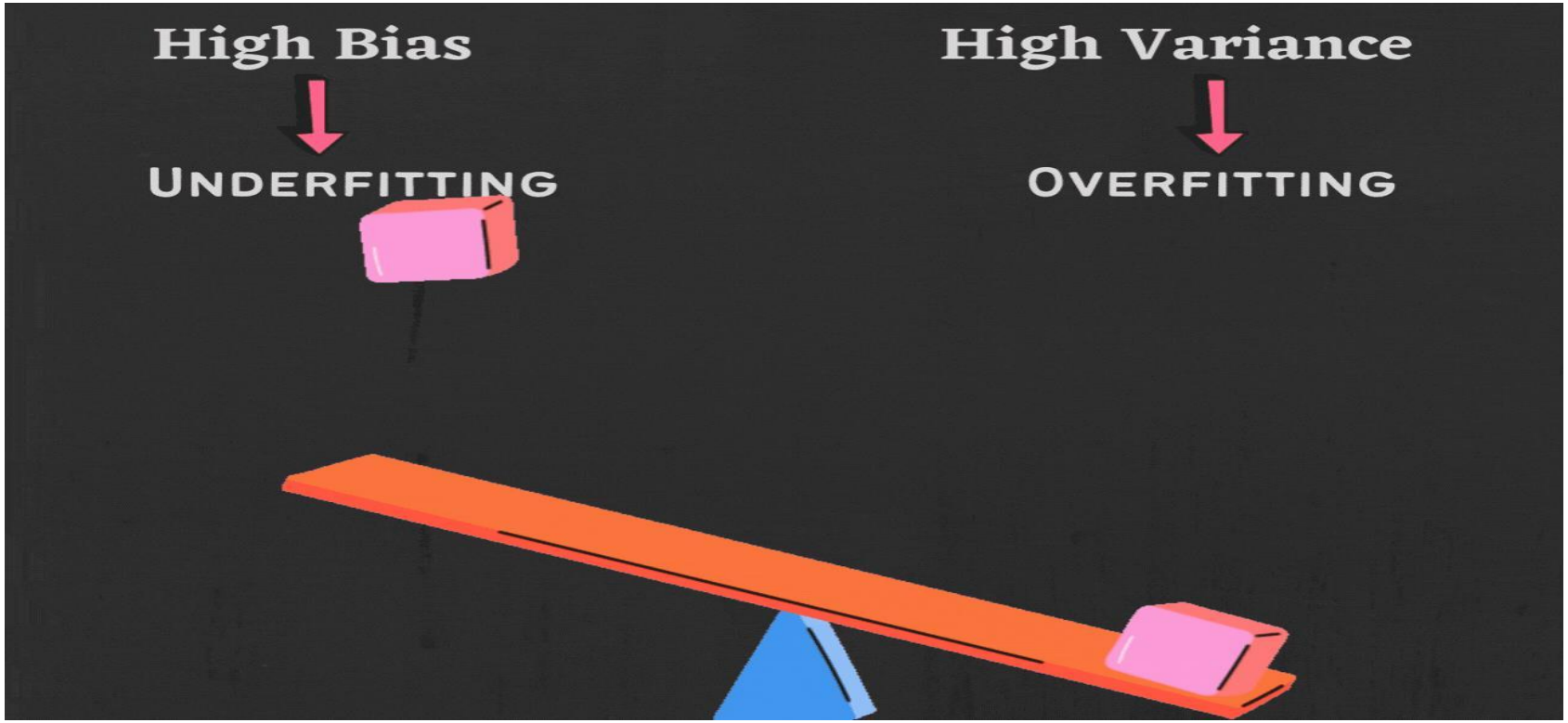
Regularization for Deep Learning



Regularization for Deep Learning

- One of the most common problems data science professionals face is avoiding overfitting.
- A central problem in machine learning is how to make an algorithm that will perform well not just on the training data, but also on new inputs.
- The ability to perform well on previously unobserved inputs is called generalization.
- Regularization is a technique used in machine learning and deep learning to prevent overfitting and improve the generalization performance of a model.

Concept	Definition	Example	Solution
Generalization	Performs well on new data	Model correctly classifies a new dog	Use diverse training data
Overfitting	Learns too much, memorizes training data	Model only recognizes trained dog breeds	Regularization, more data, dropout
Underfitting	Learns too little, fails to capture patterns	Model randomly guesses dog/cat	Increase model size, train longer
Regularization	Prevents overfitting by simplifying learning	Reduces focus on unnecessary details	L1/L2 regularization, dropout

The Bias-Variance Tradeoff: Overfitting and Underfitting

What is Bais?

- Bias is a systematic (built-in) error that makes all measurements wrong by a certain amount.
- The term bias is the difference between the average predicted value of the model and the actual value of our model which we are trying to predict.
- A model with high bias pays very little attention to the training data and oversimplifies the model

What is Variance?

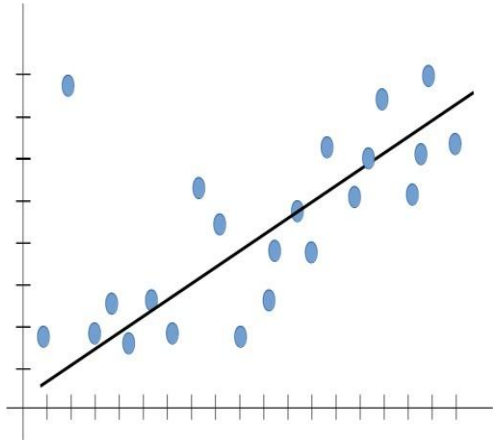
- Variance is the variability of model prediction for a given data point or a value that tells us the spread of our data.
- Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before

Aspect	High Bias (Underfitting)	High Variance (Overfitting)
Model Complexity	Too simple	Too complex
Error Type	Systematic error due to incorrect assumptions	Sensitivity to training data
Training Accuracy	Low	High
Test Accuracy	Low	Low
Example	Linear model for dog/cat classification	Deep model memorizing noise in images
Solution	Use a more complex model	Regularization, more data

The Bias-Variance Tradeoff: Overfitting and Underfitting Cont'd

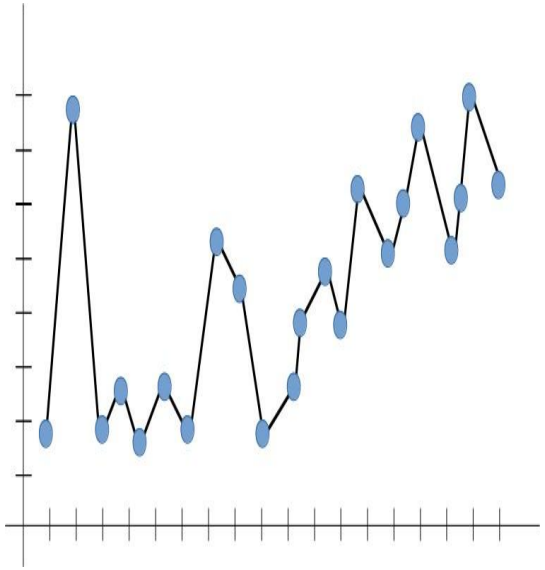
Underfitting

- The model contains high bias is simpler than it should be and hence tends to underfit.
- In other words, the model fails to learn on the given data and acquire the intricate pattern of the dataset.
- It Means a biased model will not fit on the Training Dataset properly and hence will have low training accuracy (or high training loss).



