UNIVERSIDADE CATÓLICA PORTUGUESA

BRAGA

Deep Learning

Session 9

Regularization

Applied Data Science

2024/2025

Regularization

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- What is it?
  - A technique that constrains our optimization problem to discourage complex models by limiting the model's capacity, preventing it from fitting the noise in the training data.

> "any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error."
>
> - Ch. 5.2 of Goodfellow book on Deep Learning

- Why do we need it?
  - To improve the generalization of our model on unseen data by balancing model capacity, ensuring it captures the underlying patterns without overfitting.

Preventing Overfitting

- **Get more data:** Increasing the amount of data helps the model generalize better.

- **Use a model with the right capacity:**
  - Too much capacity: The model becomes overly complex and learns noise (overfitting).
  - Too little capacity: The model fails to capture important patterns (underfitting).

- **Early stopping:** Stop training when the model starts to overfit, based on validation performance.

- **Parameter Norm Penalty:** Add a penalty to the loss function for large weights, discouraging complexity.

Preventing Overfitting

- **Droput:** Randomly drop neurons during training to prevent co-adaptation and reduce overfitting.

- **Batch Normalization:** Normalize activations in each layer, stabilizing learning and acting as a regularizer.

- **Ensemble methods:** Combine predictions from multiple models (like bagging or boosting) to reduce variance and improve generalization.

Get More Data

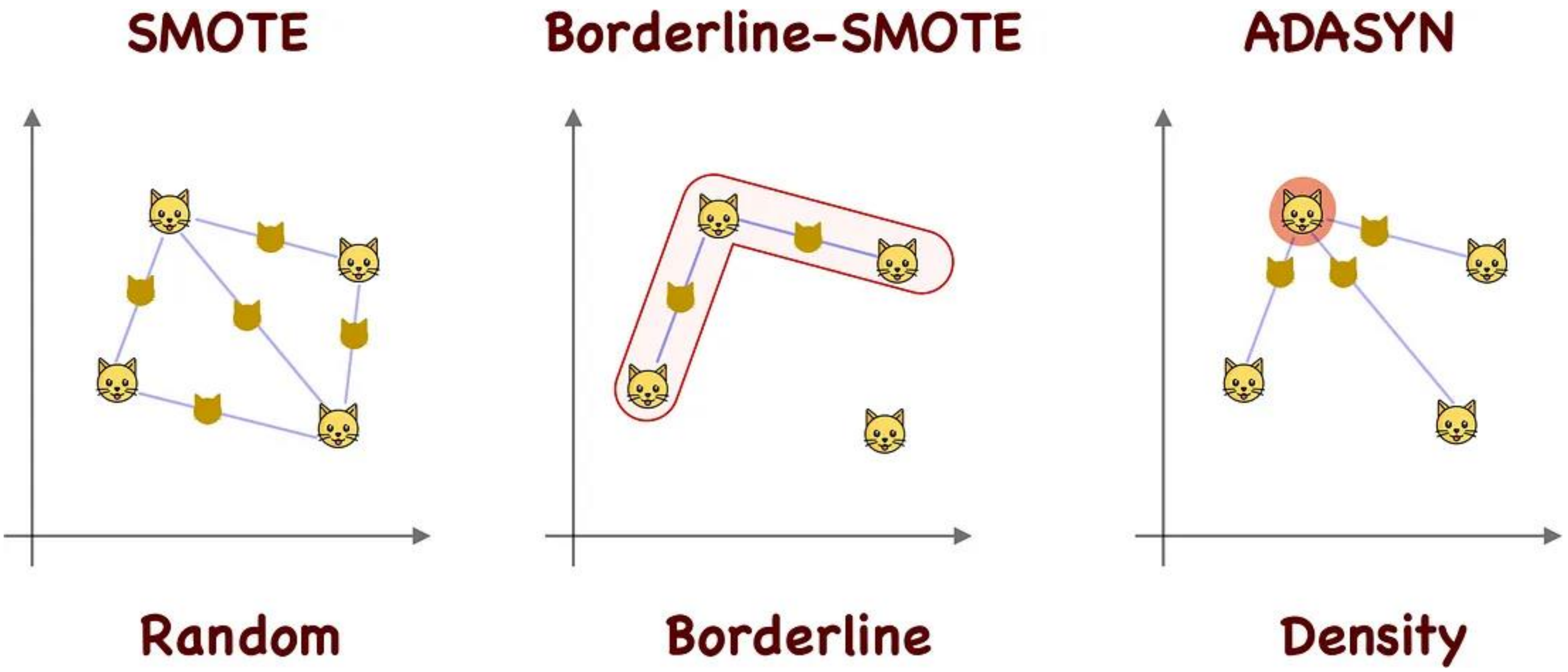
UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- If possible, **gathering more data** is always the best solution!
  - However, data collection can be expensive or time-consuming.
  - More data may require more computational resources.

- **Data Augmentation**:
  - Data augmentation involves applying various transformations to your existing dataset to artificially increase its size and diversity.

