Introduction to Machine Learning with Python

Regularization

Instructor: Tugce Gurbuz

Aug 10th, 2023

Complex Models Need Regularization

In ML, we use highly complex models with many adjustable parameters.

- Learning the noise in data?

Source

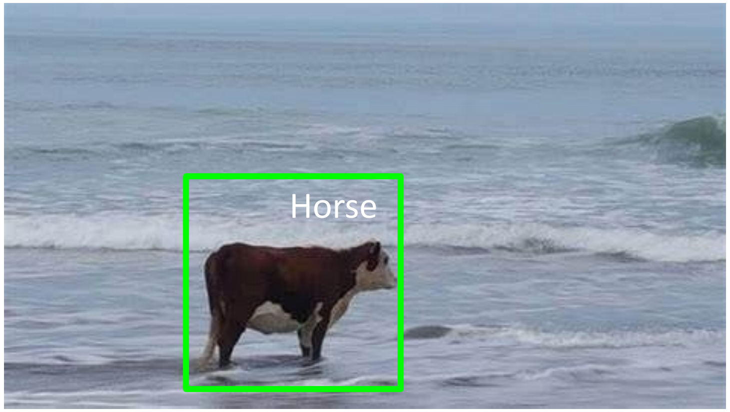
Model Complexity Matters!

- Too simple -> “underfits”

Source

Model Complexity Matters!

- Too complex -> “overfits”

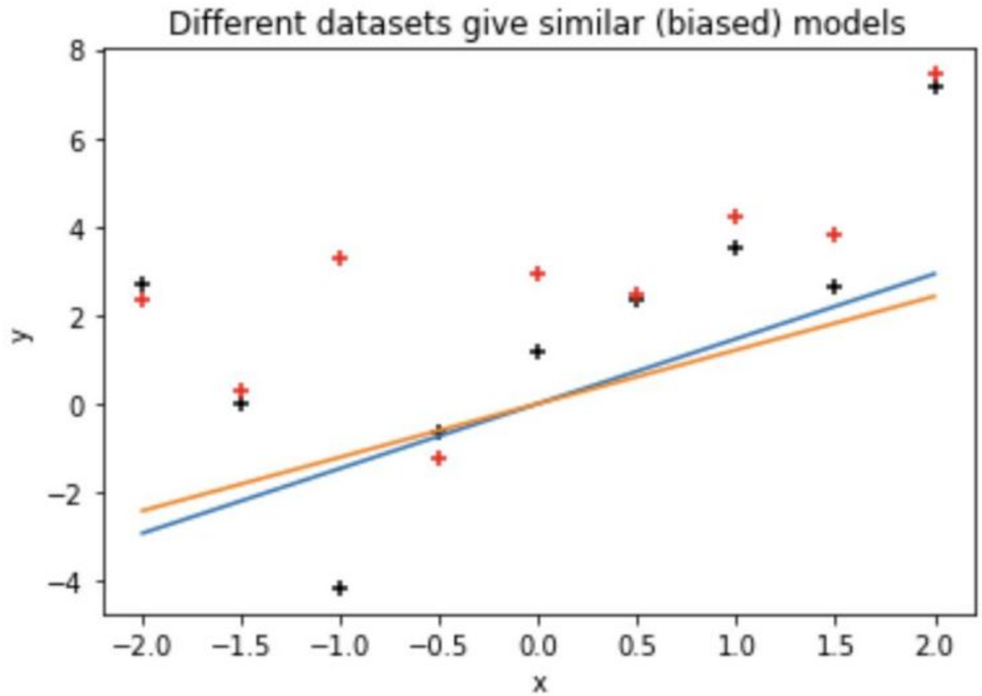


Source

Model Complexity Matters!

