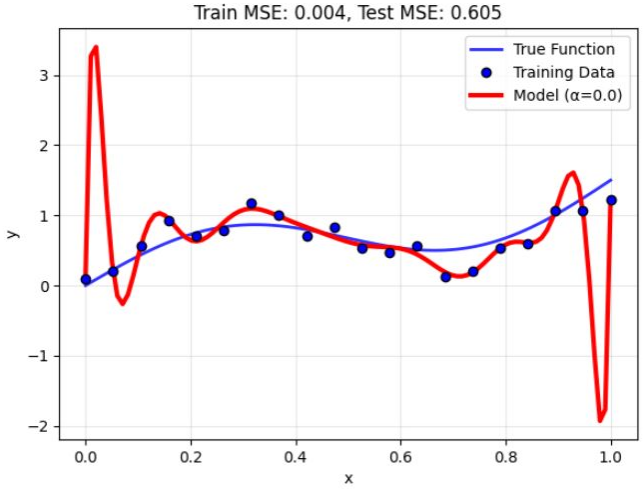
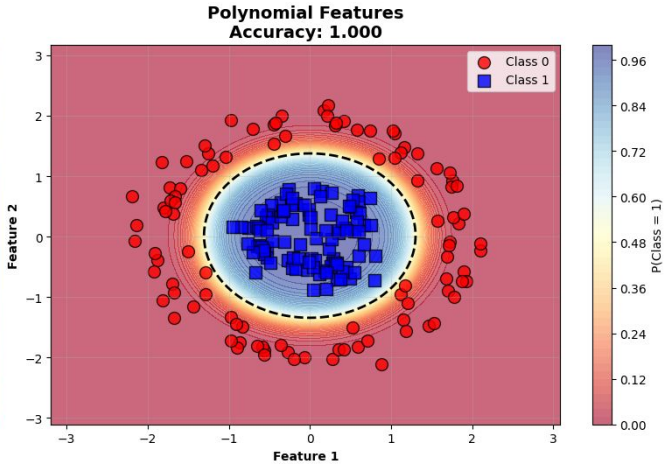
NYU CS-GY 6923 Machine Learning

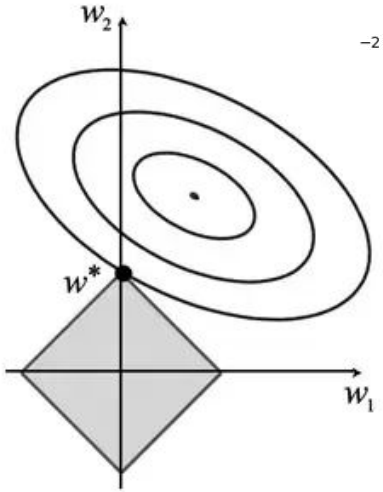
Prof. Pavel Izmailov

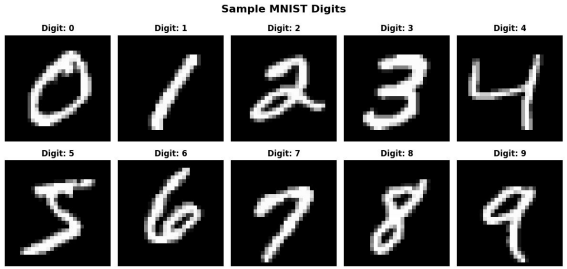
Today

- Recap overfitting and underfitting
- Regularization
- Classification and logistic regression