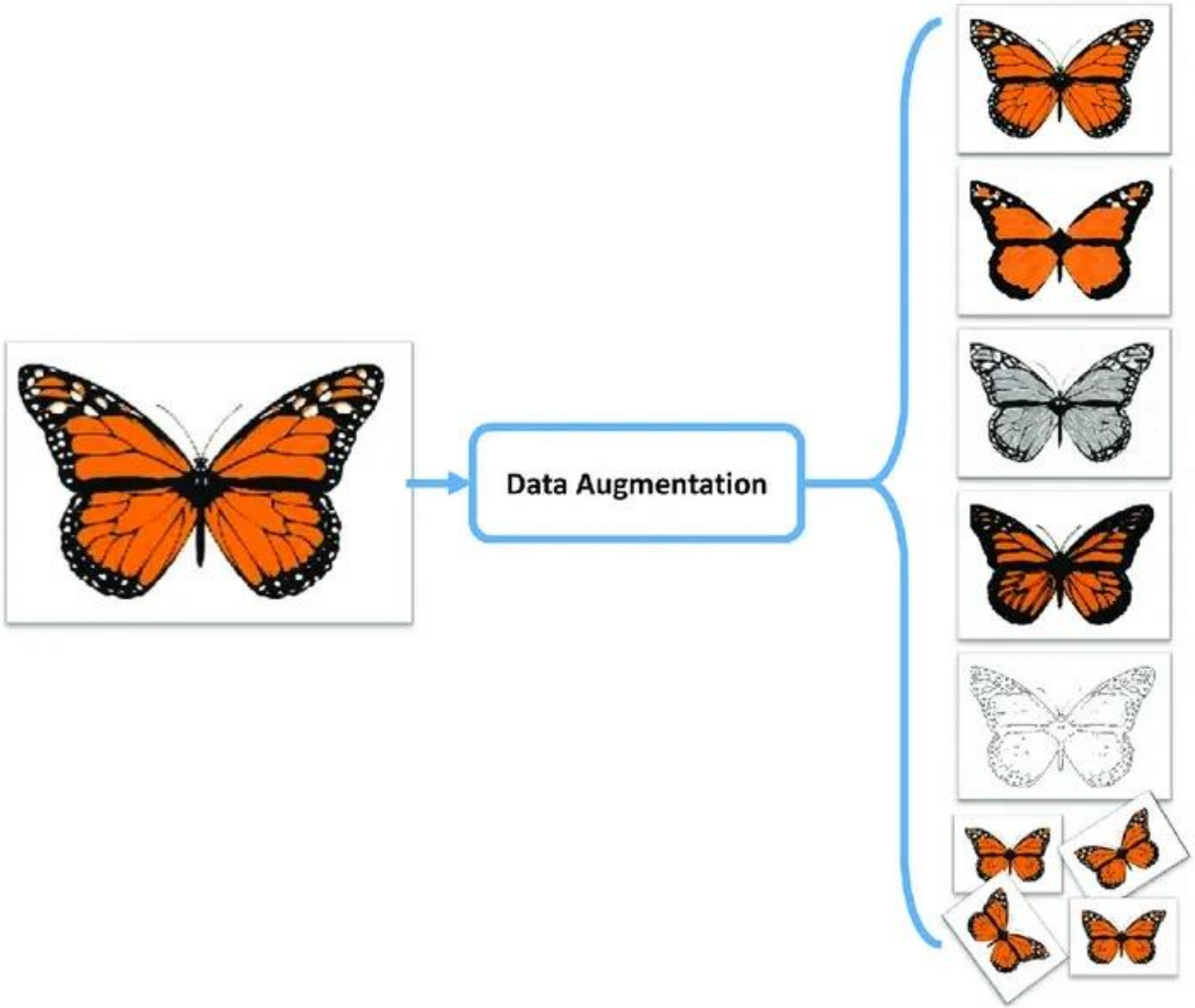
- **Leverage Pre-trained Models:**
  - Transfer learning from models trained on larger datasets can help when gathering more data is difficult.

Data Augmentation



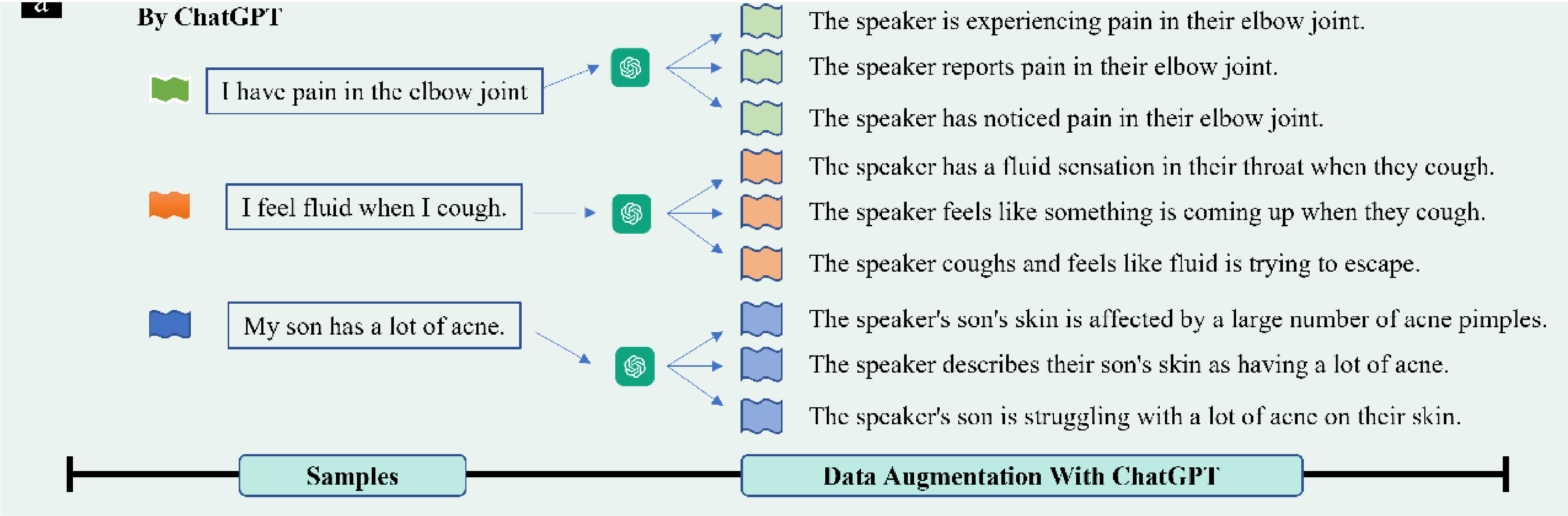
https://towardsdatascience.com/smote-synthetic-data-augmentation-for-tabular-data-1ce28090debc