Bias vs Variance Tradeoff

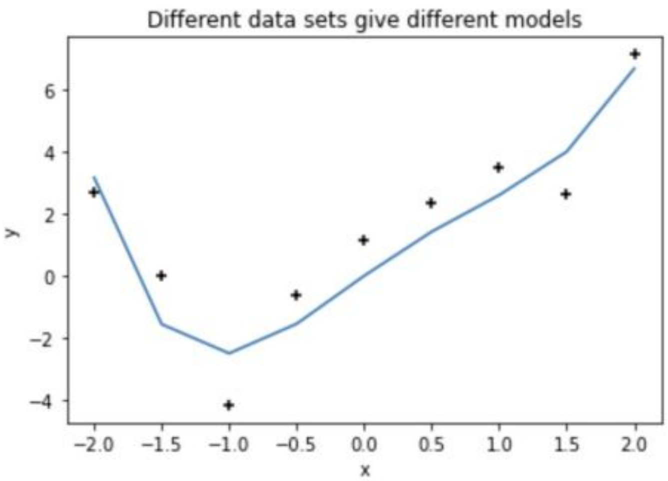
- Bias: The simple models often give systematically too small weights.

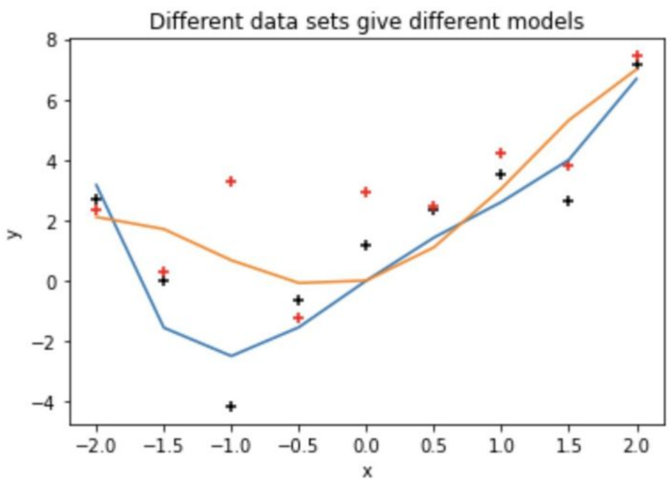


Model Complexity Matters!

Bias vs Variance Tradeoff

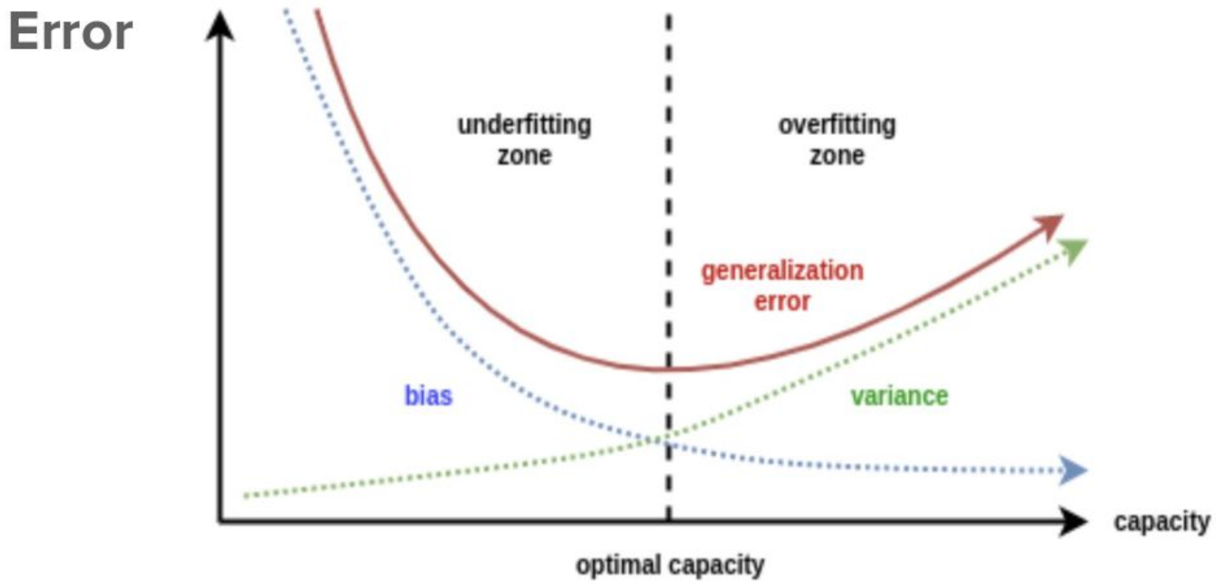
- Bias: The simple models often give systematically too small weights.
- Variance: The complex model capture the variance too much that leads to poor generalization





Model Complexity Matters!

