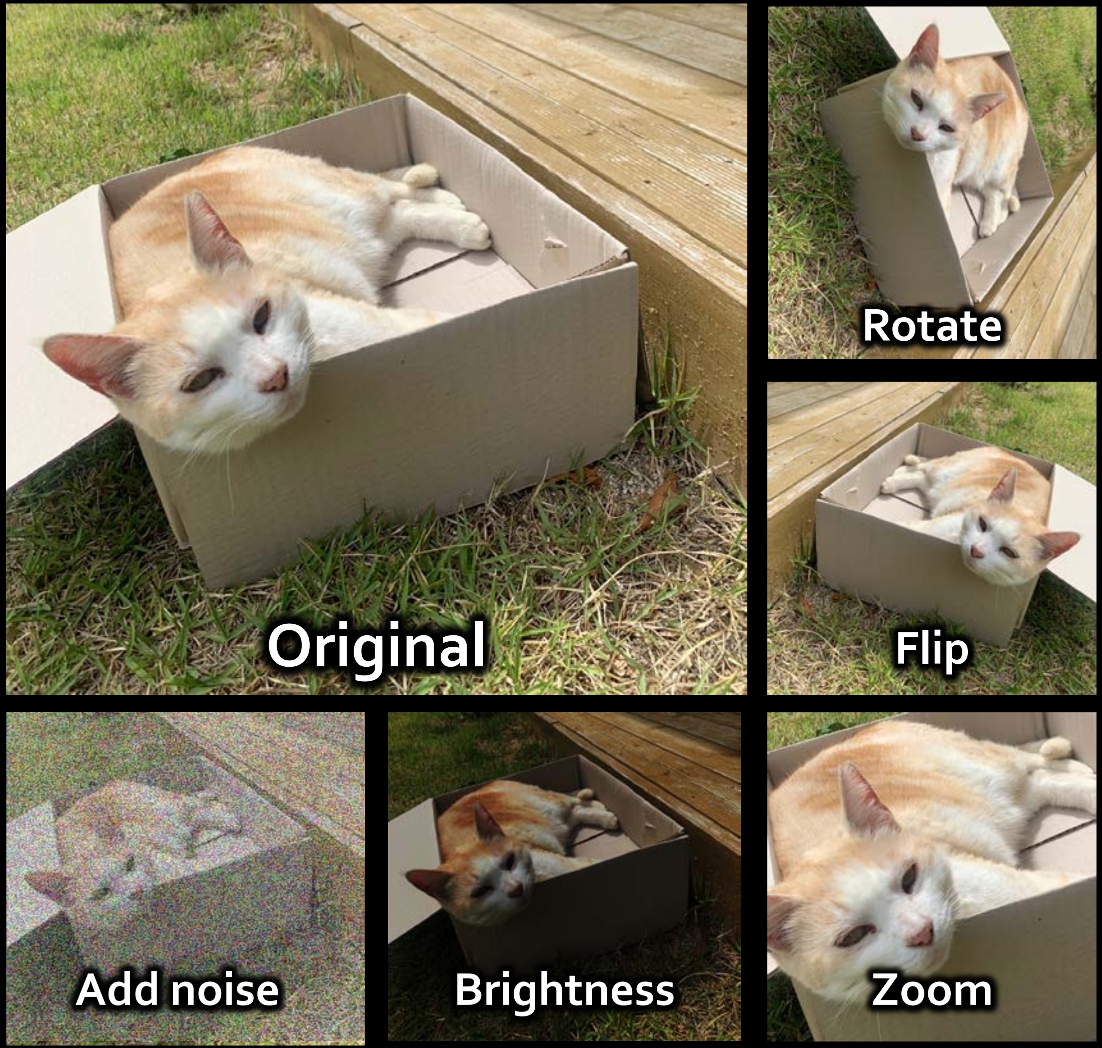


Image credit : My photo album ☺

Weight Decay

- Weight decay ($L2$ regularization)
  - The most widely-used technique for regularizing parametric machine learning models.
- Among all functions f, the function $f = 0$ (assigning the value 0 to all inputs) is in some sense the **simplest**
  - → Measure the complexity of a function by its distance from 0