Data Augmentation



https://www.labellerr.com/blog/what-is-data-augmentation-techniques-examples-benefits/

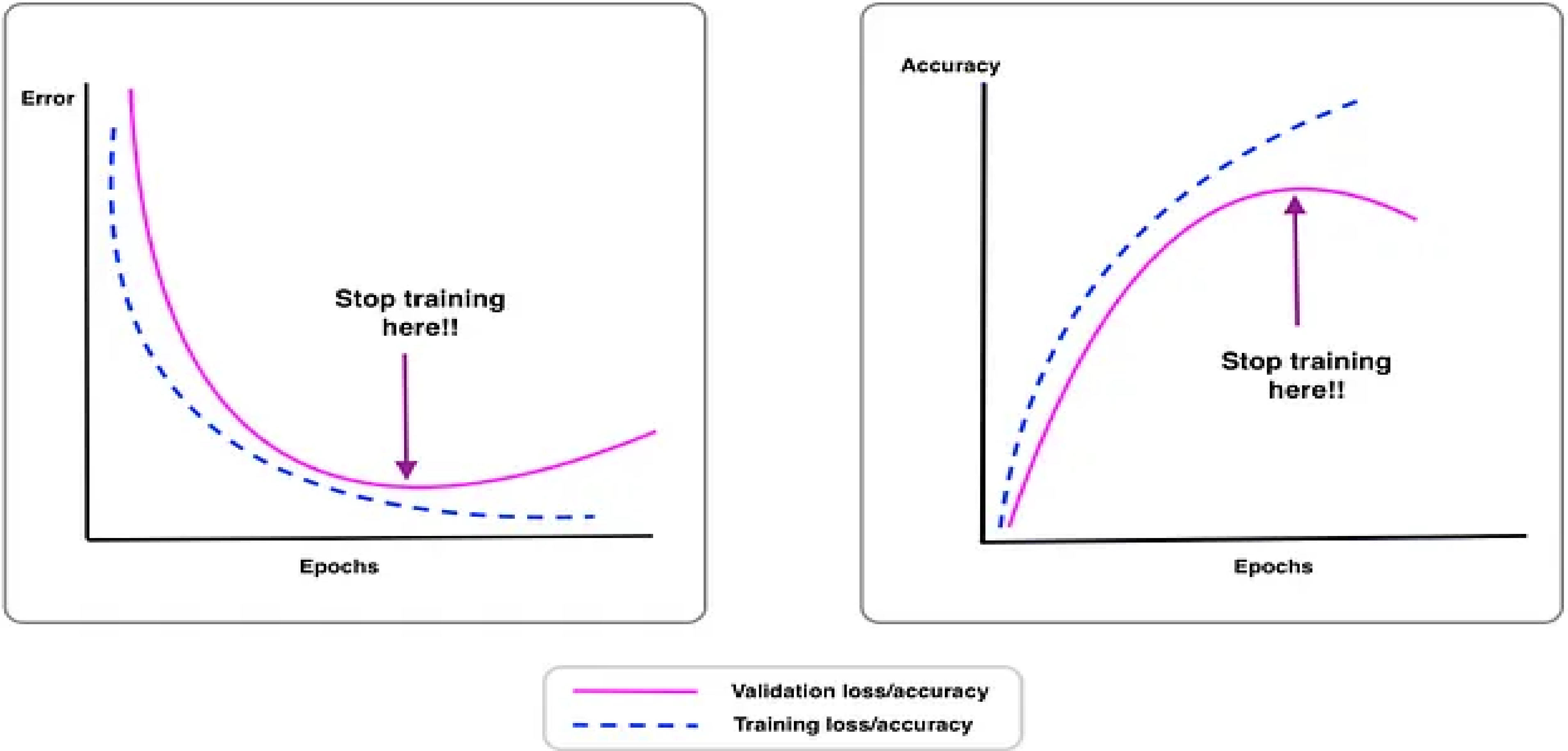
Data Augmentation



https://www.catalyzex.com/paper/auggpt-leveraging-chatgpt-for-text-data

Early Stopping

- Stop training before we have a chance to overfit



Parameter Norm Penalty

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- **Key Insight:** Large weights are often a sign of overfitting.
- **Solution:** Apply a penalty to the size of the weights to reduce overfitting.
- **Analogy:** It's like tightening a belt on oversized pants.