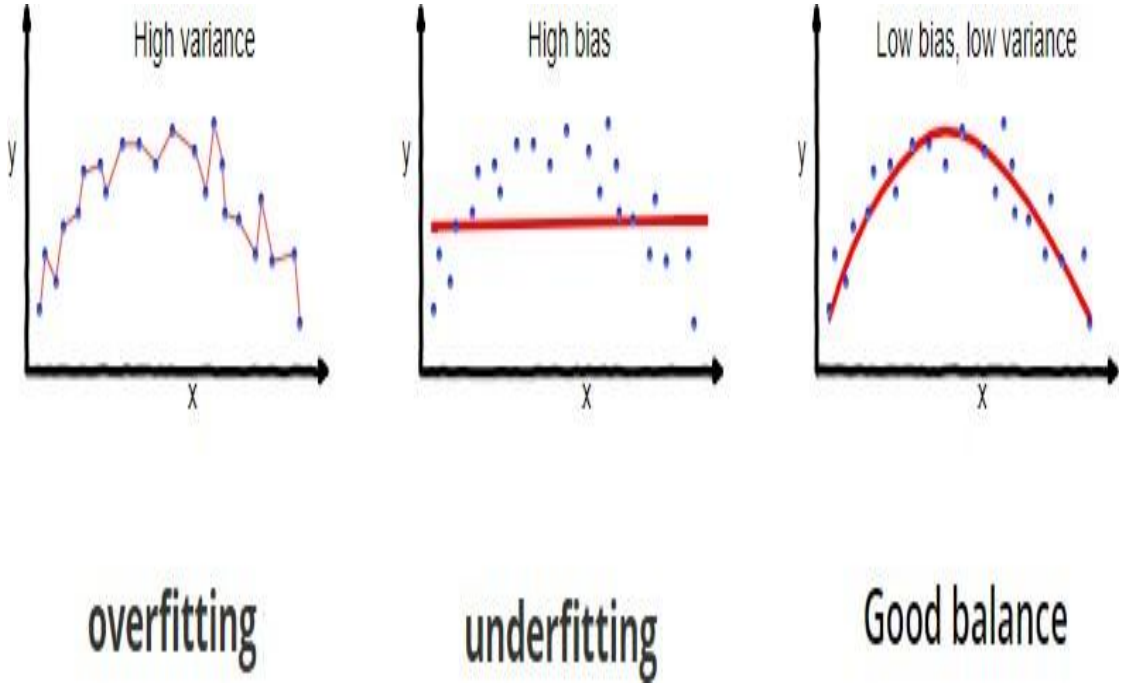
Overfitting

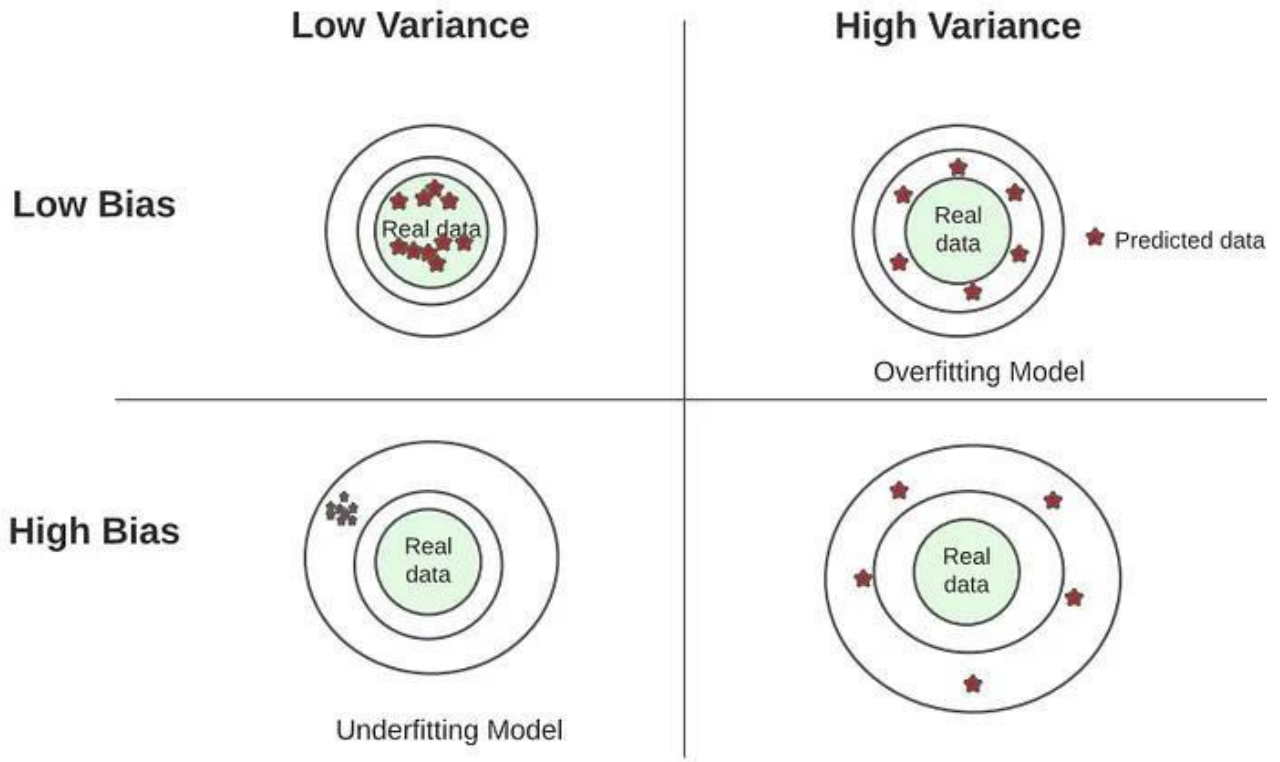
- A model with high Variance will have a tendency to be overly complex. This causes the overfitting of the model.
- Suppose the model with high Variance will have very high training accuracy (or very low training loss), but it will have a low testing accuracy (or a low testing loss).
- A model with a high variance means the model is overly complex and tries to fit a much more complex curve to relatively simpler data.



