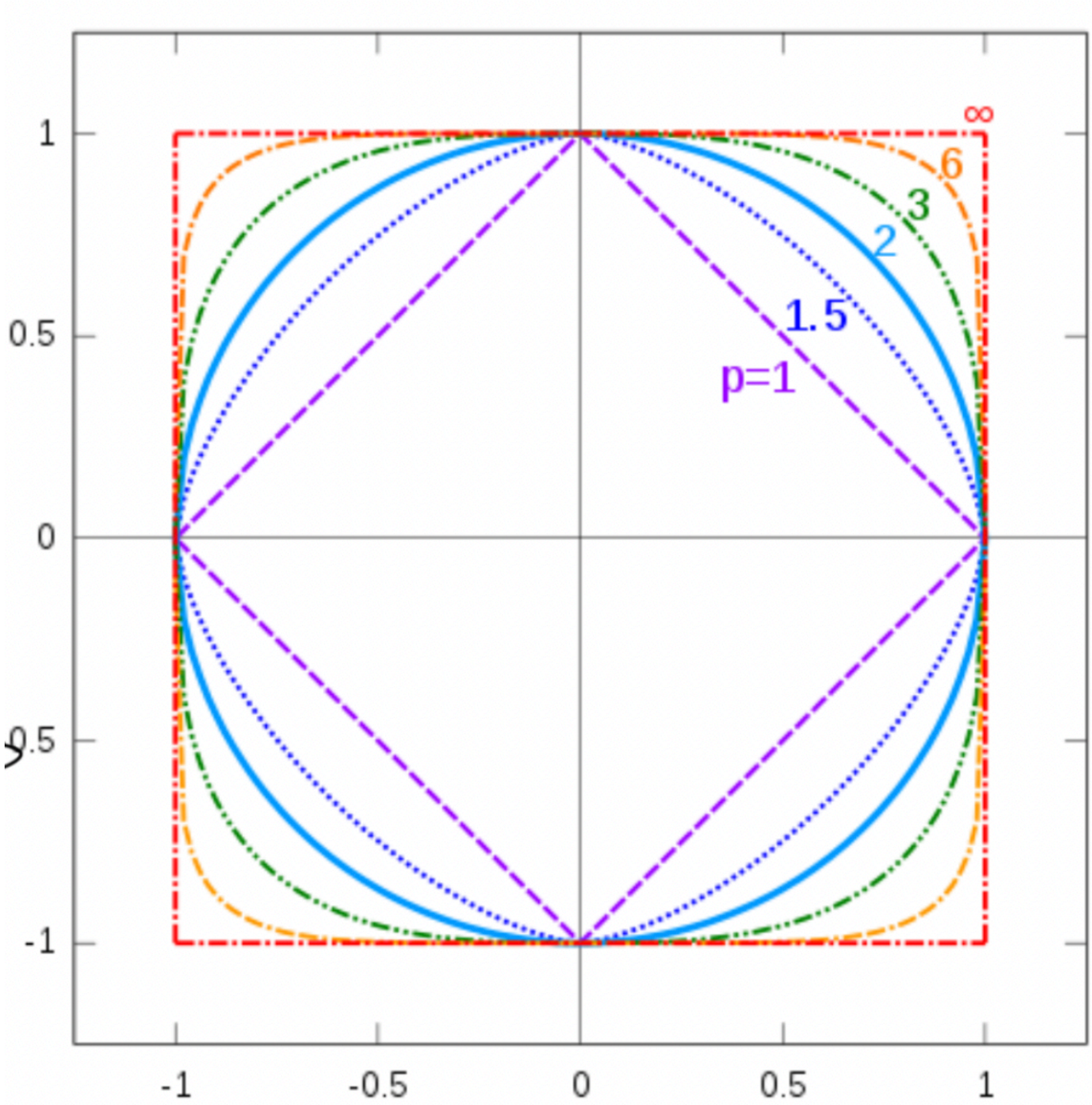
Review

1. Ordinary Least Square

- $\hat{y} = Xw$
- the intuition behind linear regression (when you should and should not use it)
  - noise: homoscedasticity (~normal distribution)
  - random sampling
  - independent variables (no collinearity)
- OLS: an analytical / closed-form solution exists.
  - $L_p: ||x||_p = (x_1^p + x_2^p \ldots x_n^p)^{1/p}$

