Me: Mom, can we have some AI?

Mom: No, we have AI at home.

The AI at home:

$$f(x,\mathbf{W}) = \mathbf{W}\,\mathbf{x} + \mathbf{b}$$

Intro Homework 2

Homework 2 Walkthrough

- Programming assignment, due Sep 20.
- Setting up environment (local, Google Colab, Kaggle)
- GradeScope submission guideline
  - Homework 2 – report
    - This should include your answers/discussion to each question, and all the plots generated in the programming part (Q2-Q4).
  - Homework 2 – notebook
    - Notebook file (ipynb file)
    - Exported PDF version of your notebook with all cell outputs.

Homework 2 Walkthrough

- **Q1 – Backpropagation**

$$f(x_1, x_2, w_1, w_2) = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2)}} + 0.5(w_1^2 + w_2^2)$$

  - **(a) Calculate the following** $\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$
  - **(b) Create computational graph, include values in forward/backward passes. Initial values are:** $w_1 = 0.2, w_2 = 0.4, x_1 = -0.4, x_2 = 0.5$
  - **(c) Formulation of loss function**

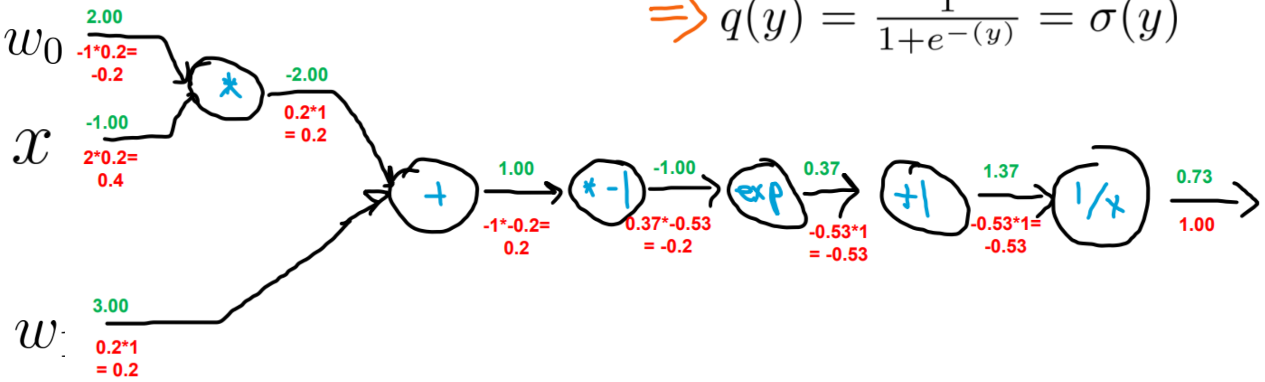
Homework 2 Walkthrough

- **Q1 – Backpropagation**
  - **Computational graph example from lecture notes**

$$f(w,x) = \frac{1}{1+e^{-\underbrace{(w_0x+w_1)}_{y}}}$$

$$\Rightarrow q(y) = \frac{1}{1+e^{-(y)}} = \sigma(y)$$

Forward pass



Homework 2 Walkthrough

- Q2 – AutoGrad
  - Implement the “TODOs” in Homework2.ipynb, including
    - Missing operations
    - backward function

```python
class Value:

    """
    Basic unit of storing a single scalar value and its gradient
    """

    def __init__(self, data, _children=()):
        """


        """
        self.data = data
        self.grad = 0
        self._prev = set(_children)
        self._backward = lambda: None
```

```python
def __add__(self, other):

    other = other if isinstance(other, Value) else Value(other)

    out = Value(self.data + other.data, (self, other))

    def _backward():
        self.grad += out.grad * 1.0
        other.grad += out.grad * 1.0
    out._backward = _backward

    return out
```

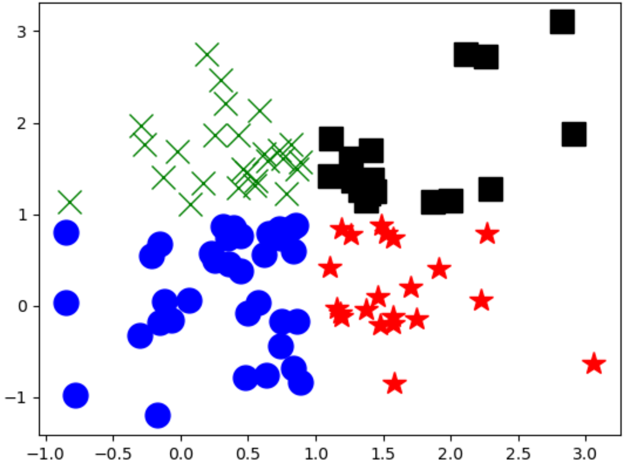
Homework 2 Walkthrough

- Q2 – AutoGrad
  - Implement the “TODOs” in Homework2.ipynb, including
    - Missing operations
    - backward function

```python
def backward(self):
    """
    Run backpropagation from the current Value
    """
    #This function is called when you start backpropagation from this Value

    #The gradient of this value is initialized to 1 for you.
    self.grad = 1

    #You need to find a right topological order all of the children in the graph.
    #As for topology sort, you can refer to http://www.cs.cornell.edu/courses/cs312/2004fa/lectures/lecture15.ht

    topo = []
    #TODO find the right list of Value to be traversed
    '''
    Hint: you can recursively visit all non-visited node from the node calling backward.
    add one node to the head of the list after all of its children node are visited
    '''

    #go one variable at a time and apply the chain rule to get its gradient

    for v in topo:
        v._backward()
```