Concept	Cause	Result
High Bias (Underfitting)	Too simple model	Poor performance on both training and test data
High Variance (Overfitting)	Too complex model	Good performance on training data, poor on test data
Low Bias & Low Variance (Good Generalization)	Balanced model	Good performance on both training and test data

The Bias-Variance Tradeoff: Overfitting and Underfitting Cont'd





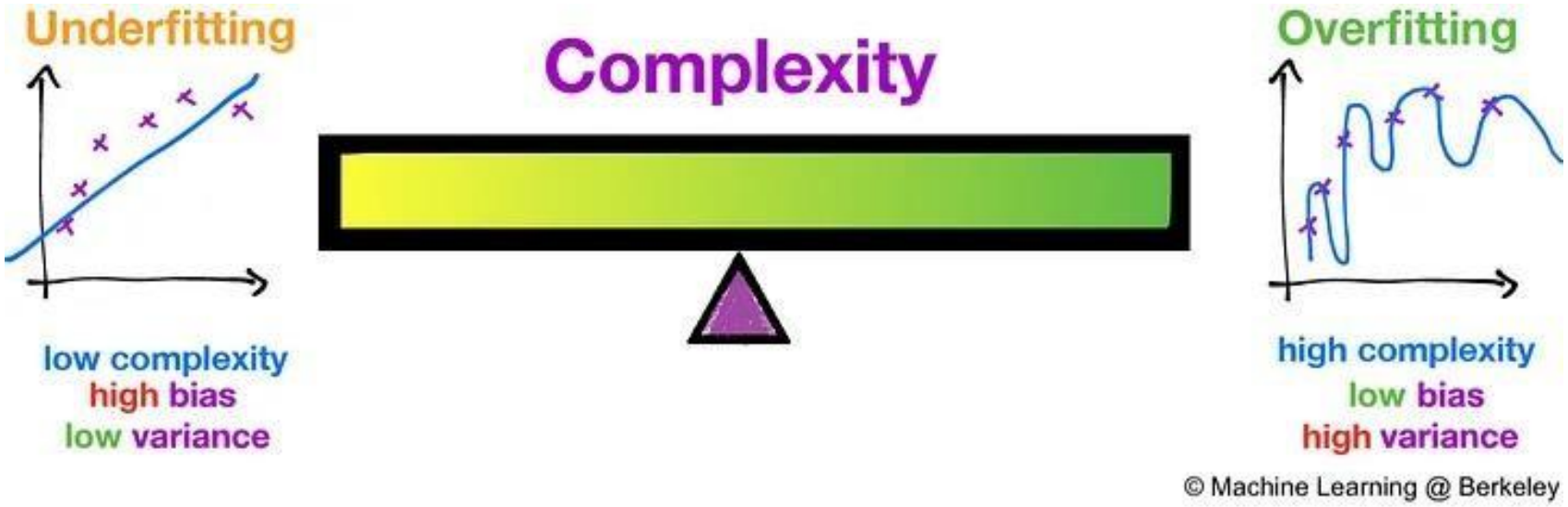
The Bias-Variance Tradeoff: Overfitting and Underfitting Cont'd

Low Bias and High Variance

- Low Training Error → Low Bias.
- If the training accuracy is high and test accuracy is low compared to the training accuracy then the model has overfitted.

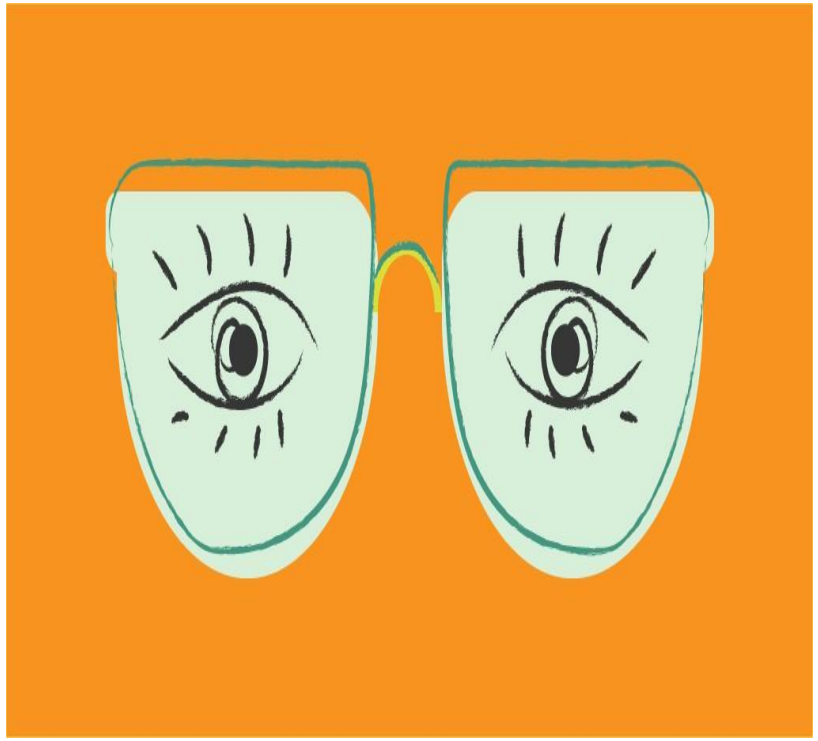
Low Bias and Low Variance(Best Case)

