MACHINE INTELLIGENCE

UNIT - 5

Regularization

feedback/corrections: vibha@pesu.pes.edu

VIBHA MASTI

1. Bias-Variance Trade-off

- Sample : 25 points out of 100 for the function $f(x) = \sin x$

- Train simple model

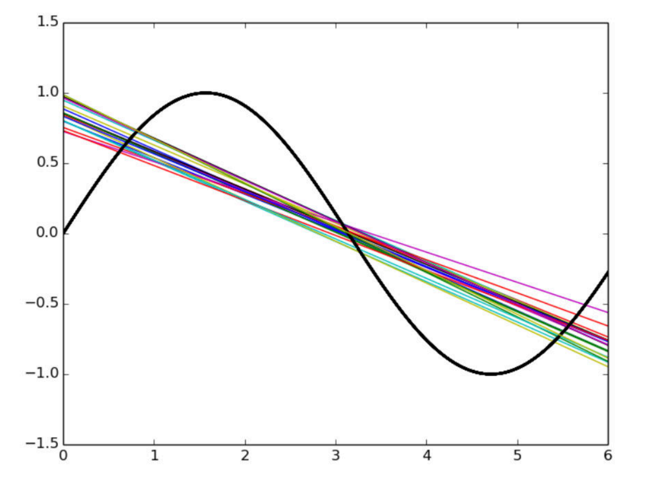
$$y = \hat{f}(x) = w_1 x + w_0$$

- Train complex model

$$y = \hat{f}(x) = \sum_{i=1}^{25} w_i x^i + w_0$$

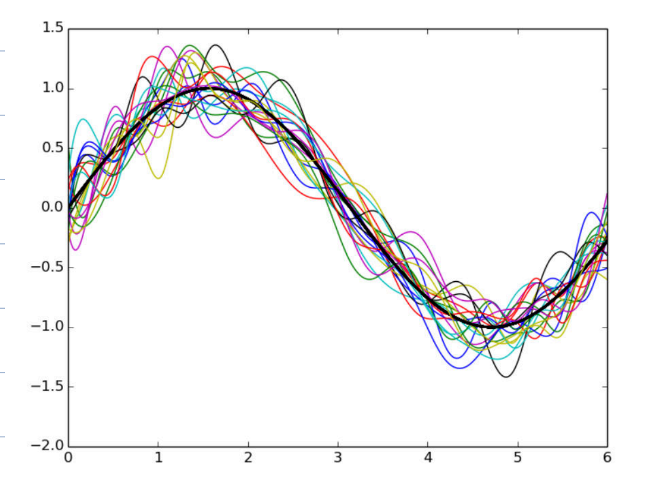
- Train k different simple & complex models on different samples of training data

- Each model gets its own sample



simple models do not differ from each other much

however, they are underfitting (high bias) - very far



complex models differ from each other a lot (high variance)

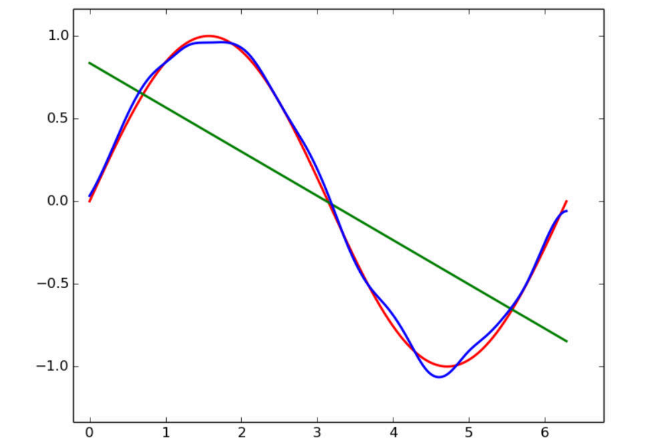
Let $f(x)$ = true model

Let $\hat{f}(x)$ = estimate of model

1.1 Bias

$$\text{bias}(\hat{f}(x)) = E[\hat{f}(x)] - f(x)$$

$E[\hat{f}(x)]$ is the average of all possible $\hat{f}(x)$ models / expected value of $\hat{f}(x)$



Green: E(simple model)
Blue: E(complex model)