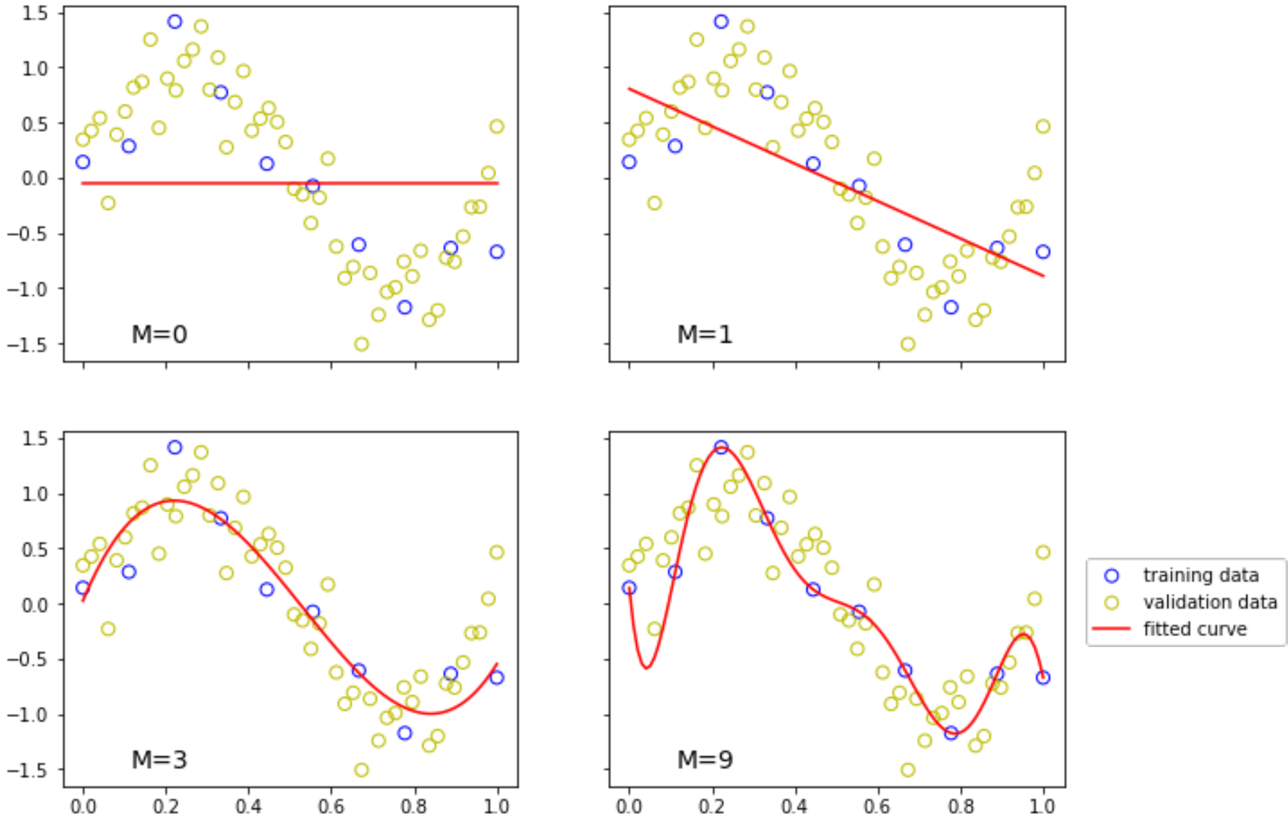


- Condition: convex & differentiable
- $\frac{\partial L(w)}{\partial w} = 0$, where $L(w) = e^T e$, $e = y - \hat{y}$
- $W* = (X^T X)^{-1} X^T y$
- to compute the inverse: `numpy.linalg.inv`

2. Bias-Variance tradeoff

- Purpose: model comparison
- Total error: $E(y_0 - \hat{f}(x_0))^2 = \mathrm{Var}(f(\hat{x}_0)) + [\mathrm{Bias}(f(\hat{x}_0))]^2 + \mathrm{Var}(\epsilon)$
    - Variance: How much does the model vary due to random training samples?

