Nonlinear features, Regularization

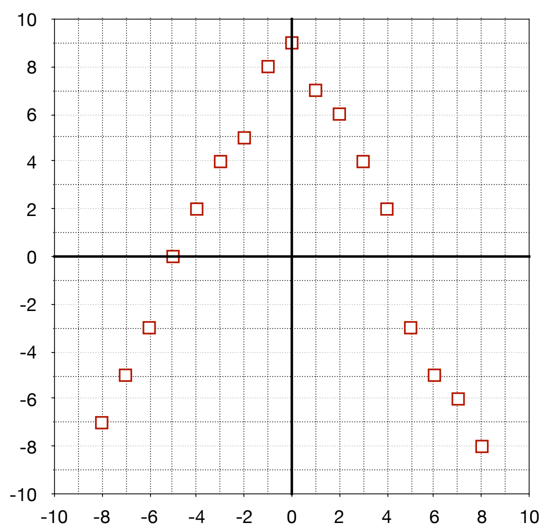
CSC 461: Machine Learning

Fall 2022

Prof. Marco Alvarez
University of Rhode Island

Nonlinear features

Data is not always 'linear'



Transforming the data

- Linear regression => **linear in the weights**
  - ✓ linear combination of the features
- Nonlinear functions
  - ✓ can transform the data nonlinearly using any feature transformations

$$\mathbf{x} = (x_0, \ldots x_d) \quad \xrightarrow{\Phi} \quad \mathbf{z} = (x_0, \ldots z_{\tilde{d}})$$

input space $\mathcal{X} = \mathbb{R}^{d+1}$ feature space $\mathcal{Z} = \mathbb{R}^{\tilde{d}+1}$

Transforming the data

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \qquad \mathbf{\Phi}(\mathbf{x}) = \mathbf{z} = \begin{bmatrix} 1 \\ \Phi_1(\mathbf{x}) \\ \vdots \\ \Phi_{\tilde{d}}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ \vdots \\ z_{\tilde{d}} \end{bmatrix}$$

$$h(\mathbf{x}) = \tilde{\mathbf{w}}^T \mathbf{\Phi}(\mathbf{x})$$

Polynomial models on one feature

- A **k-th** order polynomial model in one variable is defined as:

$$h(\mathbf{x}) = w_0 + w_1 x^1 + w_2 x^2 + \ldots + w_k x^k$$

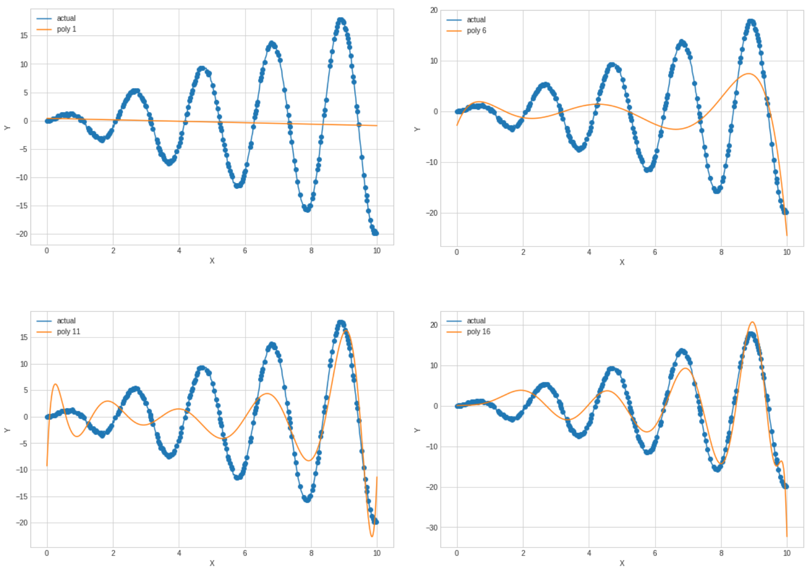
$$\mathbf{x} = \begin{bmatrix} 1 \\ x \end{bmatrix} \qquad \mathbf{\Phi}(\mathbf{x}) = \mathbf{z} = \begin{bmatrix} 1 \\ \Phi_1(\mathbf{x}) \\ \vdots \\ \Phi_k(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 1 \\ x^1 \\ \vdots \\ x^k \end{bmatrix}$$