Parameter Norm Penalty

- **Idea:** Penalize large weights in the objective function
- e.g., objective is to minimize sum of squared errors over training examples
- **L2 norm (Ridge):** penalize squated weight values

$$Error = \boxed{\sum_{i=1}^{n} (y^{(i)} - \widehat{y}^{(i)})^2} + \alpha \boxed{\sum_{j=1}^{m} w_j^2}$$

- **L1 norm (Lasso):** penalize absolute weight values

$$Error = \boxed{\sum_{i=1}^{n} (y^{(i)} - \widehat{y}^{(i)})^2} + \alpha \boxed{\sum_{j=1}^{m} |w_j|}$$

- Note: only weights are penalized, not bias terms

Parameter Norm Penalty

- e.g., objective is to minimize sum of squared errors over training examples.

- **L2 norm (Ridge):** penalize squated weight values

$$Error = \sum_{i=1}^{n}(y^{(i)} - \widehat{y}^{(i)})^2 + \boxed{\alpha}\sum_{j=1}^{m} w_j^2$$

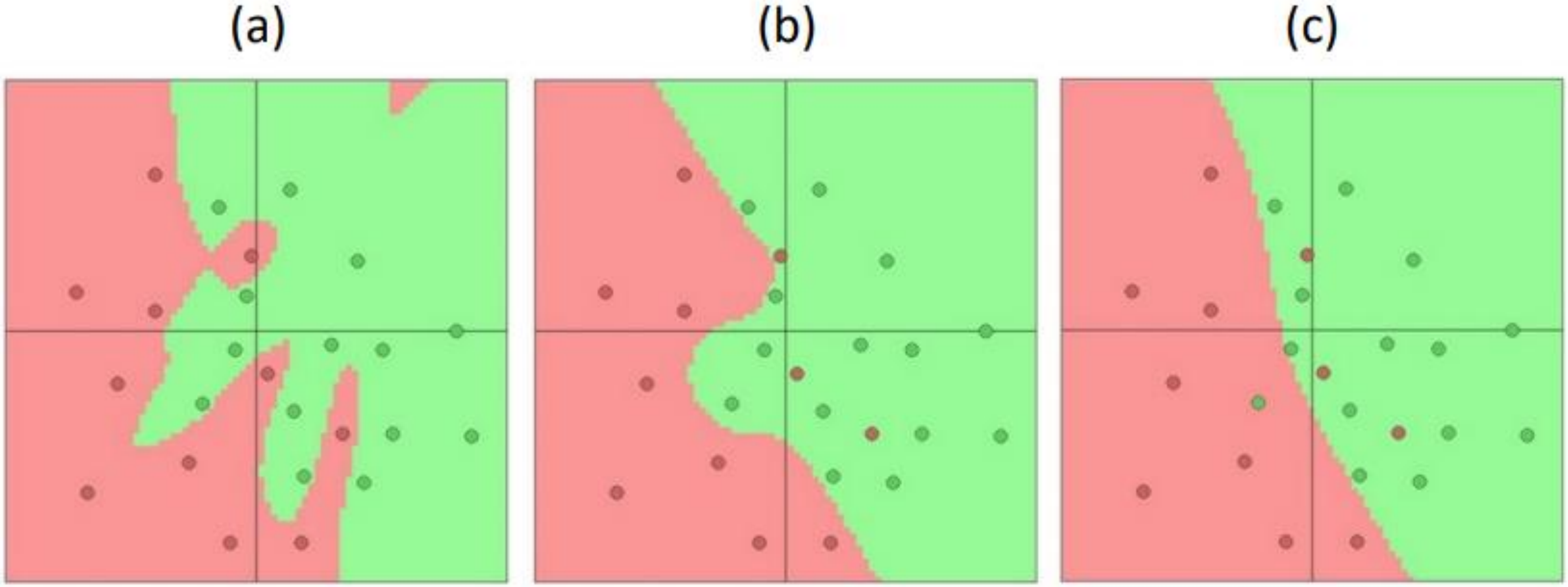
- **L1 norm (Lasso):** penalize absolute weight values

$$Error = \sum_{i=1}^{n}(y^{(i)} - \widehat{y}^{(i)})^2 + \boxed{\alpha}\sum_{j=1}^{m} |w_j|$$

- Hyperparameter determines relative contribution of norm penalty term.

Parameter Norm Penalty: How to Set Alpha?

- Shown is the same neural network with different levels of regularization. Which model has the largest value for alpha (i.e., largest norm penalty contribution)?