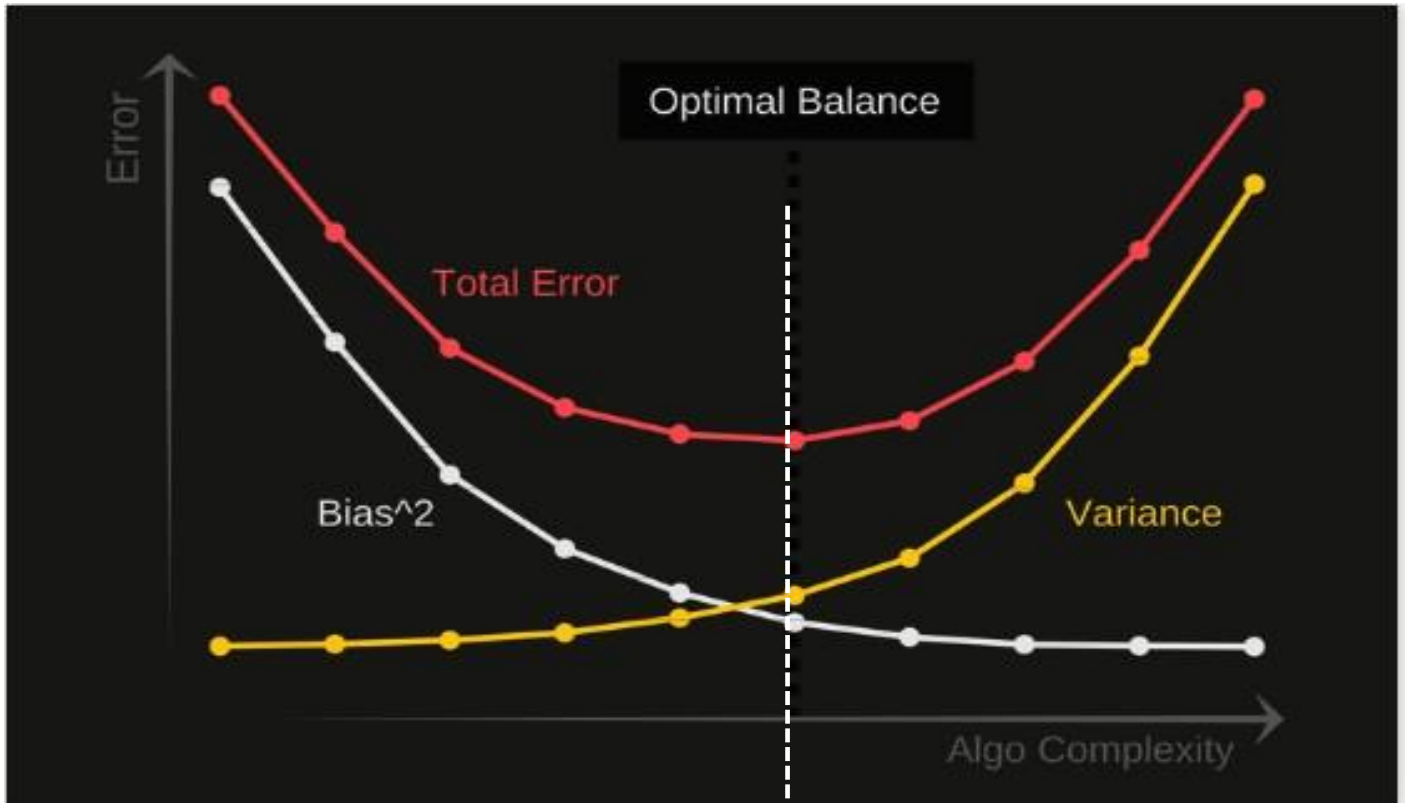
- Low Bias and Low Variance will ensure the model will have the maximum possible accuracy.



Four Different Scenarios at the end of Training

Think Before Designing Further





- we want an optimized model that contains low bias and low variance such that it minimizes the total error.

What is Regularization?

- Regularization is one of the ways to improve our model to work on unseen data by ignoring the less important features.
- Regularization minimizes the validation loss and tries to improve the accuracy of the model.
- It avoids overfitting by adding a penalty to the model with high variance, thereby shrinking the beta coefficients to zero.