Polynomial models on two features

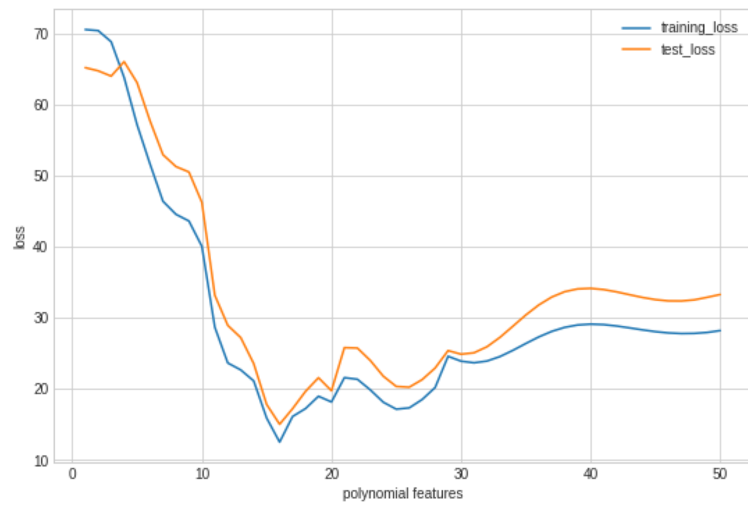
$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \qquad \mathbf{\Phi}(\mathbf{x}) = \mathbf{z} = \begin{bmatrix} 1 \\ \Phi_1(\mathbf{x}) \\ \Phi_2(\mathbf{x}) \\ \Phi_3(\mathbf{x}) \\ \Phi_4(\mathbf{x}) \\ \Phi_5(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}$$

Show me the code

```
# this function also adds the column of +1s
poly = PolynomialFeatures(p)

# transform data
_xtr = poly.fit_transform(Xtr)
_xte = poly.fit_transform(Xte)

# linear regression
w = np.linalg.pinv(_xtr).dot(Ytr)

# record losses
train_loss = np.mean((_xtr.dot(w)-Ytr)**2)
test_loss = np.mean((_xte.dot(w)-Yte)**2)
```



Trying a few transformations

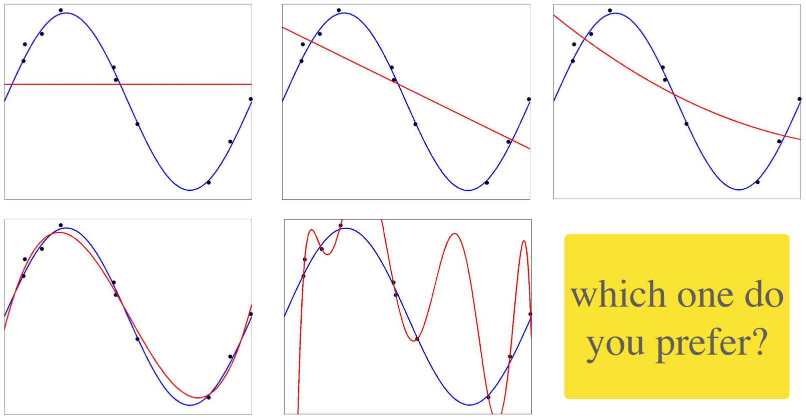


Polynomial models on more features

- `PolynomialFeatures` from *scikit-learn*
  - ✓ "all polynomial combinations of the features with degree less than or equal to the specified degree"
- Transformation function can be anything
  - ✓ choose transformation **before** looking into the data
  - ✓ use **cross-validation**
  - ✓ be aware of **computational cost**
  - ✓ be aware of **overfitting**