How to pick right model complexity?



Source:https://images.deepai.org/

Model Complexity Matters!

The magic!

Deep learning on images (Zhang, Bengio, Hardt, Recht, and Vinyalsn 2017)

- gives 0 training error -- and small test error
- gives 0 training error with randomized labels

Model Complexity Matters!

The magic!

Deep learning on images (Zhang, Bengio, Hardt, Recht, and Vinyalsn 2017)

- gives 0 training error -- and small test error
- gives 0 training error with randomized labels

Large language models memorize a lot

- GPT-3 (175B params trained on 500B words) seems to memorize a lot

  Q. What do you call a droid that takes the long way around?

Model Complexity Matters!

The magic!

Deep learning on images (Zhang, Bengio, Hardt, Recht, and Vinyalsn 2017)

- gives 0 training error -- and small test error
- gives 0 training error with randomized labels

Large language models memorize a lot

- GPT-3 (175B params trained on 500B words) seems to memorize a lot

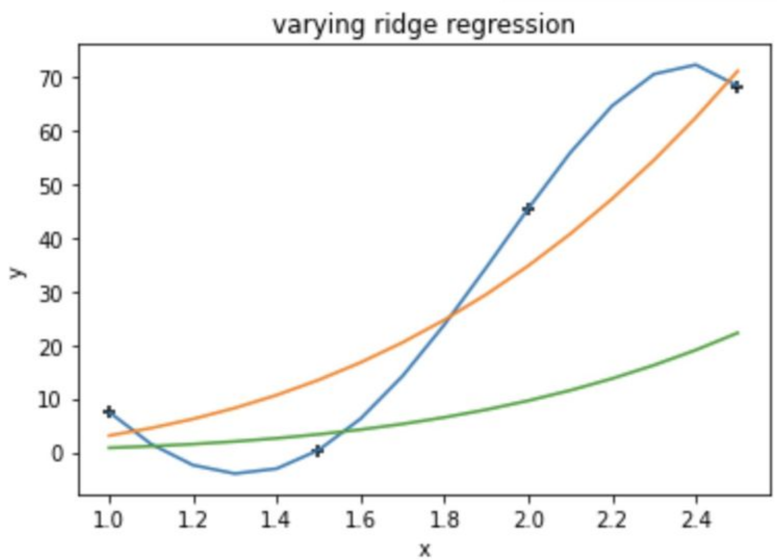
  Q. What do you call a droid that takes the long way around?

  R2 detour.

Characteristics of Regularized Models

The more regularized models give us:

- Smaller weights (less fitting to noise)
- Smoother models
- Models with lower capacity



Common Regularization Techniques

L1 and L2 penalties:

- Train to minimize normal loss + c * L1(weights)
  - L1: Lasso regression
  - Drives some weights to 0

Common Regularization Techniques

L1 and L2 penalties:

- Train to minimize normal loss + c * L1(weights)
  - L1: Lasso regression
  - Drives some weights to 0