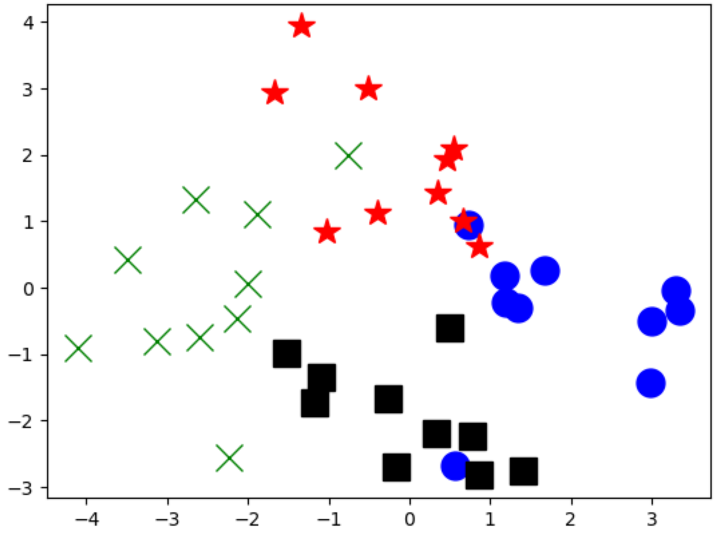
Homework 2 Walkthrough

- Q3 – Linear Classifier
  - Implement linear classifier in `LinearLayer` class
  - Implement loss functions (softmax, cross entropy loss) and accuracy computation
  - Implement training procedure
  - Train Linear classifier on HW2_Q3_Dataset



Homework 2 Walkthrough

- Q4 – MLP
  - Implement Multi-layer perceptron class
  - Train MLP on HW2_Q4_Dataset



Recitation

Linear Classification

Linear Classifier

Input image

x

Array of 32x32x3 numbers
(3072 numbers total)

3072 x 1

$$\boxed{f(x,W)} = \boxed{W}\boxed{x} + \boxed{b}$$

10 x 1   10 x 3072   10 x 1

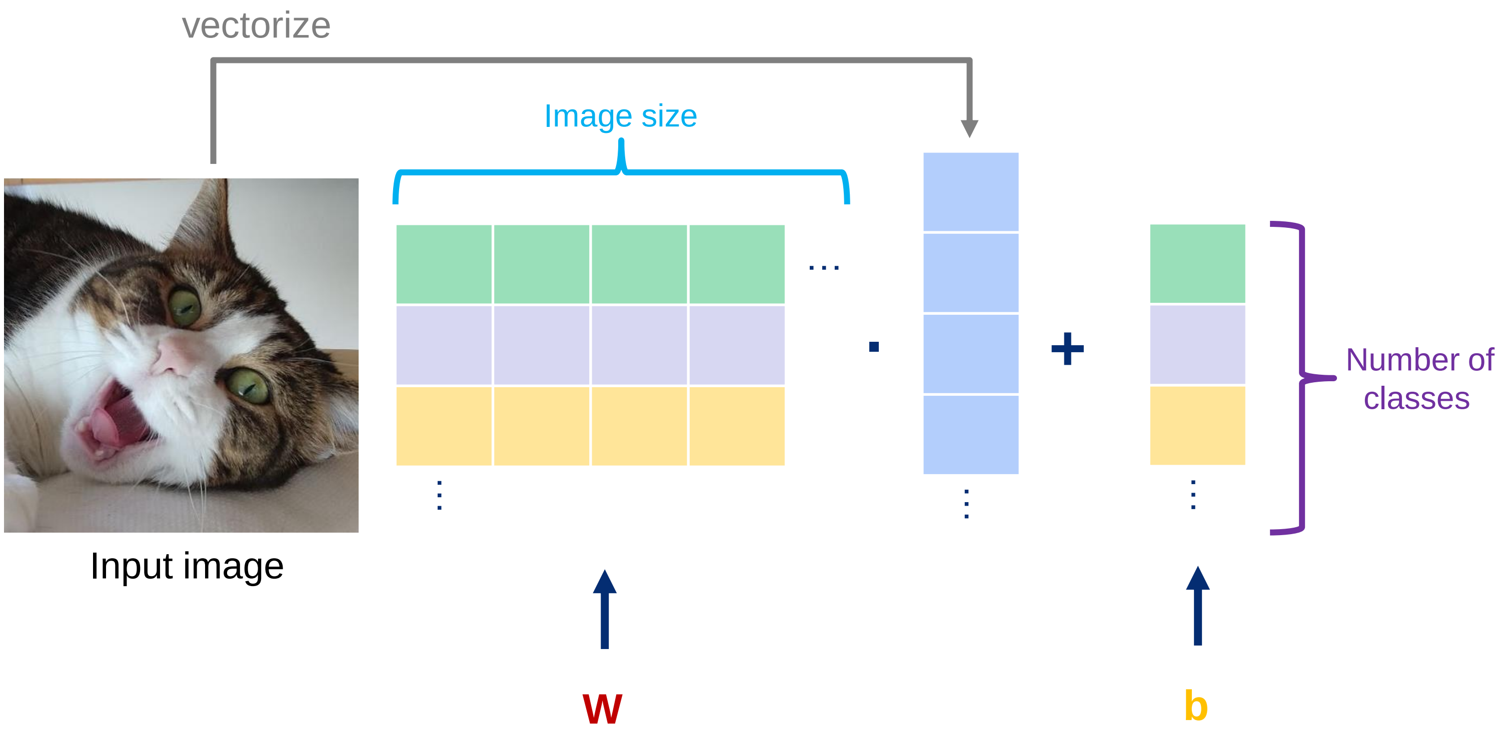
Bias term: constant!
→ Input independent preference