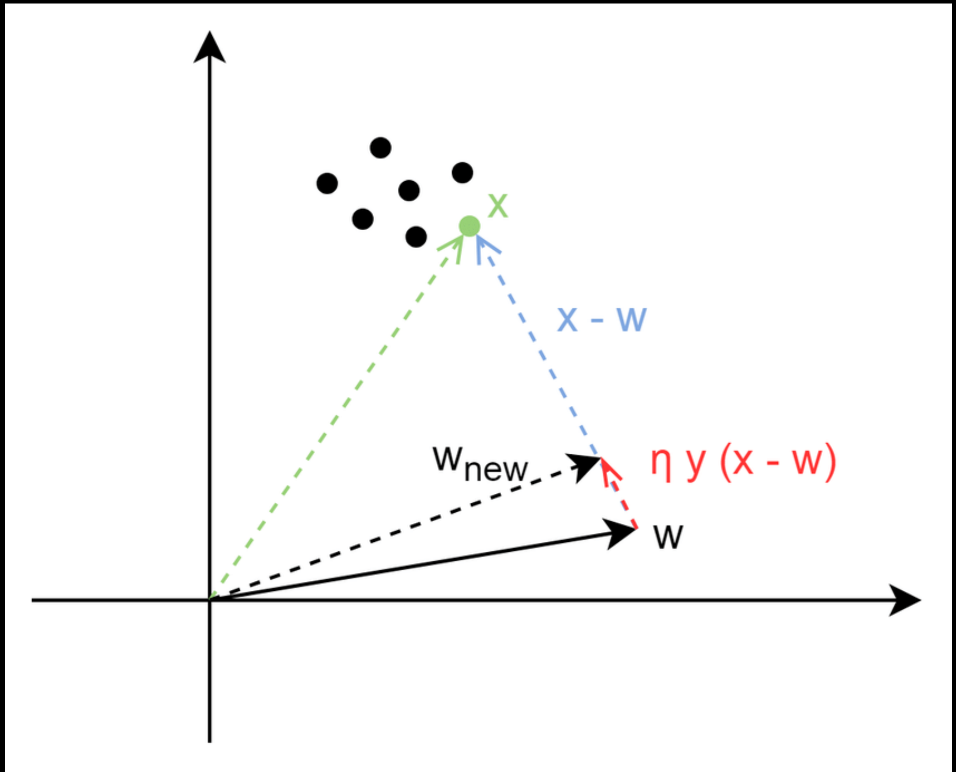


Image credit: Lagani et al. 2020

Weight Decay

- Norms (Section 2.3.10)
  - Norms : $||x||_p = (\Sigma_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$
  - L1 Norms $||x||_1 = \Sigma_{i=1}^{n} |x_i|$
  - L2 Norms $||x||_2 = \sqrt{\Sigma_{i=1}^{n} x_i^2}$
    - Frobenius norm (For a matrix X)
    - $||x||_F = \sqrt{\Sigma_{i=1}^{m} \Sigma_{j=1}^{n} x_{ij}^2}$
- Add its norm as a penalty term to the pro blem of minimizing the loss.
  - Minimizing the prediction loss on training sets → Minimizing both the prediction loss & **penal ty term**.