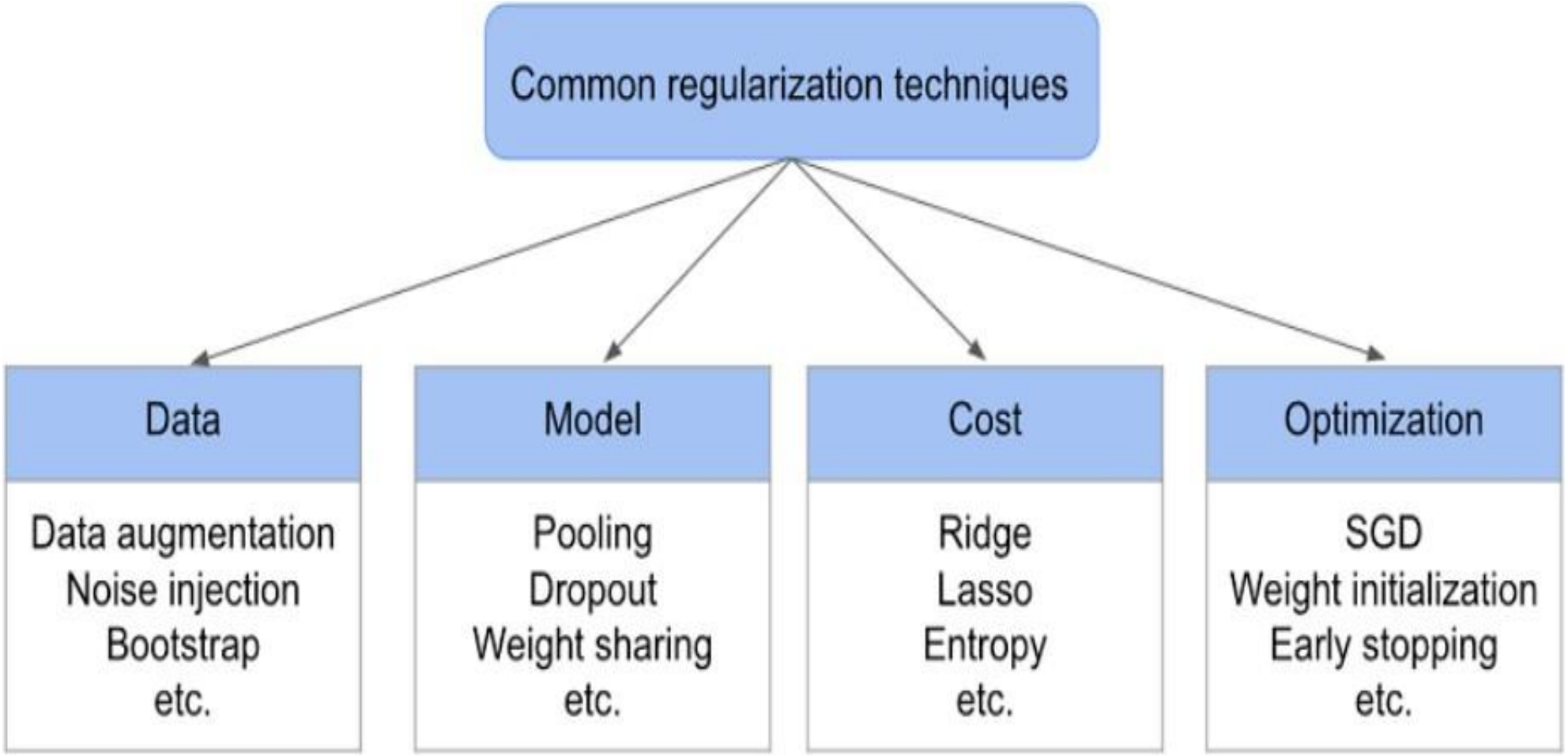
Various Regularization Methods

- Adding restrictions on the parameter values.
- Extra terms in the objective function that can be thought of as corresponding to a soft constraint on the parameter values.
- These constraints and penalties are designed to express a generic preference for a simpler model class in order to promote generalization.
- Sometimes penalties and constraints are necessary to make an underdetermined problem determined.

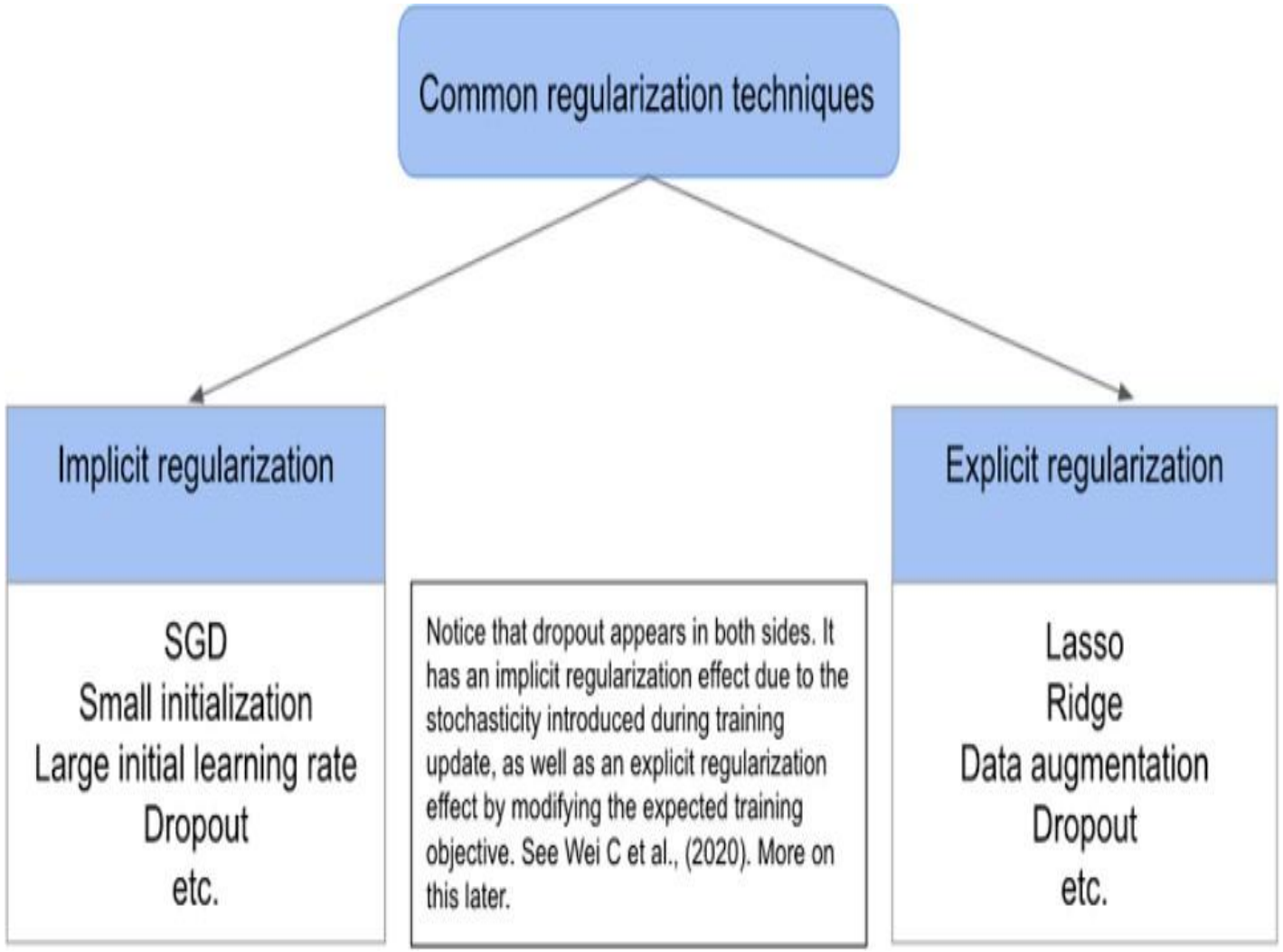
Regularization in Deep Learning Context

- Most regularization strategies are based on regularizing estimators.
- An effective regularize is one that makes a profitable trade, reducing variance significantly while not overly increasing the bias.
- There will be no access to the true data-generating process so we are not sure to see if the model family being estimated includes the generating process or not.
- Yet, most applications of deep learning algorithms are to domains where the true data-generating process is almost certainly outside the model family.
- Deep learning algorithms are typically applied to extremely complicated domains such as images, audio sequences, and text, for which the true generation process essentially involves simulating the entire universe.