$$f(\mathbf{x},\mathbf{W})$$

W
Parameters or weights

10 numbers for class scores

Interpretation of Linear Classification in this Formalism

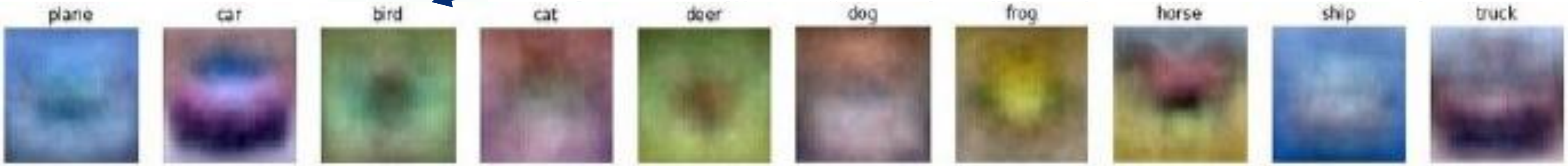


An Obvious Challenge

Seagull

Template assumption on pixel level is strong!

We need some form of abstraction
→ Enter: **Features**!

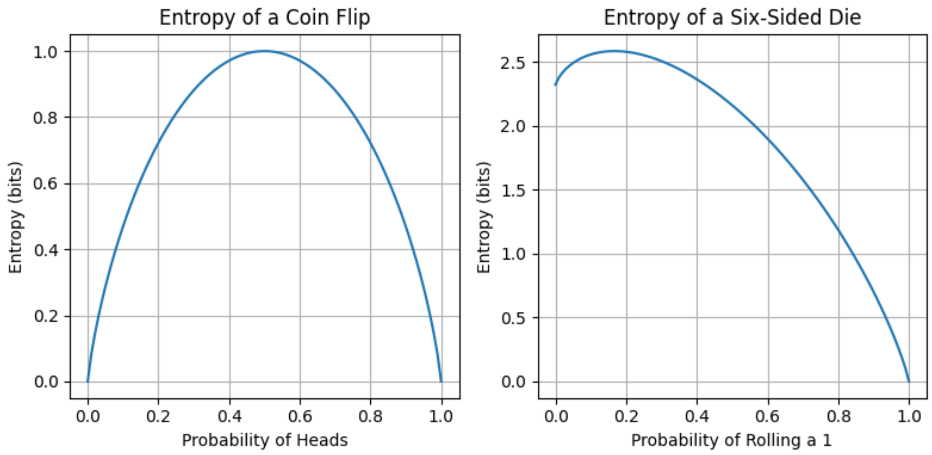


Entropy, Cross-Entropy, Relative-Entropy(KL div)

- **Entropy**: Measure of inherited randomness of a random variable

$$H(X) = -\mathbb{E}[\log p(X)] = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$$

  - Example: Coin flip
    - "Expected surprise of a coin flip" as a function of the coin's fairness
    - Maximal entropy at x= 0.5, i.e., a fair coin



Log base b:

b=2 → bits
b=e → nats
B=10 → dits

Entropy, Cross-Entropy, Relative-Entropy (KL div)

- **Cross-Entropy**: Measure of dissimilarity between two random variables

P – the “true” distribution
Q – the “predicted” distribution

$$H(P,Q) = -\mathbb{E}_{x\sim P}[\log Q(x)] = -\sum_{x\in\mathcal{X}} P(x)\log Q(x)$$

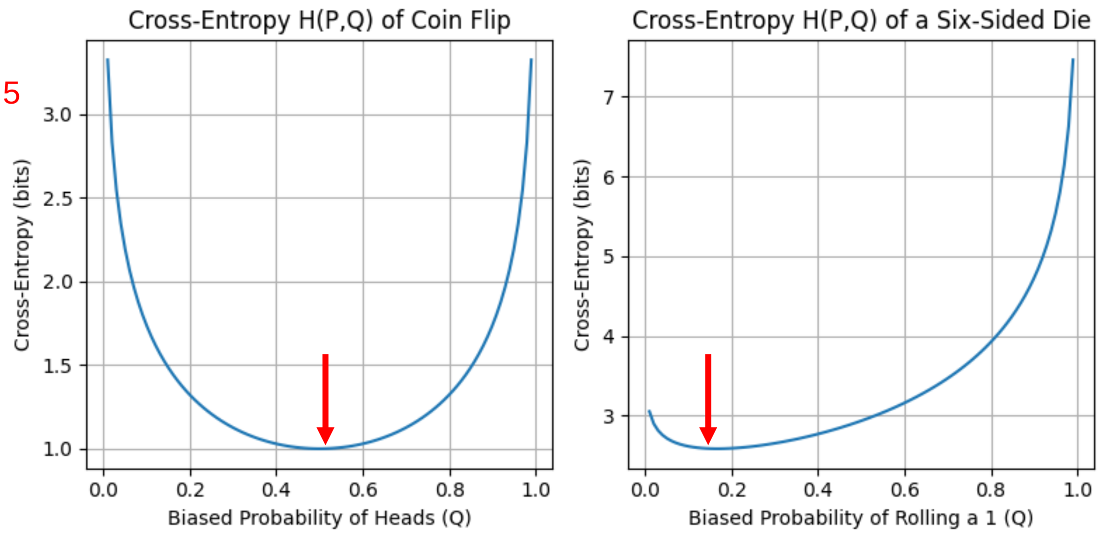
- Expected cost (bits) to encode data using the “predicted” distribution Q instead of “true” distribution P
- Minimal when P = Q, greater than 0 otherwise