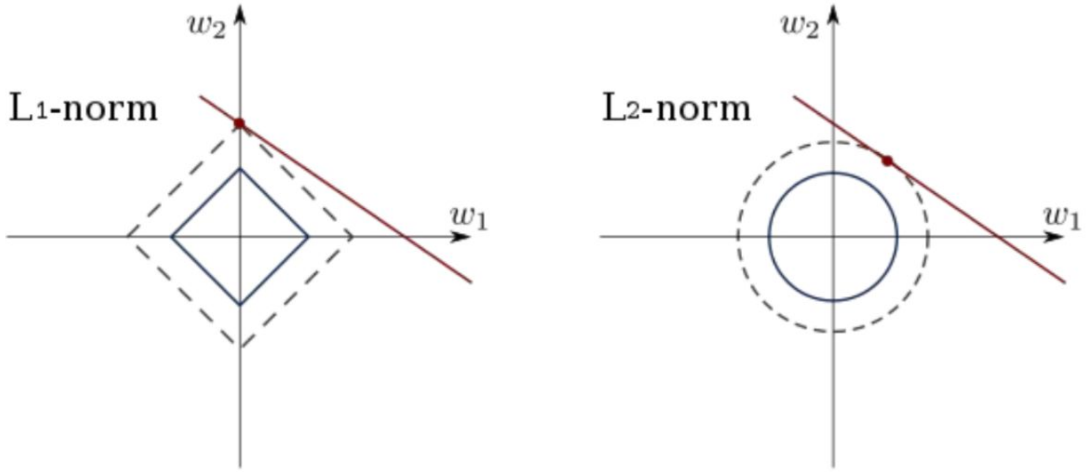
- Train to minimize normal loss + c * L2(weights)
  - L2: ridge regression
  - Makes biggest weights smaller.

Common Regularization Techniques

L1 and L2 penalties:

- Train to minimize normal loss + c * L1(weights)
    - L1: Lasso regression
    - Drives some weights to 0

- Train to minimize normal loss + c * L2(weights)
    - L2: ridge regression
    - Makes biggest weights smaller.



Common Regularization Techniques

L1 and L2 penalties:

- Train to minimize normal loss + c * L1(weights)
  - L1: Lasso regression
  - Drives some weights to 0

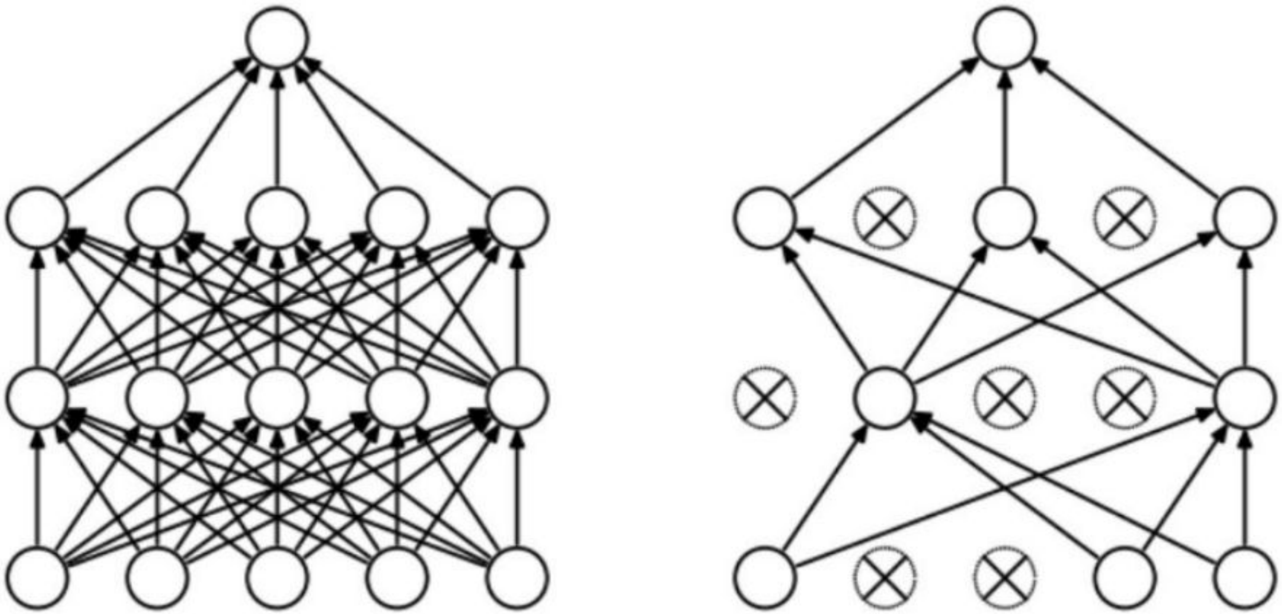
- Train to minimize normal loss + c * L2(weights)
  - L2: ridge regression
  - Makes biggest weights smaller.