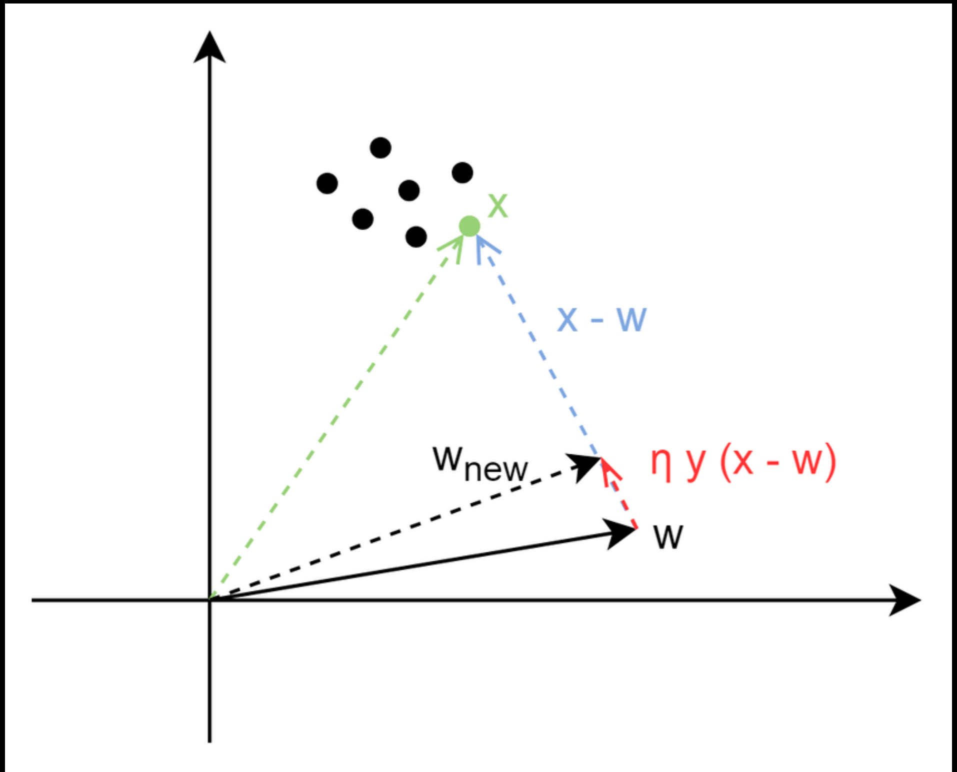


Image credit: Lagani et al. 2020

Weight Decay

WeiLeong's notebook

Gradient Descent

Therefore it is natural to define a **loss function** in linear regression, as

$$\ell^i = \frac{1}{2}\left(y^{(i)} - \hat{y}^{(i)}\right)^2 \quad (6)$$

for single sample set. For entire set,

$$L(\mathbf{w}, \mathbf{b}) = \frac{1}{n}\sum_{i=1}^{n} \ell^i = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}\left(\mathbf{y}^{(i)} - \mathbf{w}^\top \mathbf{x}^{(i)} - \mathbf{b}\right)^2 \quad (7)$$

and the goal is find

$$(\mathbf{w}^*, \mathbf{b}^*) = \operatorname*{argmin}_{\mathbf{w},\mathbf{b}} L(\mathbf{w}, \mathbf{b})$$

Besides that, we also need to control how far each update goes. Sometimes the gradient may be too extreme and the optimum value just passes off, so we need to adjust **learning rate** η, which controls the step for update process.

$$(\mathbf{w}, b) \leftarrow (\mathbf{w}, b) - \eta \frac{\partial L(\mathbf{w}, \mathbf{b})}{\partial(\mathbf{w}, \mathbf{b})}$$

$$w_j \leftarrow w_j - \eta \frac{1}{n}\sum_{i=1}^{n} \frac{\partial \ell^{(i)}(\mathbf{x}^{(\mathbf{i})}, y^{(i)}, \mathbf{w})}{\partial w_j} \quad (8)$$

Gradient Descent with L2 Regularization

λ: Regularization constant
(Degree of regularization)

$$L'(w) = L(w) + \frac{1}{2}\lambda||w||^2$$
$$= L(w) + \frac{1}{2}\lambda\Sigma_j w_j^2$$

$$w_j \leftarrow w_j - \eta\frac{\partial L'(w)}{\partial w_j}$$
$$= w_j - \eta\left(\frac{\partial L(w)}{\partial w_j} + \lambda w_j\right)$$
$$= (1 - \eta\lambda)w_j - \eta\frac{\partial L(w)}{\partial w_j}$$

Image credit: WeiLeong's notebook

Dropout

- Background from Bishop 1995
- Developed by Srivastava et al., 2014
  - Inject noise into each layer of the network before calculating the subsequent layer during training.
  - “Dropout” - We literally drop out some neurons during training.