Entropy, Cross-Entropy, Relative-Entropy (KL div)

- **Cross-Entropy**: Measure of dissimilarity between two random variables

$$H(P,Q) = -\mathbb{E}_{x\sim P}[\log Q(x)] = -\sum_{x\in\mathcal{X}} P(x)\log Q(x)$$

assume P(x)=0.5

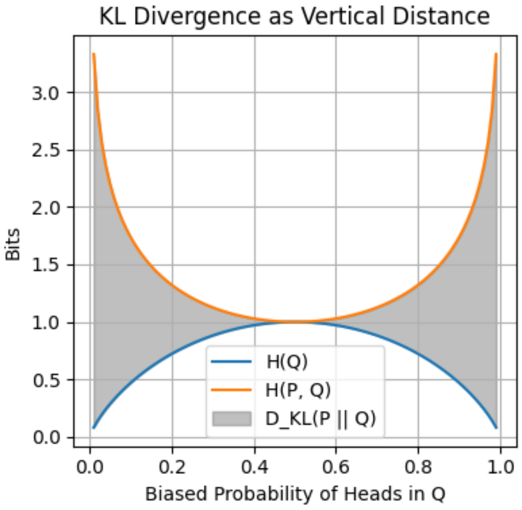


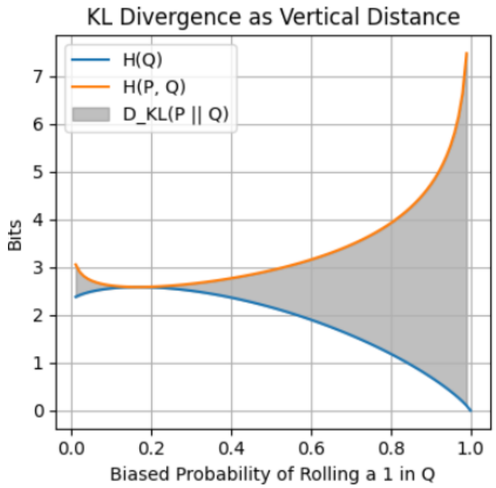
assume P(x)=1/6

Entropy, Cross-Entropy, Relative-Entropy (KL div)

- **Relative Entropy (KL divergence)**: Measures difference between probability distributions

$$KL(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)} = H(P,Q) - H(P)$$

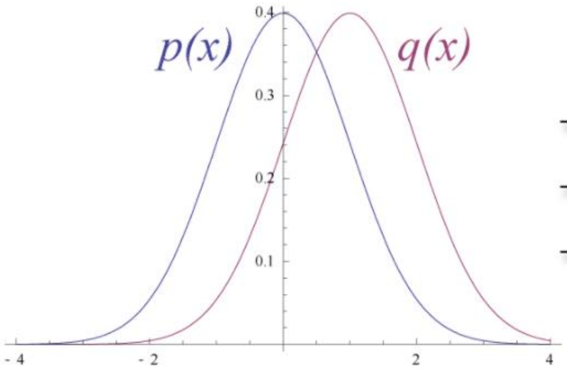




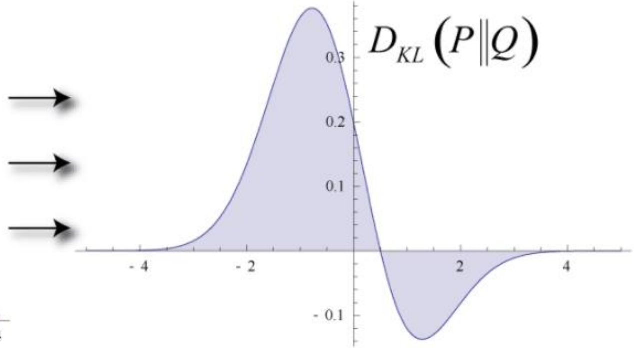
Entropy, Cross-Entropy, Relative-Entropy (KL div)

- **Relative Entropy (KL divergence)**: Measures difference between probability distributions
  - Not a metric (asymmetric and no triangle inequality)

$$KL(P||Q) = \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)} = H(P,Q) - H(P)$$



Original Gaussian PDF's



KL Area to be Integrated

Notes on Entropy, Cross Entropy, and Relative Entropy

Entropy: The uncertainty inherent in a random variable's possible outcomes

$$H(X) = -\mathbb{E}[\log p(X)] = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$$

Cross Entropy: "Average number of bits needed to encode data coming from a source with distribution p when we use model q"

$$H(P, Q) = -\mathbb{E}_{x \sim P}[\log Q(x)] = -\sum_{x \in \mathcal{X}} P(x) \log Q(x)$$