L-infinity penalty:

- Train to minimize normal loss - but don’t let the weights get too big

Common Regularization Techniques

Drop-out



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

Common Regularization Techniques

Early stopping -> Avoid overfitting by stopping your training at the right time

Common Regularization Techniques

Data Augmentation

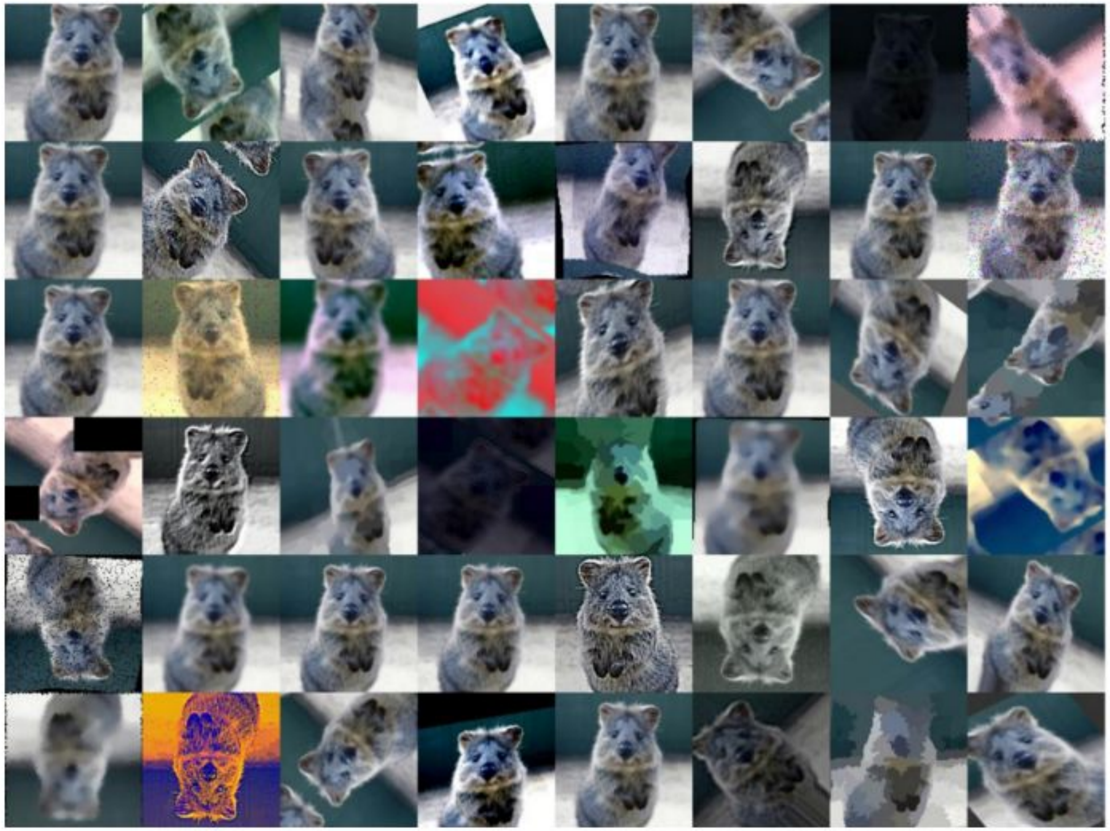


Image Augmentation Sample. Image by https://github.com/aleju/imgaug

Common Regularization Techniques

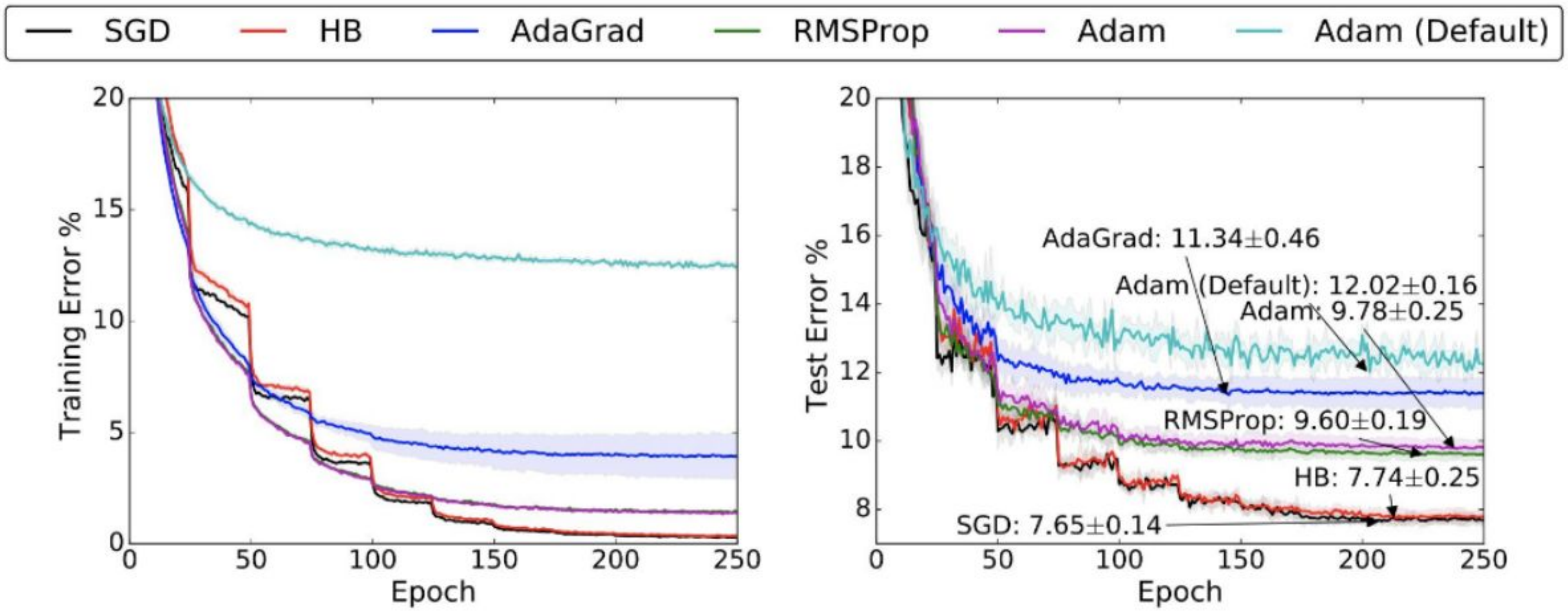
Stochastic Gradient Descent (SGD)

- Initialize with small random weights
- Weights get bigger as one iterates
- Use early stopping to avoid overfitting

Common Regularization Techniques

Stochastic Gradient Descent (SGD)

- Initialize with small random weights
- Weights get bigger as one iterates
- Use early stopping to avoid overfitting



(a) CIFAR-10 (Train) (b) CIFAR-10 (Test)

Common Regularization Techniques

Learning rates and regularization

- Small learning rates are more likely to find a deep minimum -> might be good or bad

Common Regularization Techniques

Learning rates and regularization

- Small learning rates are more likely to find a deep minimum -> might be good or bad
- Large learning rates misses deep minimas, finds broader and flatter minimas that may be more robust.

Common Regularization Techniques

Conclusion on regularization techniques: Use all!

Common Regularization Techniques

Hyperparameter tuning is a search:

- **Grid Search:** Try all possible combinations of hyperparameters
- **Random Search:** Randomly try different combinations of hyperparameters
- **Coordinate-wise Gradient Descent:** Start at one set of hyperparameters and try changing one at a time, accept any changes that reduce your validation error
- **Bayesian Optimization / Auto ML:** Start from a set of hyperparameters that have worked well on a similar problem, and then do some sort of local exploration (e.g., gradient descent) from there.