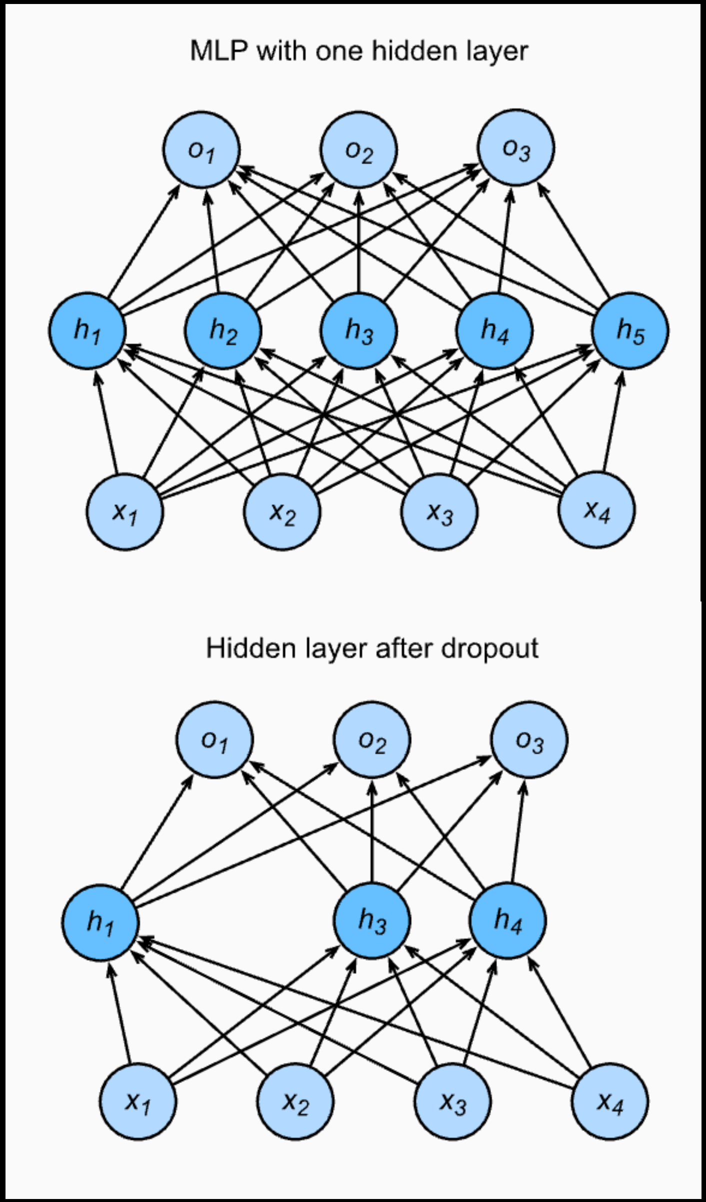


Image credit: https://d2l.ai/chapter_multilayer-perceptrons/dropout.html

Dropout

- During training, **randomly drop** hidden unit on some probability p from the neural network on each training iteration.
  - Activation h is replaced by h'
  - $h' = \begin{cases} 0 & \text{Dropout} \\ h/(1-p) & \text{Debiases each layer} \end{cases}$
    → Expectation does not change
- Can apply higher dropout probability to layers with more hidden units.
- For test / validation sets → turn off dropout
  - We don't want to add noise for evaluation