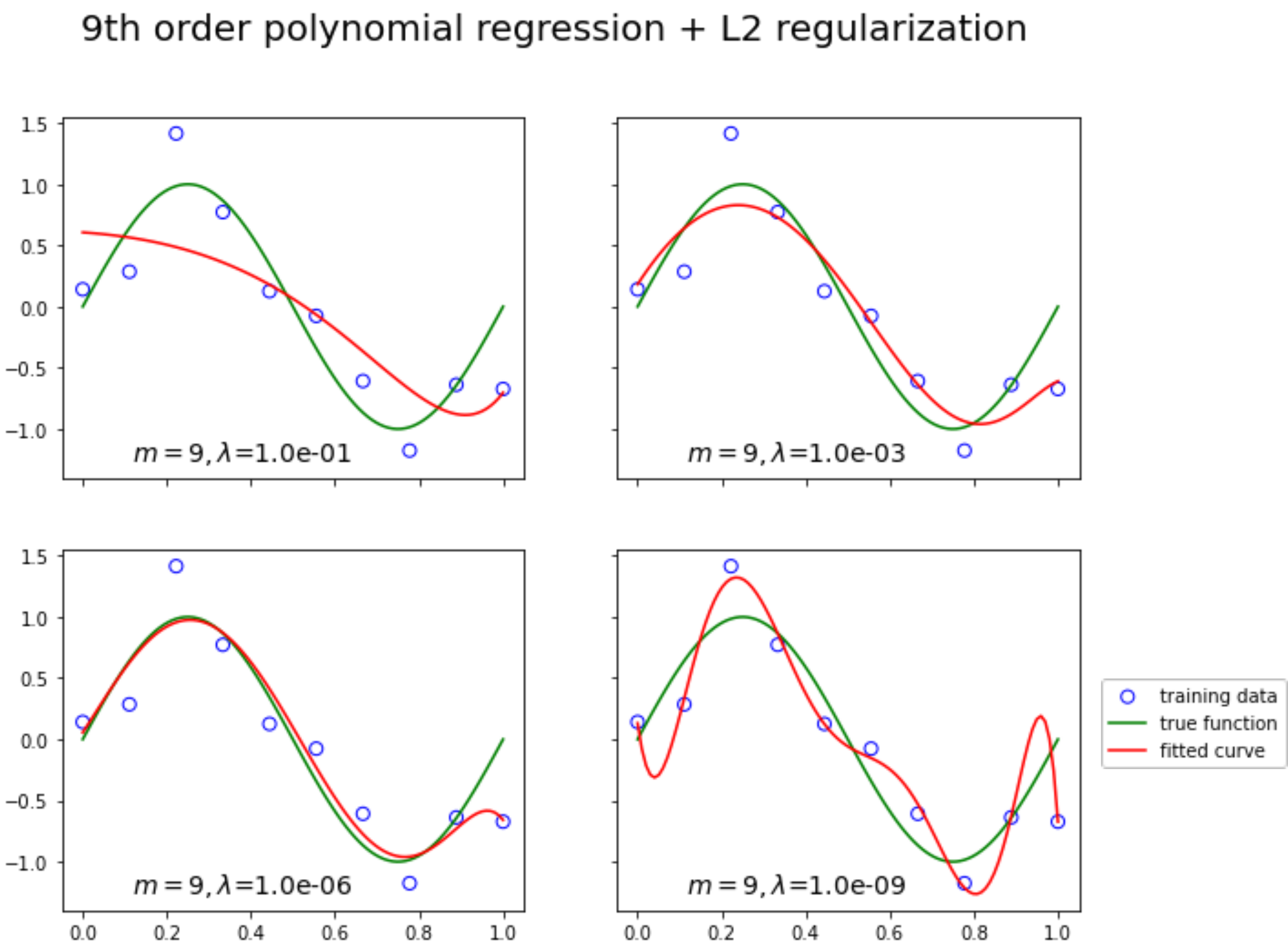
- Bias: How far off would the model be, even if we have infinitely many samples?
- Noise: irreducible error.



- Training, validation and test set

3. Regularizations

- to mitigate overfitting
- $L(w) = (Xw - y)^T(Xw - y) + \frac{\lambda}{2}||w||_2$



- Closed-form solution: $w^* = (X^T X + \lambda I)^{-1} X^T y$
- when L_1 regularization is good?

4. Robust estimation

- using regularization or L1 loss function
- Gradient descent:
  - $\frac{\partial L(w)}{\partial w} = \text{sign}(y - \hat{y})x$
  - $w_{t+1} = w_t - \lambda_t \frac{\partial L(w)}{\partial w}$

Discussion Questions

1. monotonicity

We define a loss function:

$$L(w) = \sum_{i=1}^{n} |y_i - wx_i| \tag{1}$$

Let's assume that we found: $w_a = \arg\min_w L(w) = 3.14$.

If the optimal w is denoted as $w^* = \arg\min_w [118 + \ln(L(w)) - (\theta - 21)^3]$, where θ is a constant. What is the value of $w*$?

Hints:

- $\min$ and argmin
- Monotonicity (how to prove?)
- $\log$, $\exp$, inverse, sigmoid ($S(x) = 1/(1+e^{-x})$), cumalative probability ($P(X \leq x)$)

2. gradient descent

2.1 intuition: Why do we want to use this iterative process to calculate the critical value of a function instead of analytically calculating the minimum/maximum value, like OLS?

Hint: when closed-form solution is available?