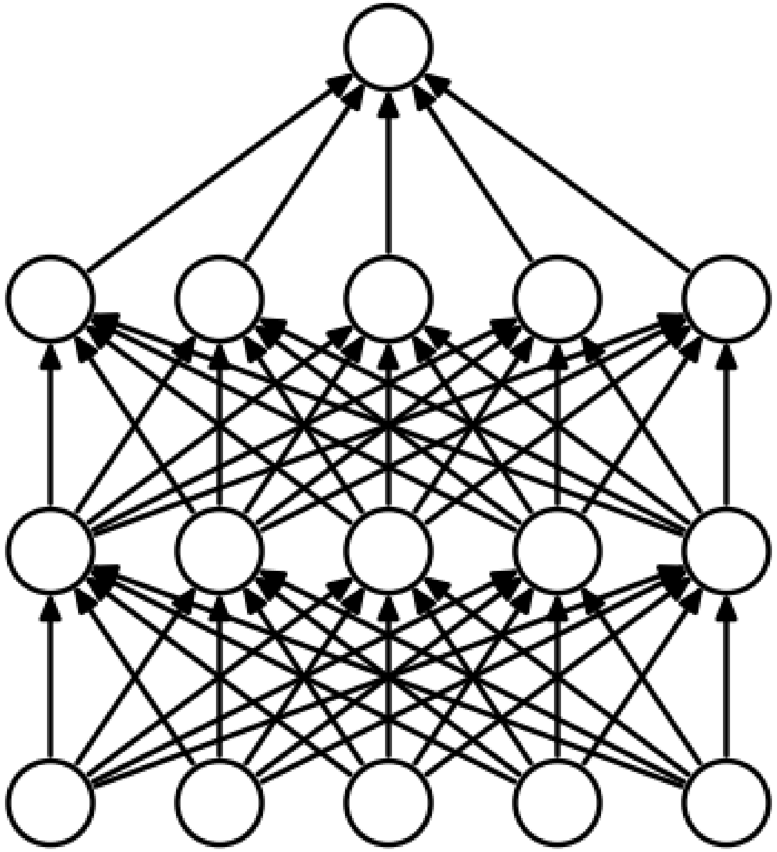


Dropout

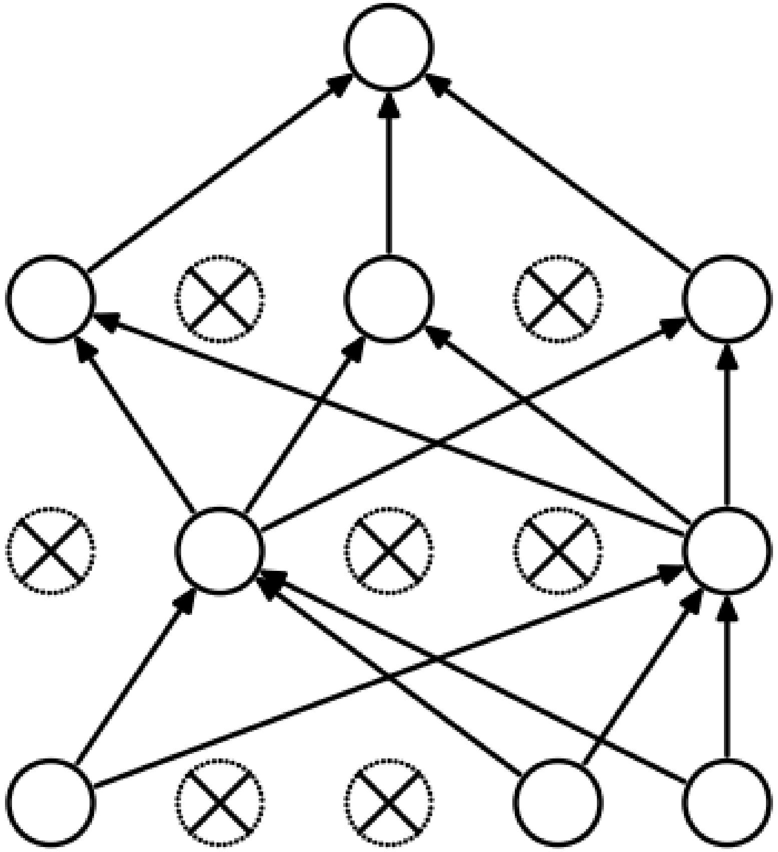
UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Dropout is a stochastic regularization method.
- In each forward pass, randomly set some neurons to zero for one pass.
- Forces the network to not rely on any single node.
- The probability of dropping a neuron is defined by an hyperparameter; 0.5 is commonly used.
- Note: During inference dropout is not applied!