Relative Entropy (KL div): Measures difference between probability distributions

$$KL(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)} = H(P, Q) - H(P)$$

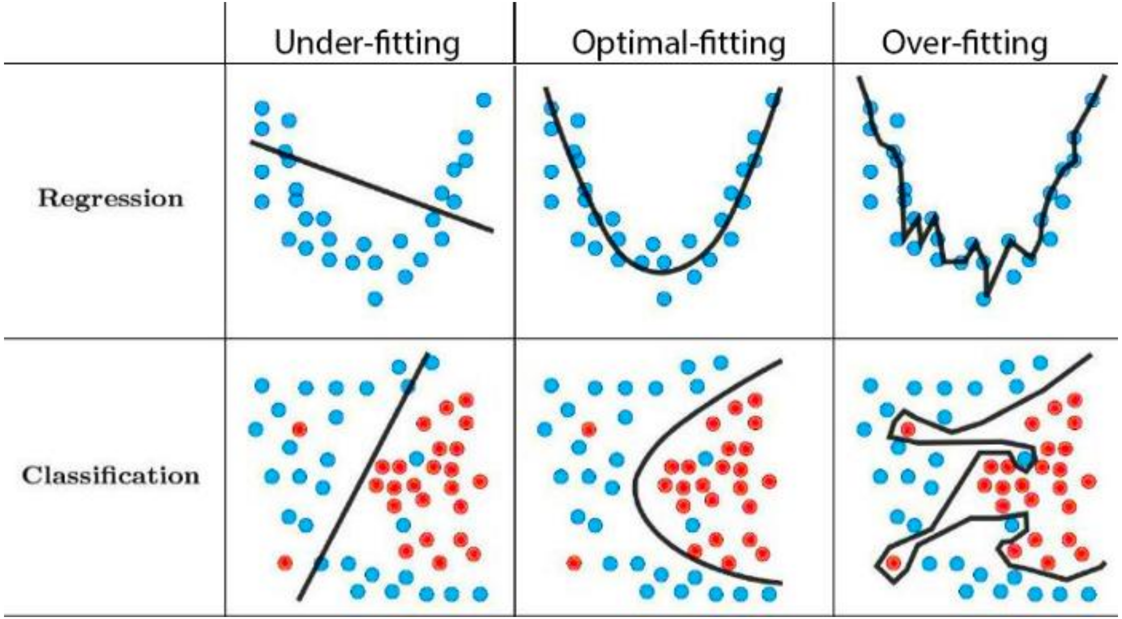
Recitation

Bias and Variance

What is overfitting?

Hastie, T., Tibshirani, R, and Friedman, J. (2017) The Elements of Statistical Learning

What is overfitting?



Remember:

Sampling process is random!

Samples for training/validation/testing are sampled from the same distribution but with different “unknowns”.

How to spot overfitting?

Approach 1: Use all data for training; test on, hum, nothing? → Bad!

Dataset

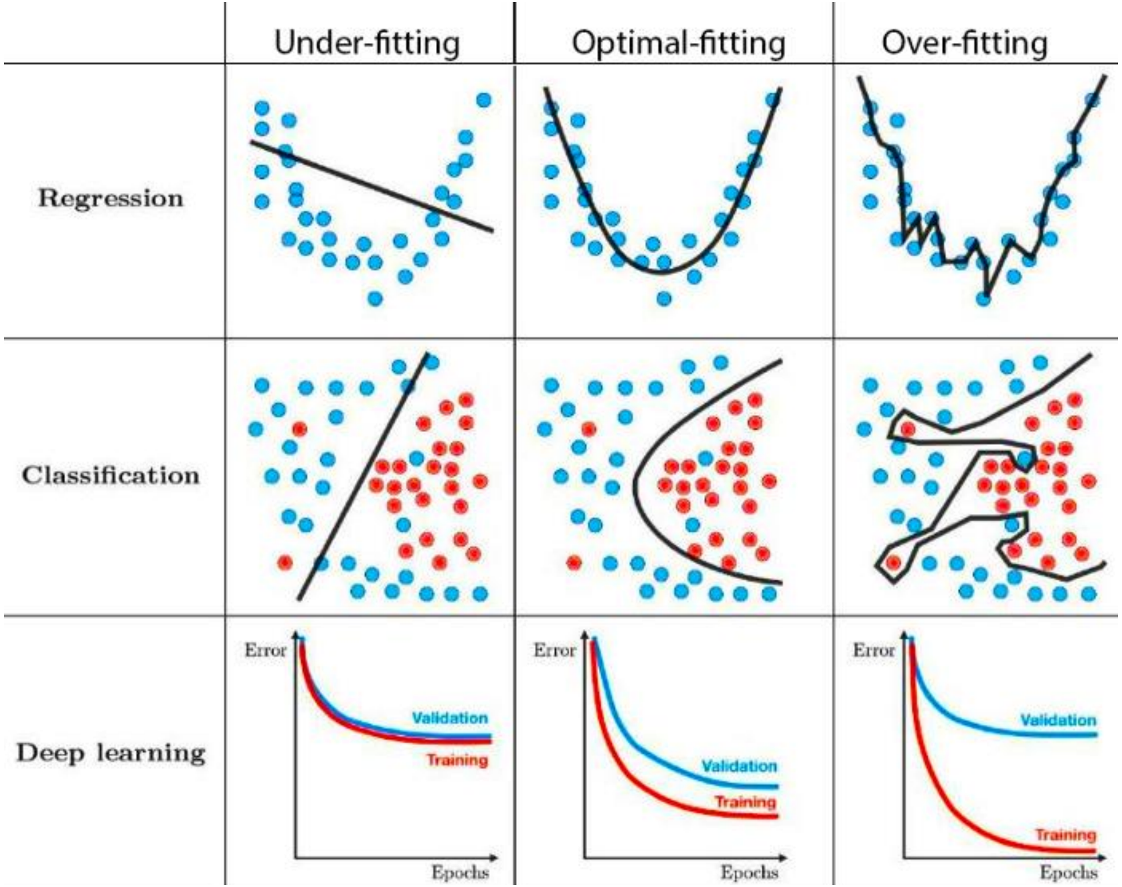
Approach 2: Split data into train and test; hyperparameters chosen to be best on test data. → Bad, no way to know how this will generalize to new data.