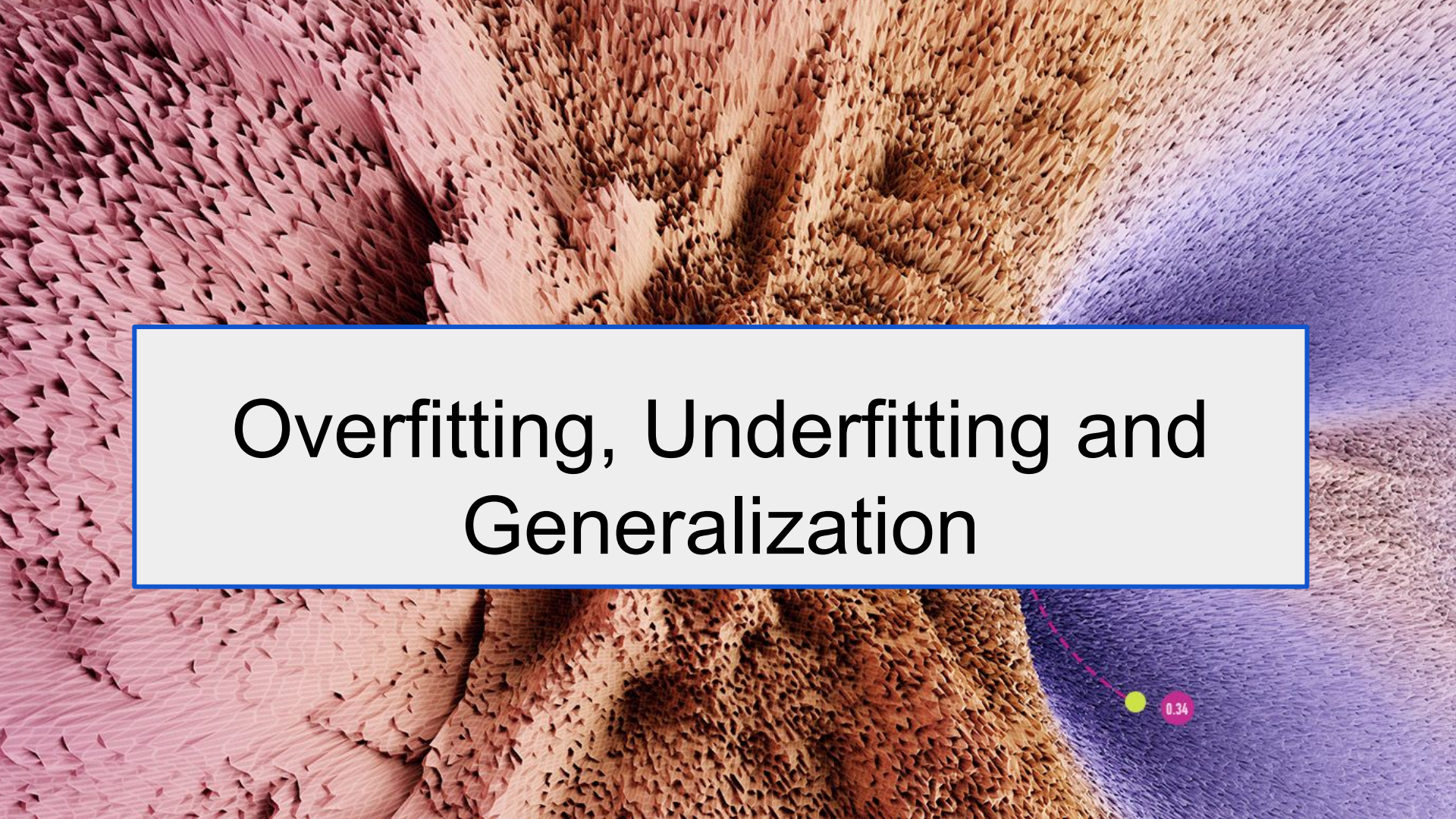








Overfitting, Underfitting and Generalization



Overfitting and underfitting

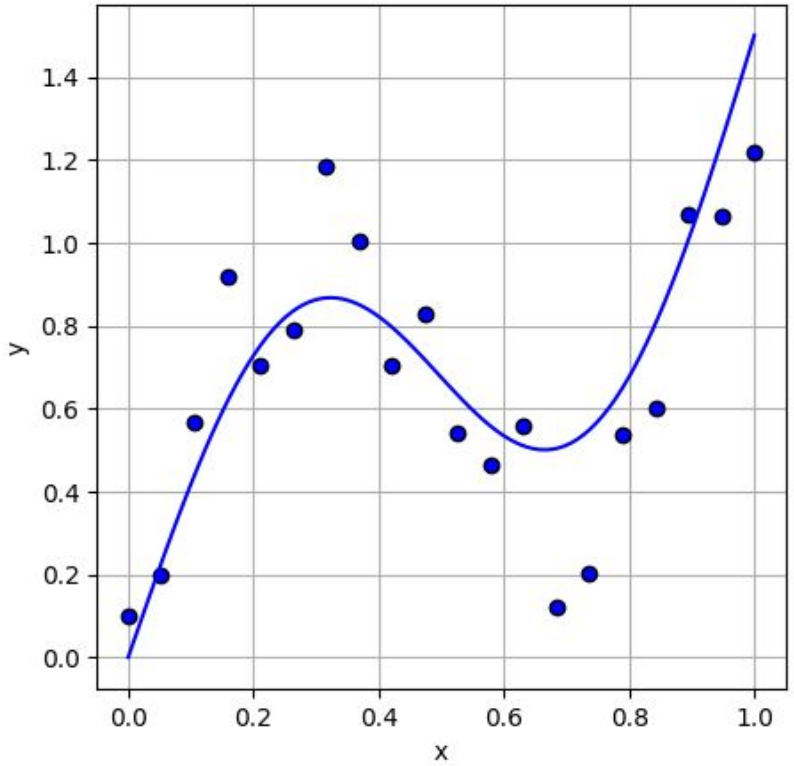
- Data generated by