Dropout

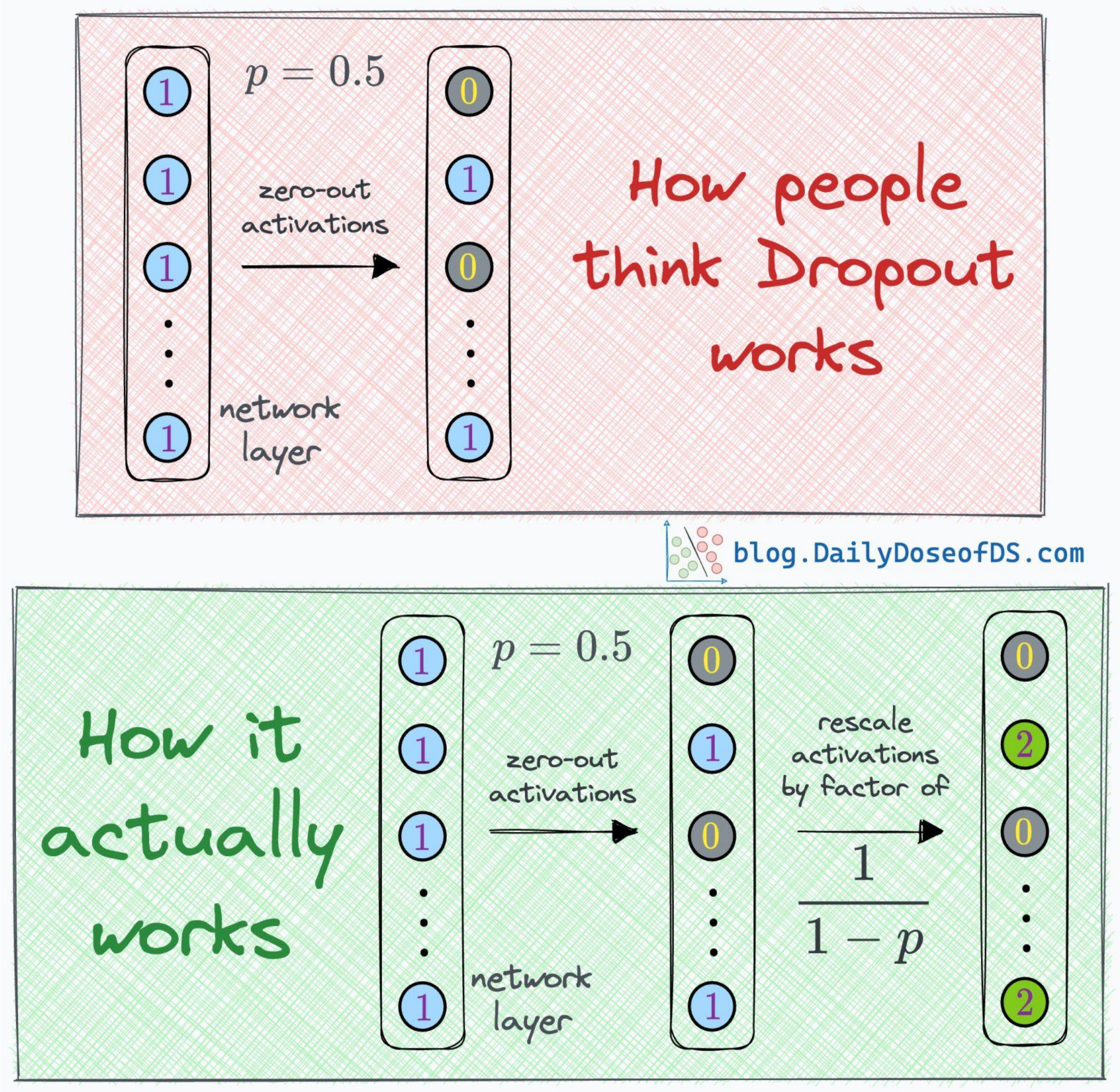


(a) Standard Neural Net



(b) After applying dropout.

Dropout



$p = 0.5$
zero-out activations
network layer
How people think Dropout works
blog.DailyDoseofDS.com
How it actually works
rescale activations by factor of
$\frac{1}{1-p}$

Batch Normalization

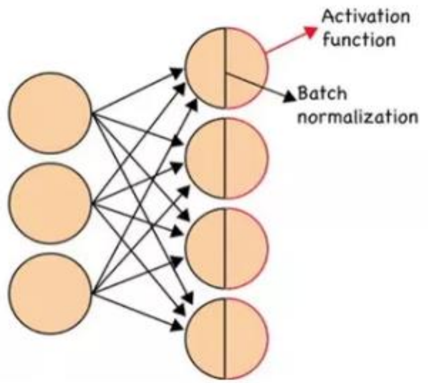
- **Motivation:** Features on different scales can cause learning to be slower and poor performance

- We normalize all training data so that it resembles a normal distribution (that means, zero mean and a unitary variance)

- In the intermediate layers the distribution of the activations is constantly changing during training
  - This slows down the training process because each layer must learn to adapt themselves to a new distribution in every training step.
  - Batch normalization is a method we can use to normalize the inputs of each layer, in order to fight the internal covariate shift problem.

Batch Normalization

- During training time, a batch normalization layer does the following:
  - Calculate the mean and variance of the layers input
  - Normalize the layer inputs using the previously calculated batch statistics
  - Scale and shift in order to obtain the output of the layer
  - γ and β are learned during training along with the original parameters of the network.
- During inference, the mean and the variance are fixed. They are estimated using the previously calculated means and variances of each training batch.



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1 \ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

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

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

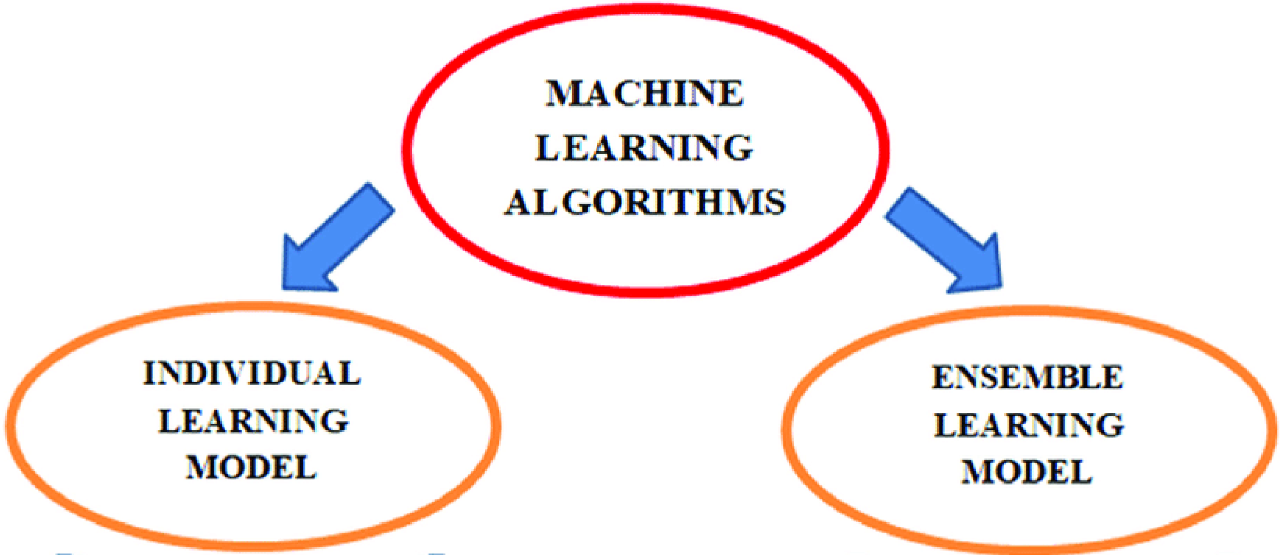
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