Train	Test

Approach 3: Split data into train, validation, and test; hyperparameters chosen on validation, then evaluated on test. → Better.

Train	Validation	Test

How to spot overfitting?



Remember:

Sampling process is random!

Samples for training/validation/testing are sampled from the same distribution but with different “unknowns”.

How to avoid overfitting?

Cause: “Over-explaining” association between in- and dependent variables
Solution: Reduce opportunity for model to overexplain

- Simpler models
- Regularization
- …

Hastie, T., Tibshirani, R, and Friedman, J. (2017) The Elements of Statistical Learning

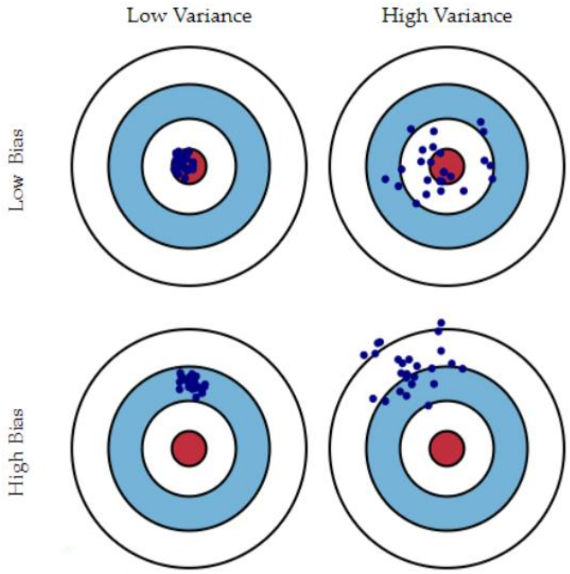
Bias Variance Tradeoff – Review

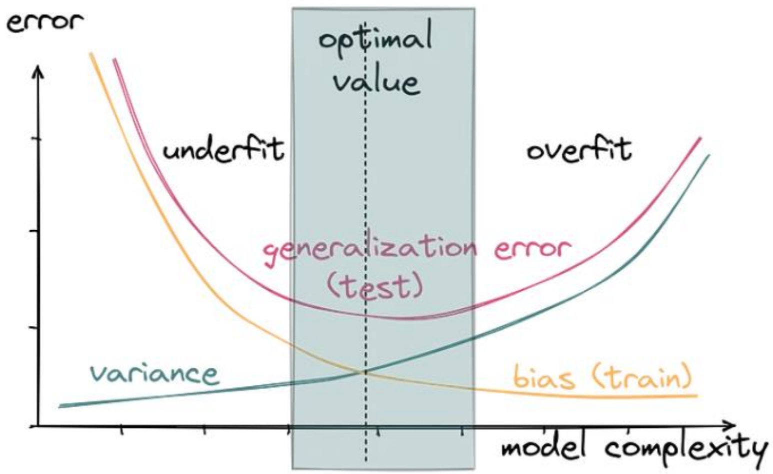
$$L(W) = \underbrace{(E[\hat{y}] - y)^2}_{\text{Bias}^2} + \underbrace{E[(\hat{y} - E[\hat{y}])^2]}_{\text{Variance}} + \sigma$$