Categorizing Regularization Techniques



Categorizing Regularization Techniques

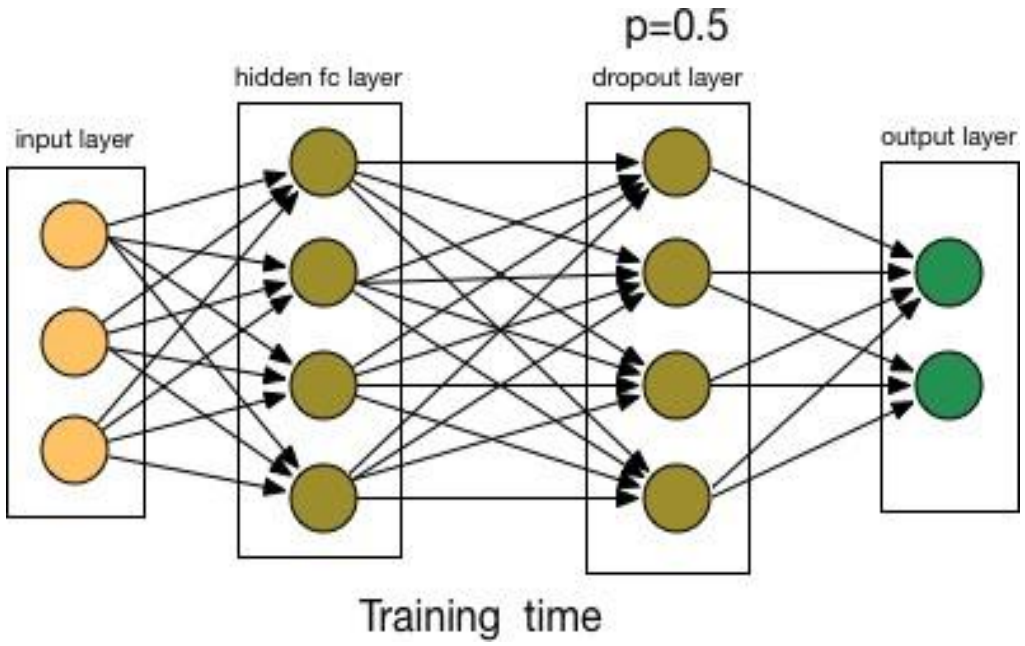


- Implicit regularization refers to a particular operation that indirectly plays a regularizing role.

- Explicit regularizes specifically target tweaking specific components during model training, such as controlling model complexity.

Regularization Techniques

- Lasso Regularization (L1)
- L2 Regularization -Ridge Regression
- Dataset Augmentation
- Dropout
- Early Stopping



What is Lasso Regularization (L1)?

- It stands for Least Absolute Shrinkage and Selection Operator.
- It adds L1 the penalty
- It is used over regression methods for a more accurate prediction.
- This model uses shrinkage. Shrinkage is where data values are shrunk towards a central point as the mean.
- The lasso procedure encourages simple, sparse models (i.e. models with fewer parameters).
- This particular type of regression is well-suited for models showing high levels of multicollinearity or when you want to automate certain parts of model selection, like variable selection/parameter elimination

Lasso Regularization (L1)

Mathematically, we express L1 regularization by extending our loss function like such:

$$LossFunction = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y} - Y)^2 + \lambda \sum_{i=1}^{N} |\theta_i|$$

(Or)

Cost function = Loss + λ + Σ ||w||

Loss = sum of squared residual
λ = penalty
w = slope of the curve

- Essentially, when we use L1 regularization, we are penalizing the absolute value of the weights.

- L1 regularization is the preferred choice when having a high number of features as it provides sparse solutions.
- The L1 regularization focuses on reducing the steepness of the best-fit line to zero, which is used to reduce the value of the cost function.
- that results in more accurate outputs.

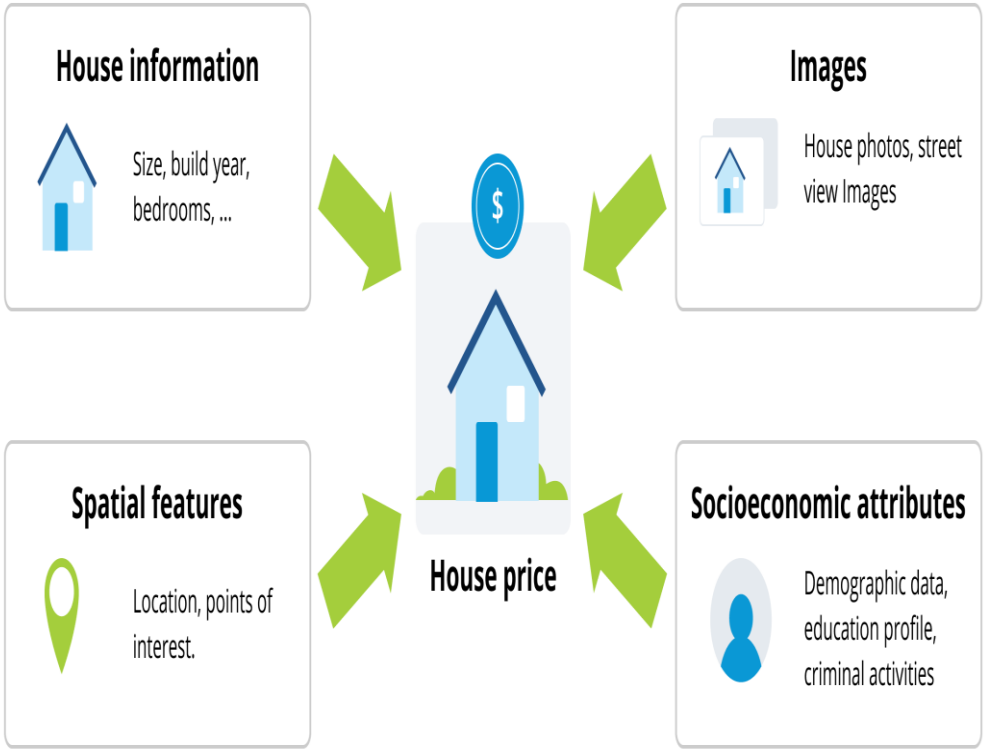
Lasso Regularization (L1)

- λ is the regularization parameter and is directly proportional to the amount of regularization applied.
- λ =0, then no regularization is applied.
- λ= 1 maximum regularization is applied to the network.
- λ is a hyperparameter which means it is not learned during the training but is tuned by the user manually or using some hyperparameter tuning techniques such as random search.

Example: Predict House Prices

Features

- Street-road access,
- Neighborhood – property location,
- Accessibility – transport access,
- Year Built – the year the house was built,
- Rooms – number of rooms,
- Kitchens – number of kitchens,
- Fireplaces – number of fireplaces in the house.



Example cont'd -Apply L1 Regularization

- For example, consider the neighborhood or the number of rooms has a higher influence on the price of the property than the number of fireplaces.
- Using this technique weight of no.of.fireplaces =0 -> no impact on the price
- The neighborhood and the number of rooms to be given non-zero weights, since these features influence the price of a property significantly.
- For example, the year our home was built and the number of rooms in the home may have a high correlation.
- When L1 regularization is used then we have highly correlated features, the L1 norm would select only 1 of the features from the group of correlated features in an arbitrary nature, which is something that we might not want.