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

Ensemble Methods

- **Idea:** Use the wisdom of the crowd

- **Why Choose Ensemble vs One Predictor?**
  - Reduces probability for making a wrong prediction



Ensemble Methods

- Suppose:
  - n classifiers for binary classification task
  - Each classifier has same error rate ε
  - Classifiers are independent (not true in practice!)
  - Probability mass function indicates the probability of error from an ensemble:

Number of classifiers

Classifier error rate

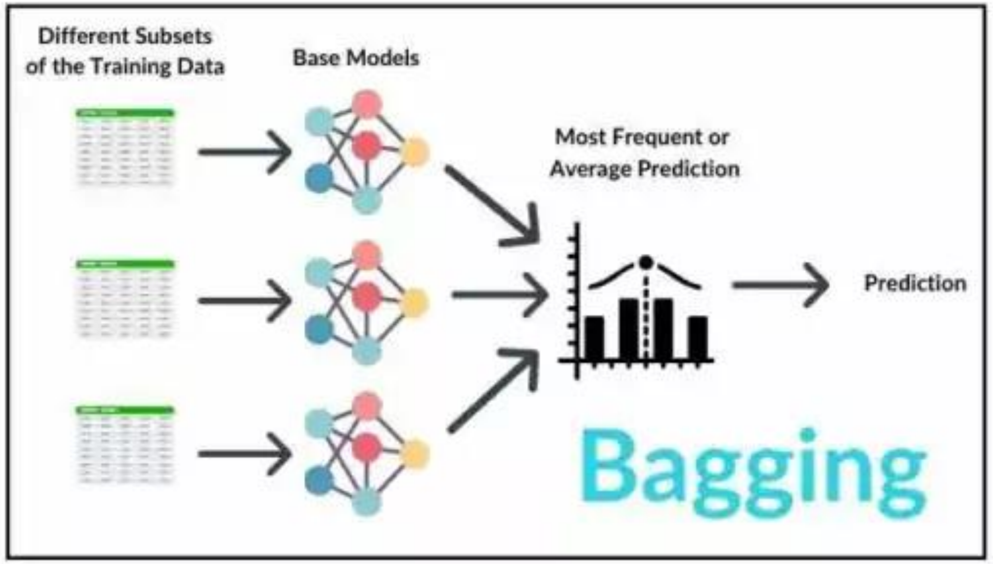
$$P(y \geq k) = \sum_{k}^{\boxed{n}} \boxed{\binom{n}{k}} \varepsilon^k (1 - \boxed{\varepsilon})^{n-k} = \boxed{\varepsilon_{ensemble}}$$