Irreducible error

y : ground truth label

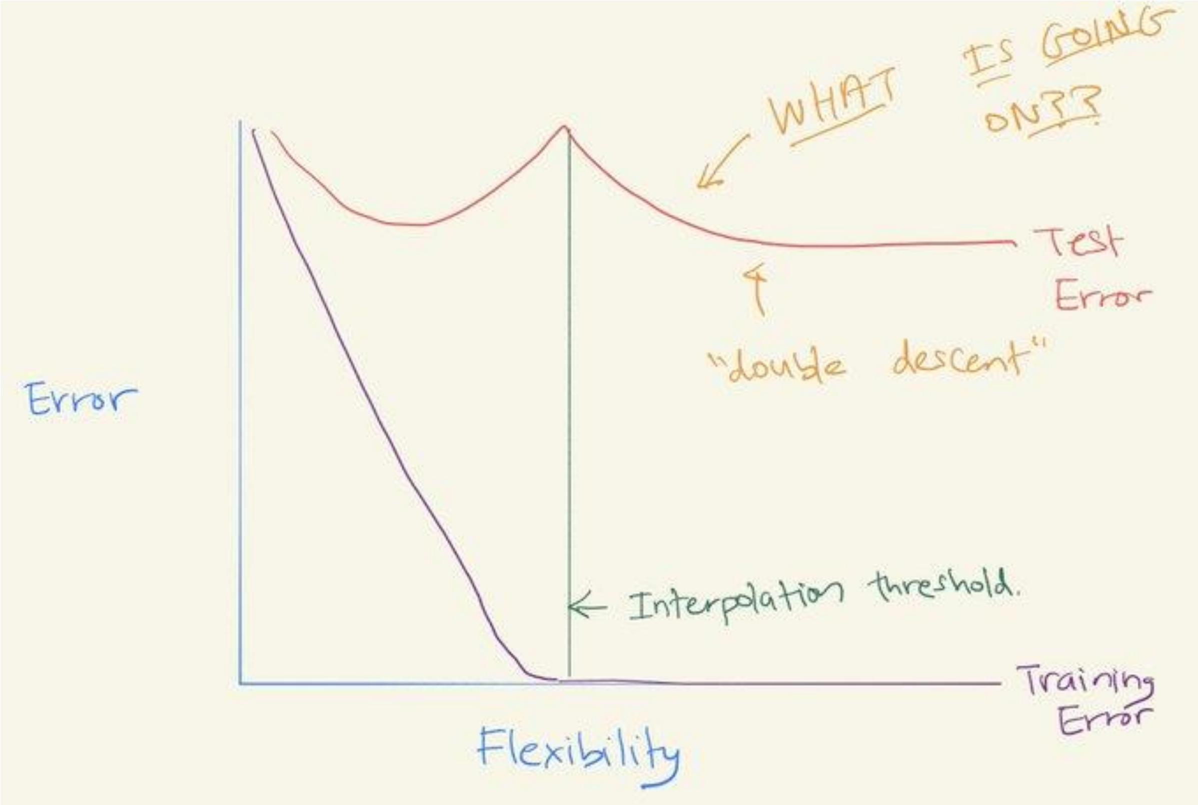
$\hat{y}$: predicted label





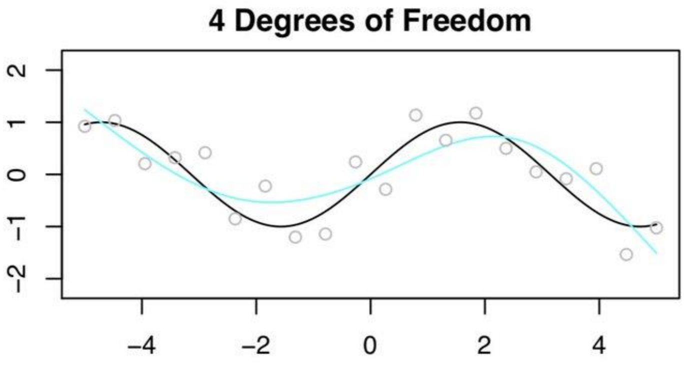
https://web.stanford.edu/~hastie/ElemStatLearn/
https://medium.com/@ivanreznikov/stop-using-the-same-image-in-bias-variance-trade-off-explanation-691997a94a54

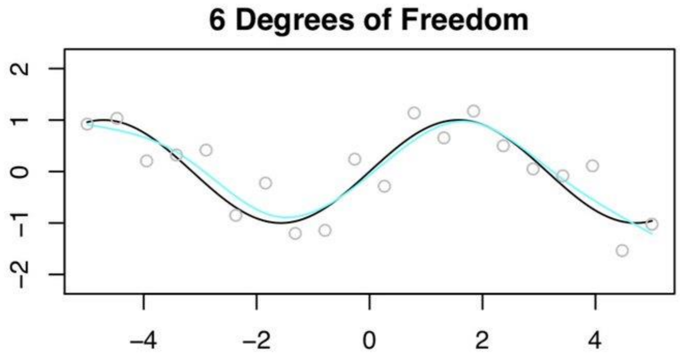
Bias Variance Tradeoff – Double Descent

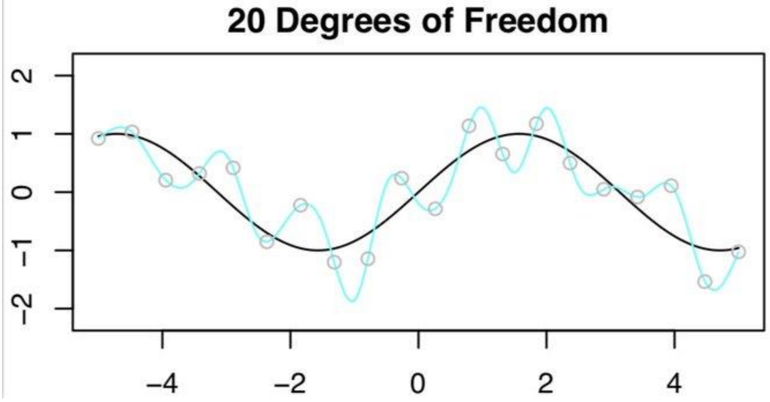


Based on material from this tweet: https://twitter.com/daniela_witten/status/1292293102103748609?s=20

Bias Variance Tradeoff – Double Descent

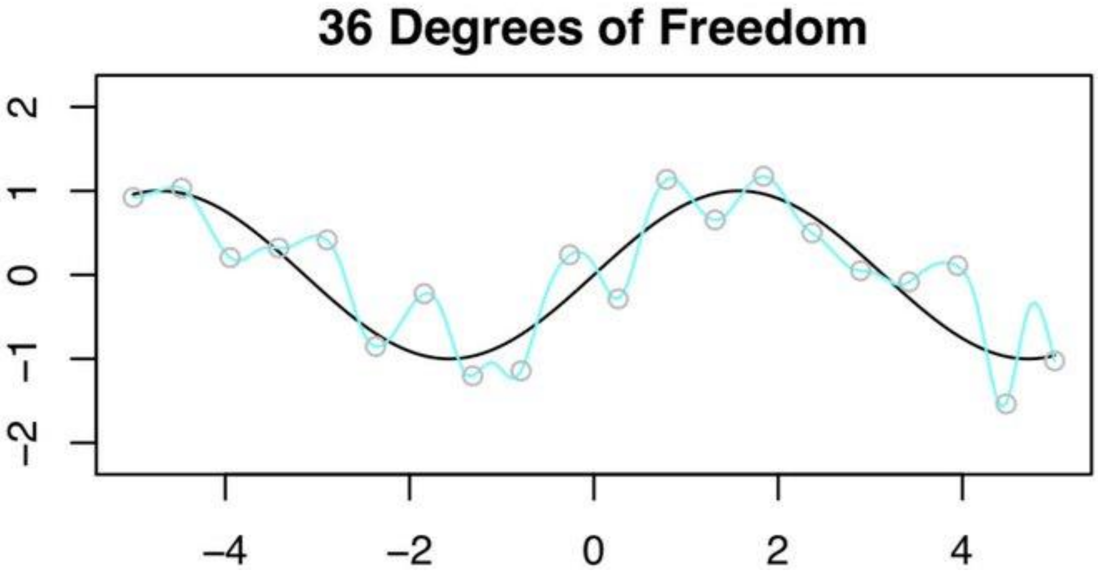






Based on material from this tweet: https://twitter.com/daniela_witten/status/1292293102103748609?s=20