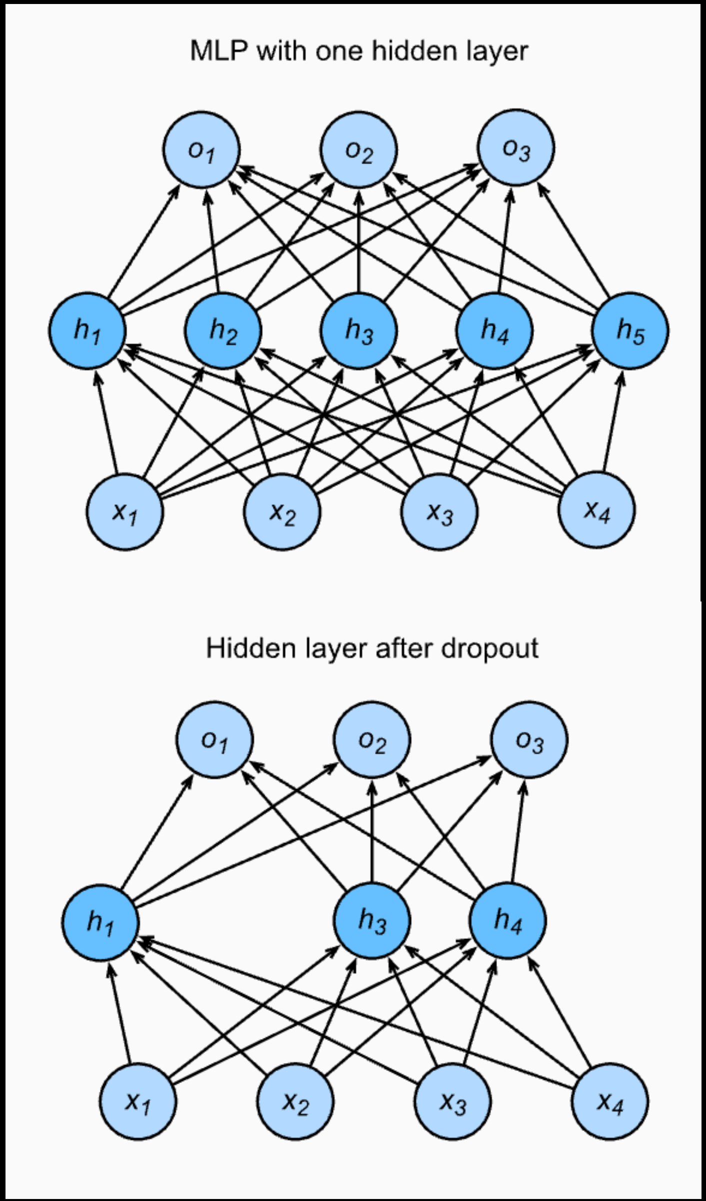


Image credit: https://d2l.ai/chapter_multilayer-perceptrons/dropout.html

Dropout

- Dropout breaks co-adaption among neurons
  - Some neurons are highly dependent on others

- Model weights are more motivated to spread out across many hidden units
  - Not depending too much on a small number of potentially spurious associations.
  - → Similar to the effect of applying L2 regularization.