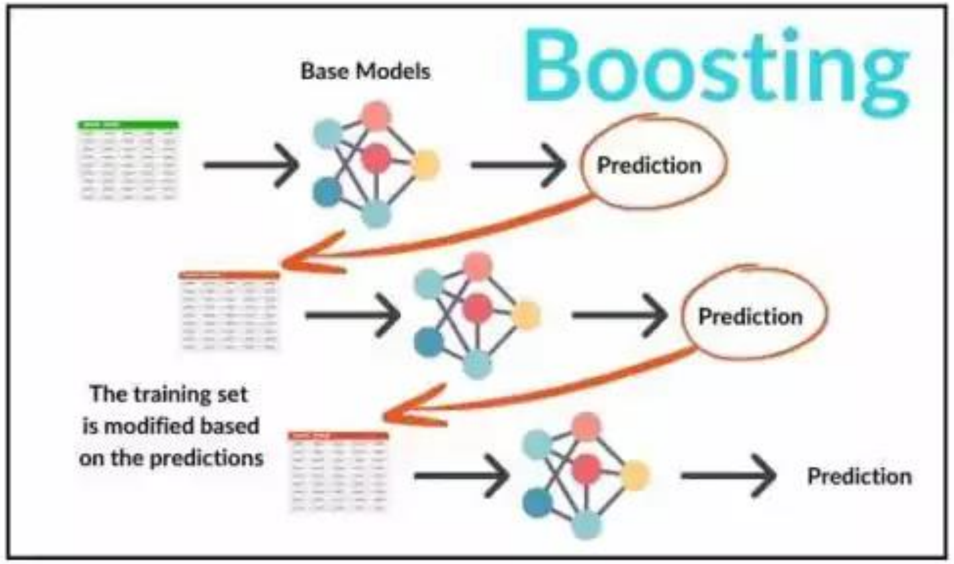
Error probability

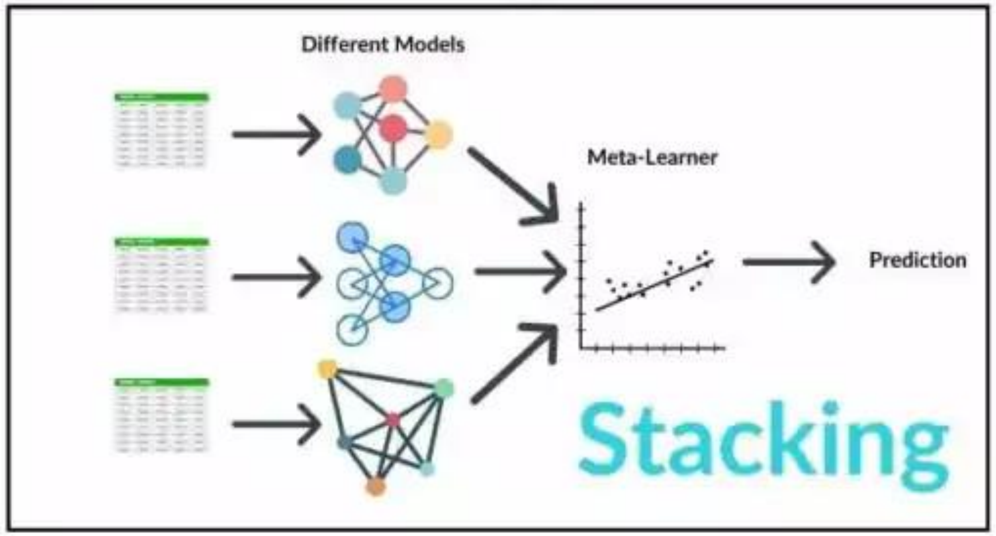
ways to choose *k* subsets from set of size *n*

- e.g., n = 11, ε = 0.25; k = 6: probability of error is ~0.034 which is much lower than probability of error from a single algorithm (0.25)

How to Produce an Ensemble?







Ensembles for Neural Networks

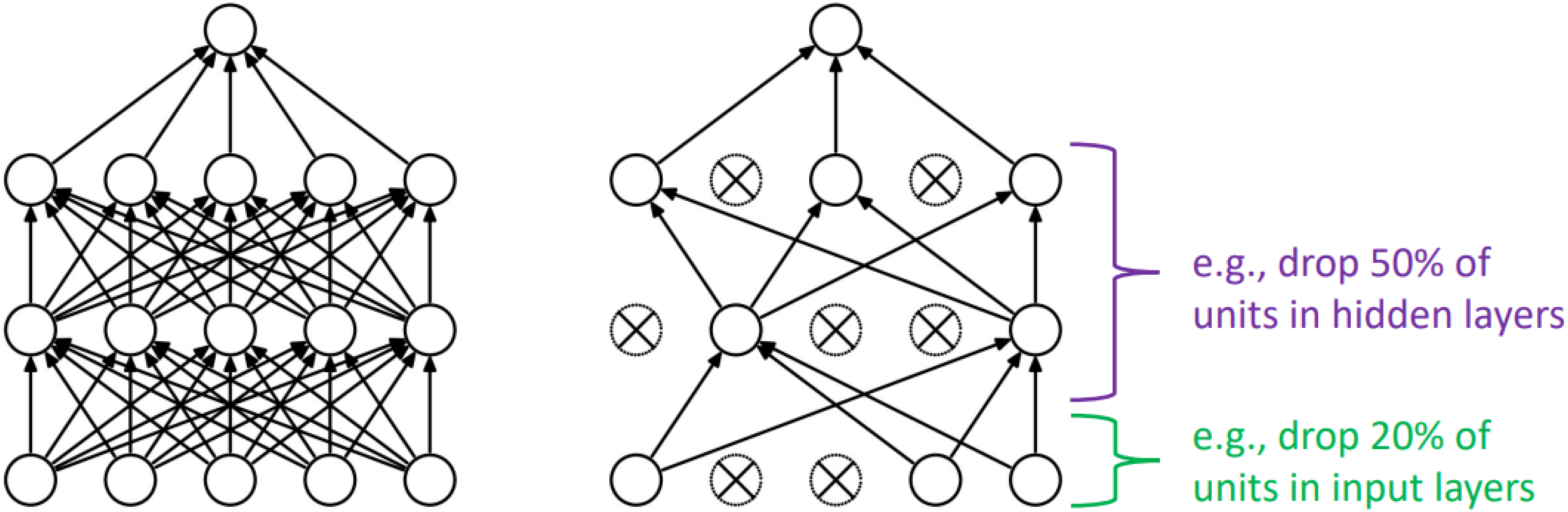
UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Why could ensembling neural networks be difficult?
  - **Hyperparameter Tuning:** Finding optimal hyperparameters for each model is time-consuming.
  - **High Resource Usage:** Ensembles require significant memory and computational power.
  - **Extended Training Time:** Training multiple models increases total time.
  - **Increased Complexity:** Managing multiple architectures and data pipelines adds complexity.
  - **Diminishing Returns:** Additional models may only slightly improve performance.
  - **Deployment Issues:** Ensemble models increase inference time and hardware needs.

Ensembles for Neural Networks

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- **Idea:** approximate bagging with dropout during training so different sub-models in the network are trained with different training data



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