2.2: Consider the following function

$$f(x, y, z) = 3x^3y^2z + 4yz \tag{2}$$

Suppose that a single iteration of gradient descent is run on this function with a starting location of $(x, y, z) = (1, 1, 1)$ and a learning rate of 0.1. What is the new value of (x, y, z) after one iteration? Solve by computing the gradient of f w.r.t. (x, y, z).

$$\begin{aligned}
\nabla f(x, y, z) &= (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}) \\
&= (9x^2y^2z, 6x^3yz + 4z, 3x^3y^2 + 4y) \\
\nabla f(x_t, y_t, z_t) &= (9, 10, 7) \\
(x_{t+1}, y_{t+1}, z_{t+1}) &= (x_t, y_t, z_t) - \lambda \nabla f(x_t, y_t, z_t) \\
&= (1, 1, 1) - 0.1 * (9, 10, 7) \\
&= (0.1, 0, 0.3)
\end{aligned} \tag{3}$$

2.3 In this section, we will compute the gradient of the loss with respect to w_0 and w_1 by iterating through a for loop. In practice, we compute gradients with vectorized code, which is what we ask of you in A2. Here, we will walk through a basic example to build the intuition behind gradient descent.

```
def gradient(w0, w1):
    """
```

```
    Compute the L1 gradient of the loss function.
    """
    w0_grad, w1_grad = [0, 0]
    for xi, yi in zip(x, y):
        loss = w0 + w1 * xi - yi #? y_hat - y_actual
        if loss > 0: # sign(y_hat - y_actual) > 0
            w0_grad += 1 #?
            w1_grad += xi #?
        else:
            w0_grad -= 1 #?
            w1_grad -= xi #?
    return w0_grad, w1_grad
```

Hint: for L_1 norm, $\frac{\partial L(w)}{\partial w} = \text{sign}(y - \hat{y})x$

Run the code in the notebook to see how fitting changes over time

3. Polynomial Regression

Question: Suppose you are working on a polynomial linear regression problem with one input feature x, and the model is to be trained using a polynomial of degree 3. Construct a design matrix for the following dataset:

x	y
1	2
2	5
3	10
4	17

In polynomial regression, we can use a polynomial function of the input feature to model the relationship between the input and output variables. For a polynomial of **degree 3**, the model equation can be written as:

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 \tag{4}$$

Solution:

1	x	x^2	x^3
1	1		
1	2		
1	3		
1	4		

```
x = np.vstack([np.ones(x.shape), x,x**2, x**3])
```

Assignment 2

Q 1

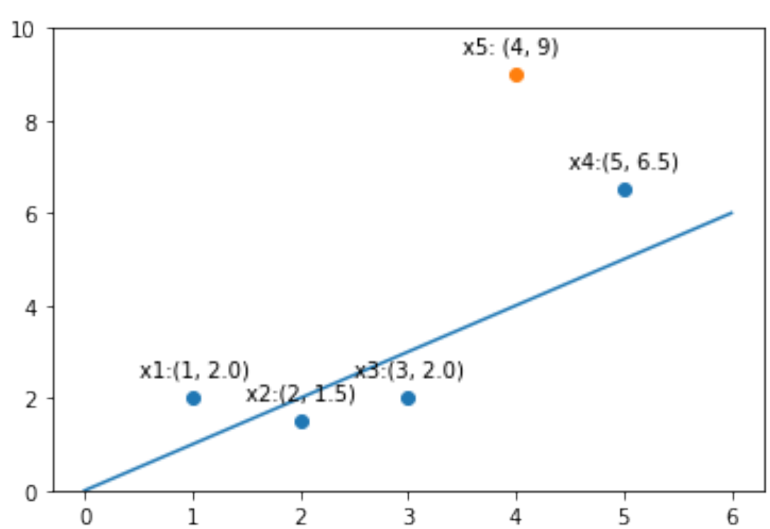
- see the first discussion question

Q 2

- check lecture 4 notes (and the start of lecture 5)

Q 3

- Q 3.1
    - what should be X and W?
    - what algorithm should be used?
- Q 3.2
    - what algorithm should be used?
- Q3.3:
    - How the mixing rate α affect curve fitting?
        - what if the loss function when $\alpha = 0$?
        - what if the loss function when $\alpha = 1$?
        - L1 and L2 norm, which is more robust against 'outliers'? What are the outliers in the given data points? How does α relate to the 'robustness'?
    - consider a toy dataset as follows, where the true generative model is $y = x$



	$\sum_{i=1}^{4} e_i$	e_5
L1	$\sum_{i=1}^{4} \lvert y_i - \hat{y}_i \rvert = 4$	$\lvert y_5 - \hat{y}_5 \rvert = 5$
L2	$\sum_{i=1}^{4} (y_i - \hat{y}_i)^2 = 4.5$	$(y_5 - \hat{y}_5)^2 = 25$

how do you compare $\sum_{i=1}^{4} e_i$ and e_5? What is their effect on curve fitting?