Overfitting and Regularization

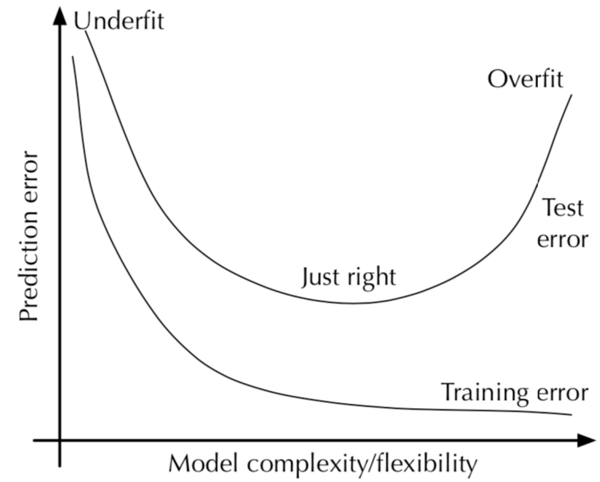
Lets talk about overfitting



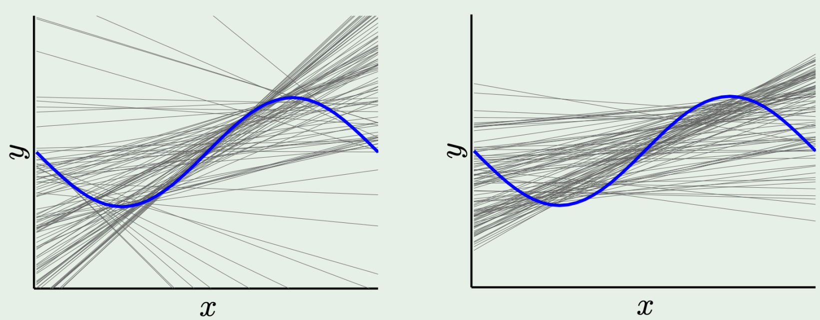
which one do you prefer?

Underfitting: model is too simple
Overfitting: model is too complex

Model complexity



What if we restrict the hypothesis space?



without regularization

with regularization

Regularization

- Adding a **penalty** to the weights to control the complexity of the model
  - ✓ usually penalizing higher weights (**except intercept**)
  - ✓ results in **simpler** or **more sparse** solutions
- Impact of regularization can be controlled by a parameter (*lambda*)

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \quad \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda R(\mathbf{w})$$

L2 regularization

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \quad \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

a.k.a. Ridge Regression

Can solve using matrix calculus again:

$$\mathbf{w}^* = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

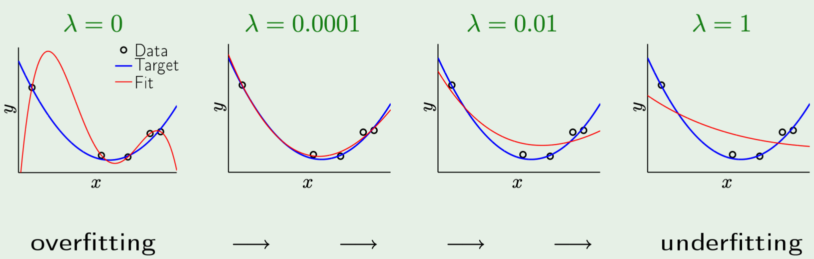
always invertible

L2 regularization

- If using the closed form solution for regularization the top-left corner of the identity matrix can be set to 0 (to handle intercept)

$$\begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

How does it work?



http://work.caltech.edu/slides/slides12.pdf

L1 regularization

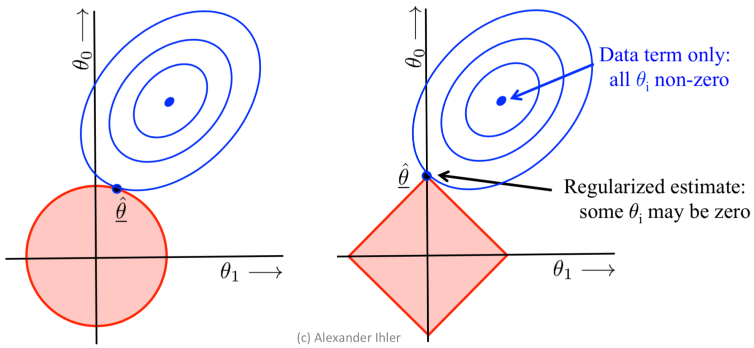
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \quad \frac{1}{n}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_1$$

a.k.a. Lasso Regression

- Lasso does not have a **closed form solution**
  - can solve with quadratic programming or variants of gradient descent (subgradient methods)
- The regularization term is not differentiable

Comparison

- L1 regularization tends to generate **sparser** solutions



Final remarks

- Linear regression
  - ✓ solved by defining a hypothesis space and a loss function
  - ✓ essentially an optimization problem that can be solved directly (closed-form) or using other techniques, such as, gradient descent
- Important concepts
  - ✓ nonlinear features
  - ✓ overfitting/underfitting
  - ✓ regularization

Colab notebook

https://colab.research.google.com/drive/1W9kR_cbjYw0Ek2rsTO7_ojbfzxVN3pSJ#scrollTo=Wlm7SPzqhWnP