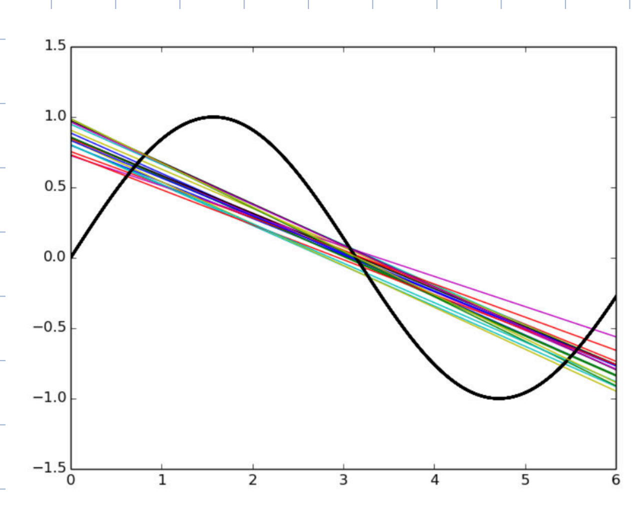
Red: true model

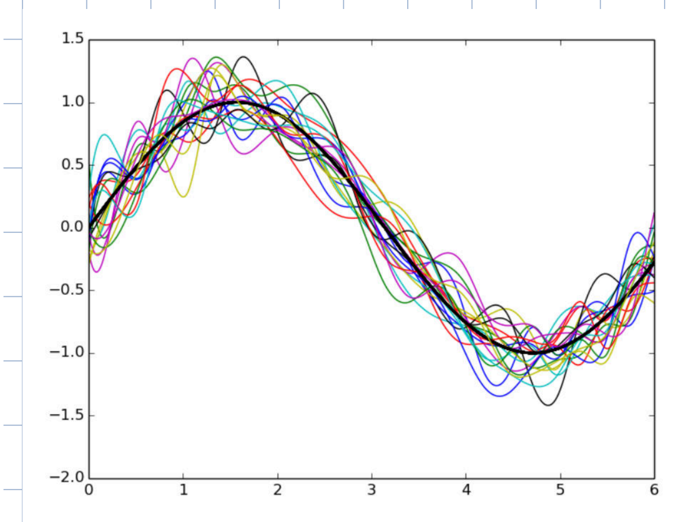
- Simple model has high bias
- Complex model has low bias

1.2 Variance

$$\text{variance}(\hat{f}(x)) = E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right]$$

$E[(\hat{f}(x) - E[\hat{f}(x)])^2]$ is the variance of every possible model wrt the average model

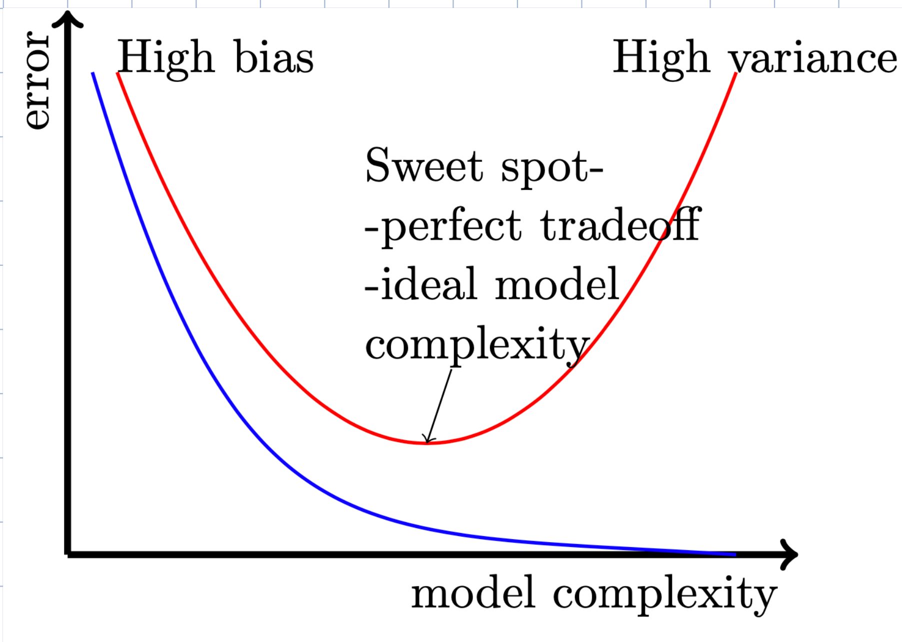




- Simple model has low variance
- Complex model has high variance

1.3 Train-Test Error (overfitting)

- As model complexity increases, train error decreases but test error increases



Blue: $train_{err}$

Red: $test_{err}$

- Data $D = \{x_i, y_i\}_{i=1}^{m+n}$ split into train (n) and test (m)

- For any point (x_i, y_i),

$$y_i = f(x_i) + \varepsilon_i$$ ← noise (assume white)

- $\varepsilon \sim N(0, \sigma^2)$

- f is unknown, only $\hat{f}$ known after training such that

$$y_i = \hat{f}(x_i) \quad \forall \quad (x_i, y_i) \in D_{train}$$

2. Regularization

- Introduce extra information in the model to prevent overfitting

- $\min_{\text{wrt } \theta} \mathcal{L}oss_{train}(\theta) + \Omega(\theta) = \mathcal{L}(\theta)$

  ↖ increases with model complexity

- For us: L1 & L2 regularization

2.1 p-norm of a vector w

$$\|w\|_p = (|w_1|^p + |w_2|^p + \dots + |w_N|^p)^{\frac{1}{p}}$$

2.2 L1 Regularization

- If model is linear regression and it uses L1 regularization, it is called Lasso regression

- LR model

$$\hat{y} = w_1 x_1 + w_2 x_2 + \dots + w_N x_N + b$$

- Loss function without L1 regularization

$$\text{loss} = \text{error}(y, \hat{y})$$ ← could be MSE

- Loss function with L1 regularization

$$\text{loss} = \text{error}(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

$$= \text{error}(y, \hat{y}) + \lambda \|w\|_1$$

2.3 L2 Regularization

- If model is linear regression and it uses L2 regularization, it is called Ridge regression