Bias Variance Tradeoff – Double Descent

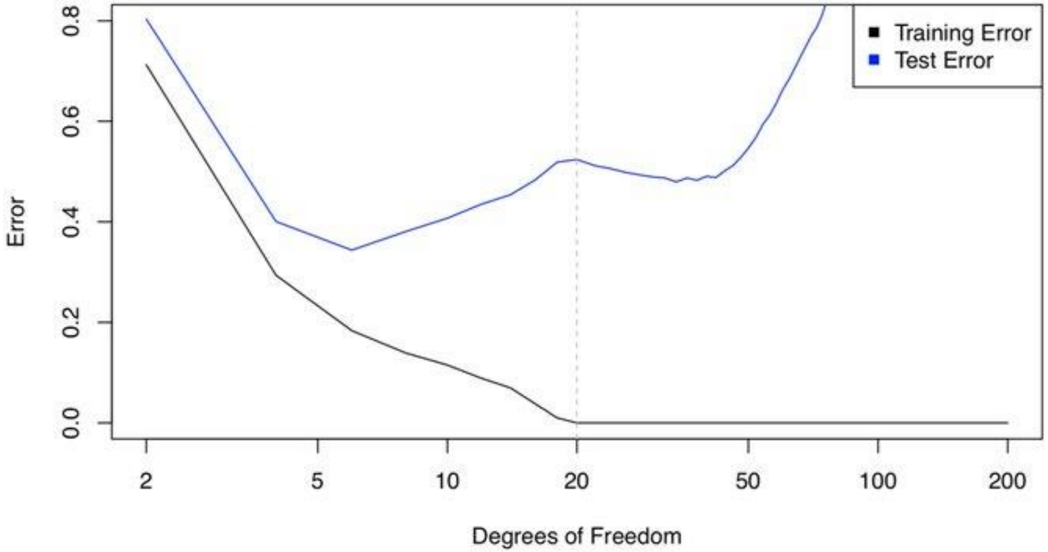


How can we understand this?

- Beyond the interpolation threshold, many solutions will do ~ equally well
- Regularization: Norm minimal solution
- This solution is “less wiggly”, better generalization

Based on material from this tweet: https://twitter.com/daniela_witten/status/1292293102103748609?s=20

Bias Variance Tradeoff – Double Descent

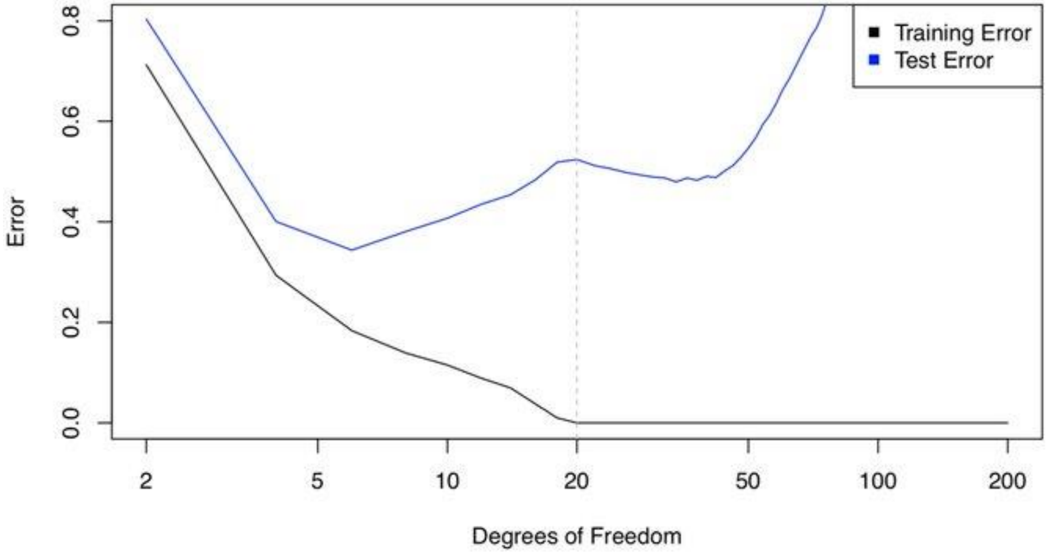


Double descent!

- Here, true process is precisely known (6 DoF spline)
- In reality, true process is unknown → Second descent may result in better solutions
- Hard to achieve in practice (interpolation threshold too high)

Based on material from this tweet: https://twitter.com/daniela_witten/status/1292293102103748609?s=20

Bias Variance Tradeoff – Early Stopping?



Bias Variance Tradeoff – so far: plots model error vs. model parameters

- So far here: plots model error vs. model parameters
- Often also used to explain the need for early stopping – plots model error vs. training “time”
- How is this related?

Based on material from this tweet: https://twitter.com/daniela_witten/status/1292293102103748609?s=20