L2 Regularization -Ridge Regression

- L2 regularization can deal with the multicollinearity (independent variables are highly correlated) problems by constricting the coefficient and by keeping all the variables.
- Ridge regression adds the "squared magnitude" of the coefficient as the penalty term to the loss function.

$$LossFunction = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y} - Y)^2 + \lambda\sum_{i=1}^{N}\theta_i^2$$

(Or)

Cost function = Loss + λ + Σ ||w||²

- L2 regression can be used to estimate the significance of predictors and based on that it can penalize the insignificant predictors.

L2 Regularization -Ridge Regression

Assume the cross-entropy cost function is

$$C = -\frac{1}{n}\sum_{xj}\left[y_j \ln a_j^L + (1-y_j)\ln(1-a_j^L)\right]$$

where y_j is the true class label and a_j^L is the predicted class label of layer L

- To apply L2 regularization to any network having cross-entropy loss, we add the regularizing term to the cost function.

$$\frac{\lambda}{2n}\sum_w w^2.$$

$$C = -\frac{1}{n}\sum_{xj}\left[y_j \ln a_j^L + (1-y_j)\ln(1-a_j^L)\right] + \frac{\lambda}{2n}\sum_w w^2.$$

The general formula for L2 regularization

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2,$$

The differences between L1 and L2 regularization

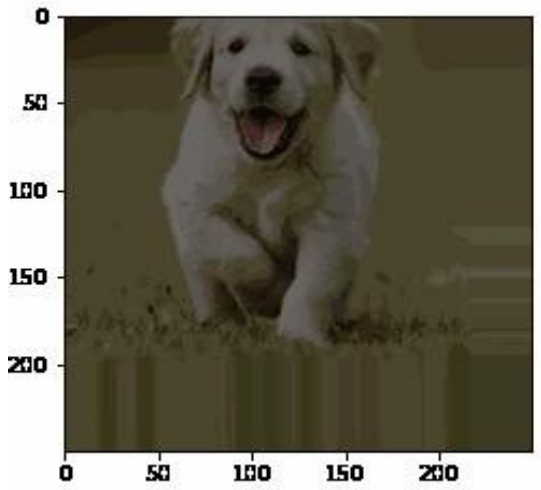
L1 Regularization	L2 Regularization
Panelizes the sum of absolute value of weights.	penalizes the sum of square weights.
It has a sparse solution.	It has a non-sparse solution.
It gives multiple solutions.	It has only one solution.
Constructed in feature selection.	No feature selection.
Robust to outliers.	Not robust to outliers.
It generates simple and interpretable models.	It gives more accurate predictions when the output variable is the function of the whole input variables.
Unable to learn complex data patterns.	Able to learn complex data patterns.
Computationally inefficient over non-sparse conditions.	Computationally efficient because of having analytical solutions.

Choosing between L1 and L2

- The choice between L1 and L2 regularization depends on the specific problem and the goals of the model.
- L1 regularization is well suited for feature selection, as it promotes sparse solutions and can be used to identify the most important features in the data.
- L2 regularization is well suited for preventing overfitting, as it reduces the variance of the model and improves its generalization performance.

Dataset Augmentation

- Dataset augmentation applies transformations to your training examples.
- They can be as simple as flipping an image, or as complicated as applying neural style transfer.
- The idea is that by changing the makeup of your data, you can improve your performance and increase your training set size.
    - Audio Data Augmentation
    - Text Data Augmentation
    - Image Augmentation



Dataset Augmentation Cont'd

When Should You Use Data Augmentation?

- To prevent models from overfitting.
- The initial training set is too small.
- To improve the model accuracy.
- To Reduce the operational cost of labeling and cleaning the raw dataset.

Limitations of Data Augmentation

- The biases in the original dataset persist in the augmented data.
- Quality assurance for data augmentation is expensive.
- Research and development are required to build a system with advanced applications.
  - For example, generating high-resolution images using GANs can be challenging.
- Finding an effective data augmentation approach can be challenging.

Data Augmentation Techniques

Audio Data Augmentation

- Noise injection
- Shifting Changing the speed
- Changing the pitch

Text Data Augmentation

- Word or sentence shuffling
- Word replacement
- Syntax-tree manipulation
- Random word insertion

Image Augmentation

- Geometric transformations
- Color space transformations
- Kernel filters
- Random erasing
- Mixing images

Advanced Techniques

- Generative adversarial networks (GANs)
- Neural Style Transfer

Noise Robustness

- Noise applied to inputs is a data augmentation, For some models, the addition of noise with extremely small variance at the input is equivalent to imposing a penalty on the norm of the weights.
- Noise applied to hidden units, Noise injection can be much more powerful than simply shrinking the parameters. Noise applied to hidden units is so important that Dropout is the main development of this approach.
- Adding Noise to Weights, This technique is primarily used with Recurrent Neural Networks(RNNs). This can be interpreted as a stochastic implementation of Bayesian inference over the weights.

Noise Robustness Cont'd

- Bayesian learning considers model weights to be uncertain and representable via a probability distribution p(w) that reflects that uncertainty.
- Adding noise to weights is a practical, stochastic way to reflect this uncertainty.
- Noise applied to weights is equivalent to traditional regularization, encouraging stability.
- This can be seen in a regression setting.
- Train $\hat{y}(x)$ to map x to a scalar using least squares between model prediction $\hat{y}(x)$ and true values y.

$$J = E_{p(\boldsymbol{x},y)}\left[\left(\hat{y}(\boldsymbol{x}) - y\right)^2\right]$$

Training set: m labeled samples $\{(\boldsymbol{x}^{(1)},y^{(1)}),\ldots,(\boldsymbol{x}^{(m)},y^{(m)})\}$

- We perturb each input with εW~N(ε;0,ηI) For small η, this is equivalent to a regularization term η Ep(x,y)[‖∇wŷ (x)‖2]. It encourages parameters to regions where small perturbations of weights have a small influence on the output.

Injecting Noise at Output Targets

- Most datasets have some mistakes in y labels.

  - Harmful to maximize log $p(y|x)$ when y is a mistake.