- loss function with L2 regularization

$$\text{loss} = \text{error}(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|^2$$

$\tilde{L}(w)$

$$= \text{error}(y, \hat{y}) + \lambda \left(\|w\|_2\right)^2$$

empircal estimate of training error $(L(w))$

Rewriting as

for mathematical simplicity

$$\tilde{\mathcal{L}}(w) = \mathcal{L}(w) + \frac{\alpha}{2}\|w\|_2^2$$

- In SGD, we want the gradient of the loss function wrt the weight

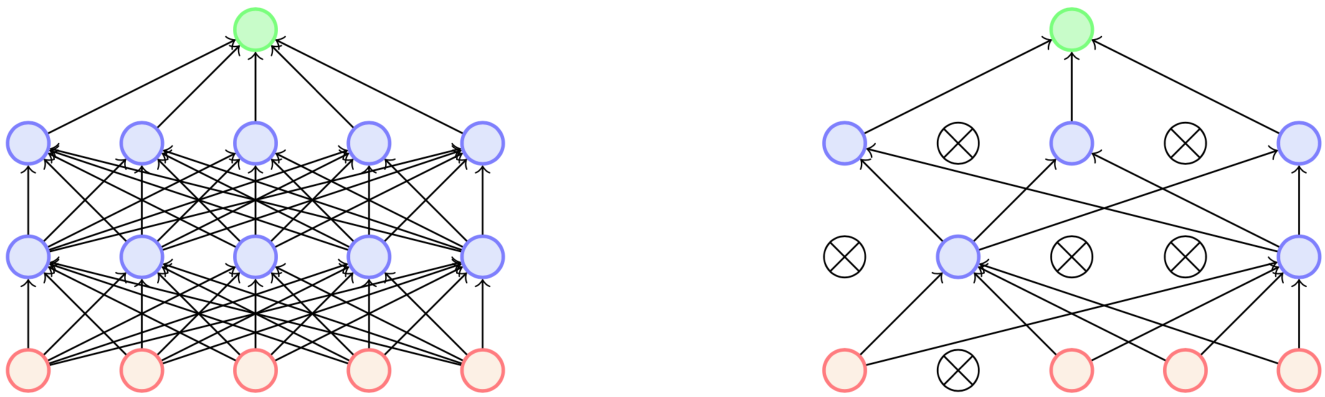
$$\nabla\tilde{\mathcal{L}}(w) = \nabla\mathcal{L}(w) + \alpha w$$

- Update rule

$$w_{t+1} = w_t - \eta\nabla\mathcal{L}(w_t) - \eta\alpha w_t$$

3. Dropout

- Computationally expensive to train multiple NNs and create an ensemble
- In neural network, randomly drop some nodes with all their incoming and outgoing edges
- Each node retained with fixed prob (typically 0.5) for hidden nodes and $p = 0.8$ for visible nodes (input)



- If n nodes present, total possible no of thinned networks $= 2^n$
- Probability p is a hyperparameter
- Combining all models: use full NN and scale output of each node by the fraction of times it was on during training

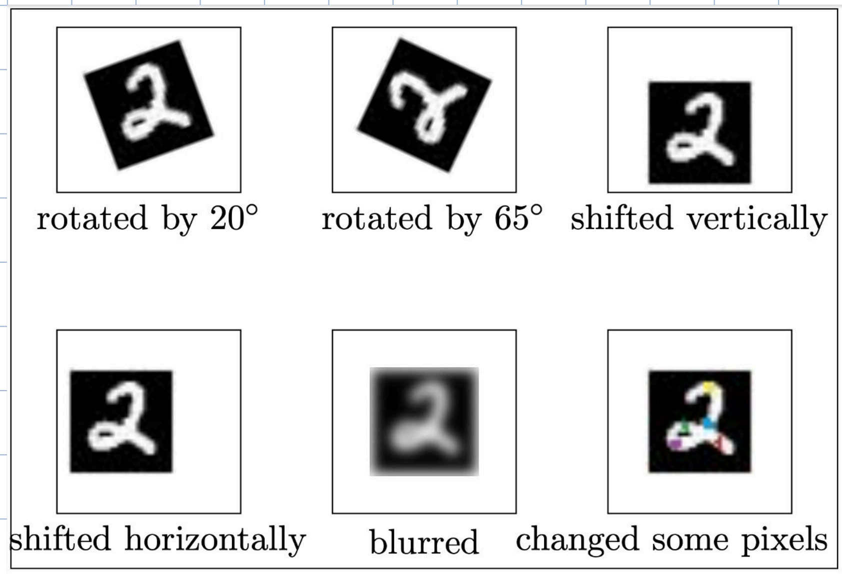
4. Dataset Augmentation

- Another technique to reduce overfitting without having to fetch more training data
- Transform training data so that the labels do not change (eg: shifting, scaling, rotating an image)

- Augmented data for given training data 2


label = 2

augment →



label = 2

5. Early stopping

- split training (labelled) data into train and validation
- If performance of validation set goes down, stop training