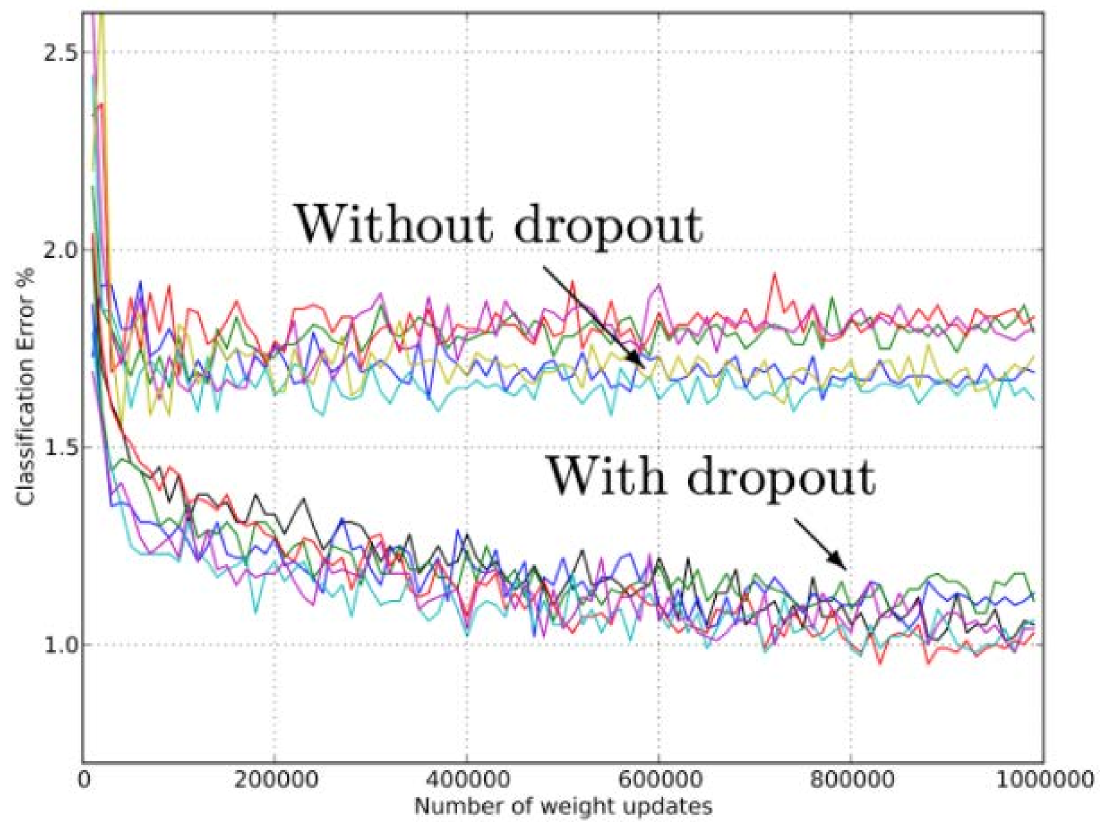


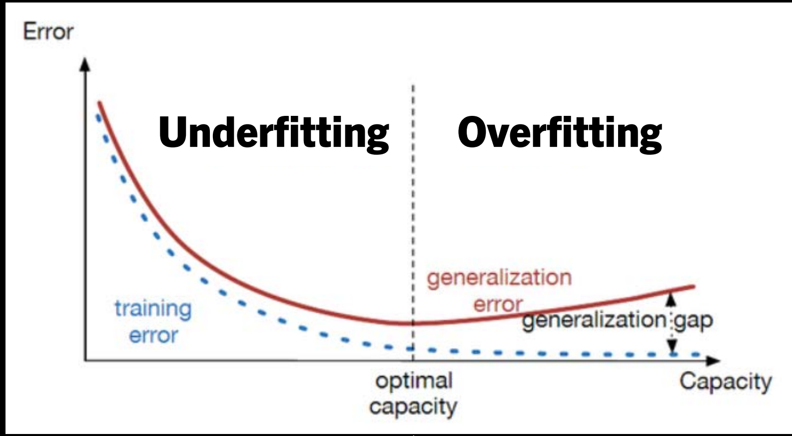
Figure 4: Test error for different architectures with and without dropout. The networks have 2 to 4 hidden layers each with 1024 to 2048 units.

Image credit: Srivastava et al. 2014

Image credit: https://srdas.github.io/DLBook2/ImprovingModelGeneralization.html

How to deal with Underfitting/Overfitting?

Complexity & Size of data



- Complexity ↑ for…
  1. A model with more parameters
     - Degree of Freedom (DoF)
  2. A model whose parameters can take a wider range of values
     - When weights can take a wider range of values
  3. A model that takes more training iterations

1. Add more parameters
   - More hidden layers
   - More number of units per layer
3. Train longer

1. Add more data
   - ~Data augmentation
2. Weight decay, Dropout
3. Early stopping

- **Regularization techniques (1-3)**

+Different optimization algorithms, network architectures

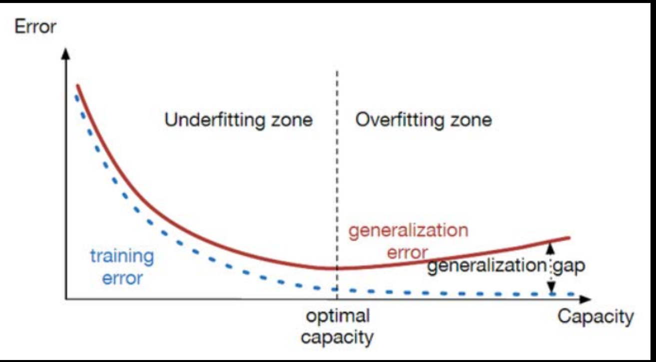
Summary

Introduction

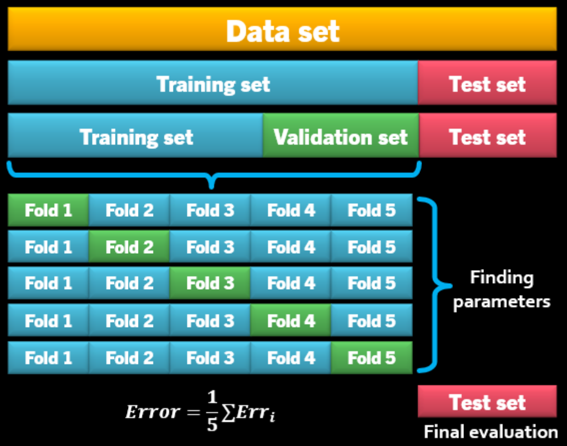
DB - Result: Not found

ML - Result: Spiral

Under/ Overfitting



Data Sets



How to Kill Overfitting

Complexity ↓ & Size of data ↑