- To prevent it we explicitly model noise on labels

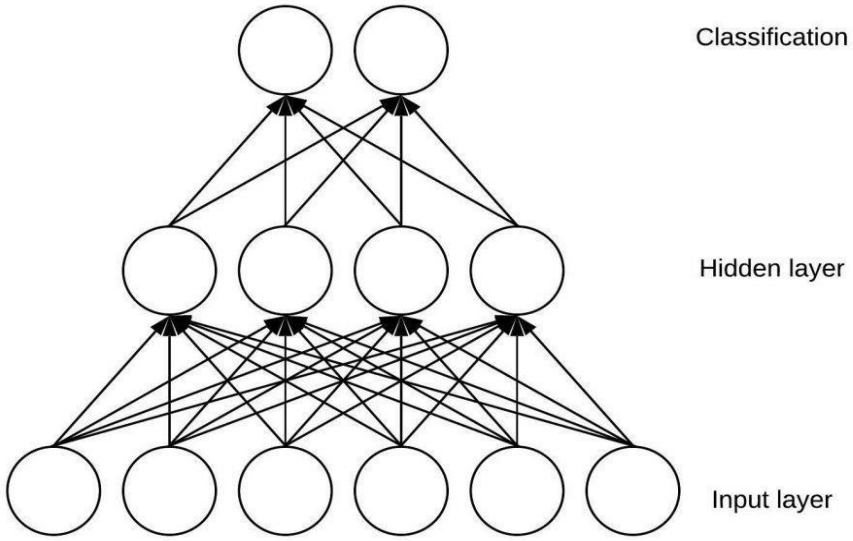
- Ex: we assume training set label y is correct with probability 1-ε, and otherwise any of the other labels may be correct.

- This can be incorporated into the cost function.

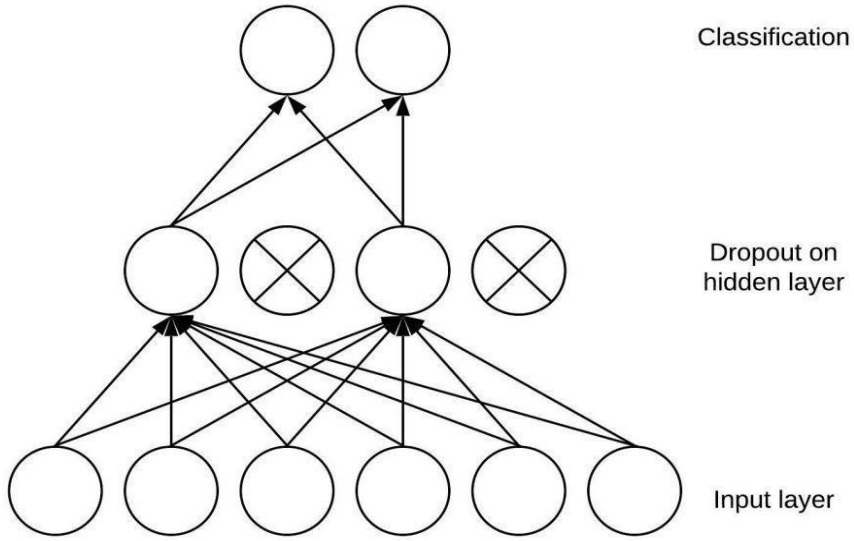
- Ex: Local Smoothing regularizes a model based on a softmax with k output values by replacing the hard 0 and 1 classification targets with targets of ε/(k-1) and 1-ε respectively.

Dropout

- Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel.
- During training, some number of layer outputs are randomly ignored or "dropped out."
- Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsible for the inputs.



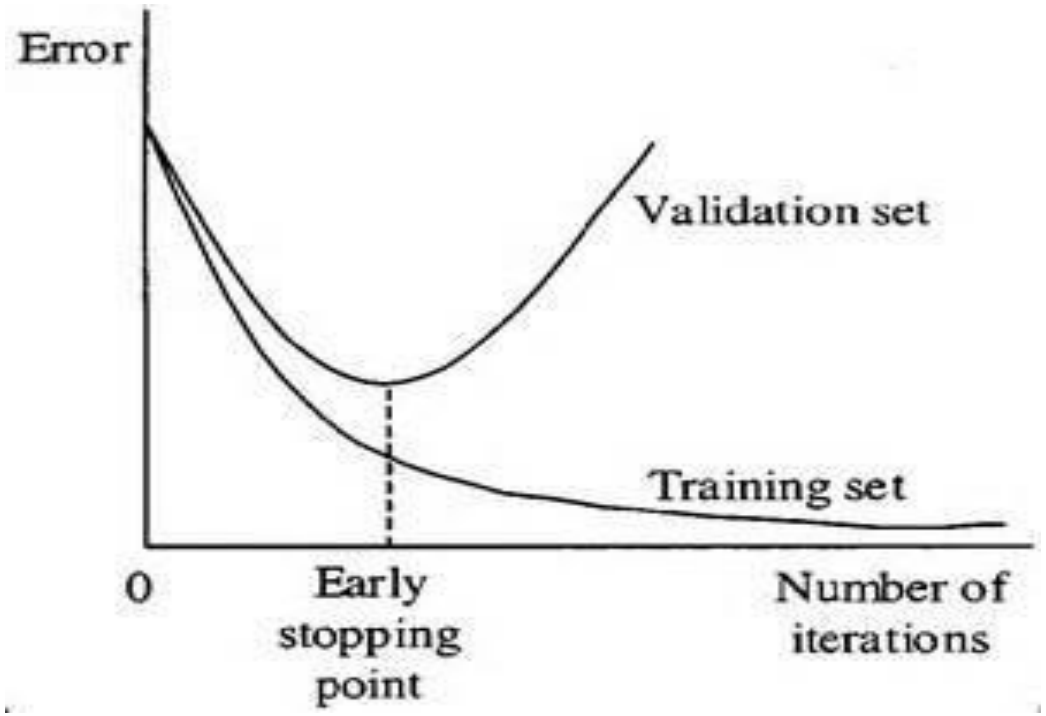
Without Dropout



With Dropout

Early Stopping

- Early stopping is an optimization technique used to reduce overfitting without compromising on model accuracy. The main idea behind early stopping is to stop training before a model starts to overfit.



Early stopping approaches

- Training model on a preset number of epochs.
- Stop when the loss function update becomes small
- Validation set strategy

Early Stopping Cont'd

Training model on a preset number of epochs

- This method is a simple, but naive way to early stop.
- By running a set number of epochs, we run the risk of not reaching a satisfactory training point.
- With a higher learning rate, the model might possibly converge with fewer epochs, but this method requires a lot of trial and error.
- Due to the advancements in machine learning, this method is pretty obsolete.

Stop when the loss function update becomes small

- The training is stopped when the update becomes as small as 0.001, as stopping at this point minimizes loss and saves computing power by preventing any unnecessary epochs.
- However, overfitting might still occur.

Early Stopping Cont'd

Validation set strategy

- it's important to look at how training and validation errors change with the number of epochs.
- The training error decreases exponentially until increasing epochs no longer have such a large effect on the error.
- The validation error, however, initially decreases with increasing epochs, but after a certain point, it starts increasing. This is the point where a model should be early stopped as beyond this the model will start to overfit.