$$y_{\text{true}} = 1.2 \cdot x + 0.5 \cdot \sin(2\pi x) + 0.3 \cdot x^2$$

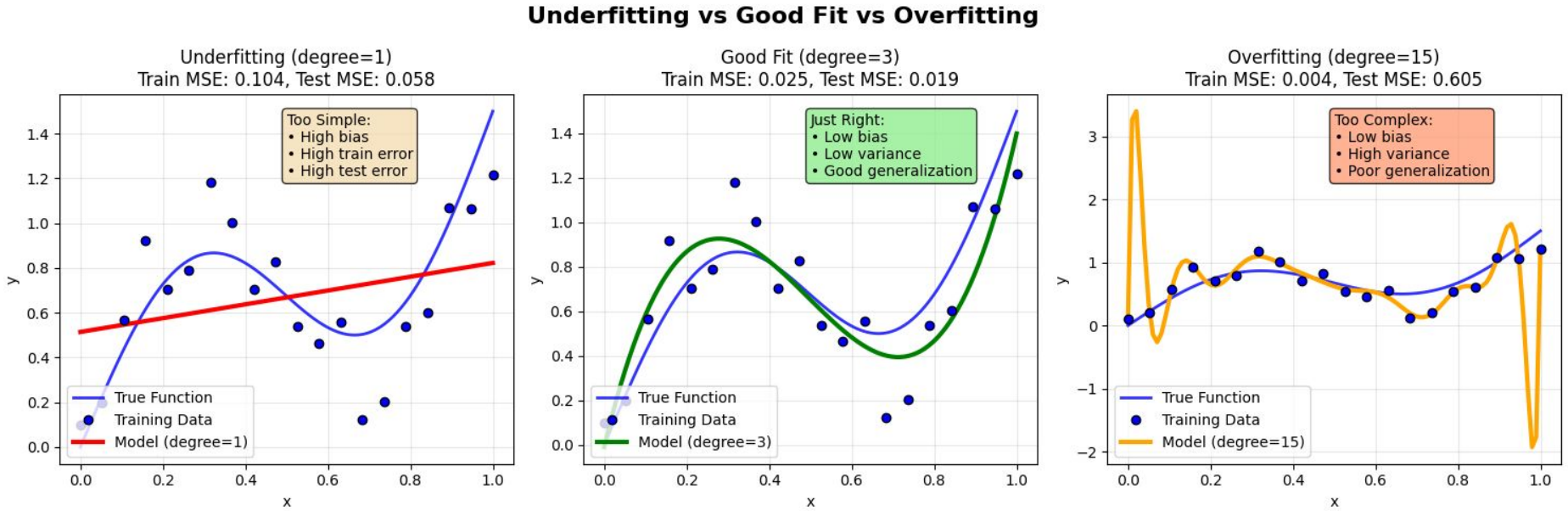
$$y = y_{\text{true}} + \varepsilon, \quad \varepsilon \sim N(0, 0.2^2)$$

- Polynomial features

$$[1, x, x^2, \ldots, x^d]$$

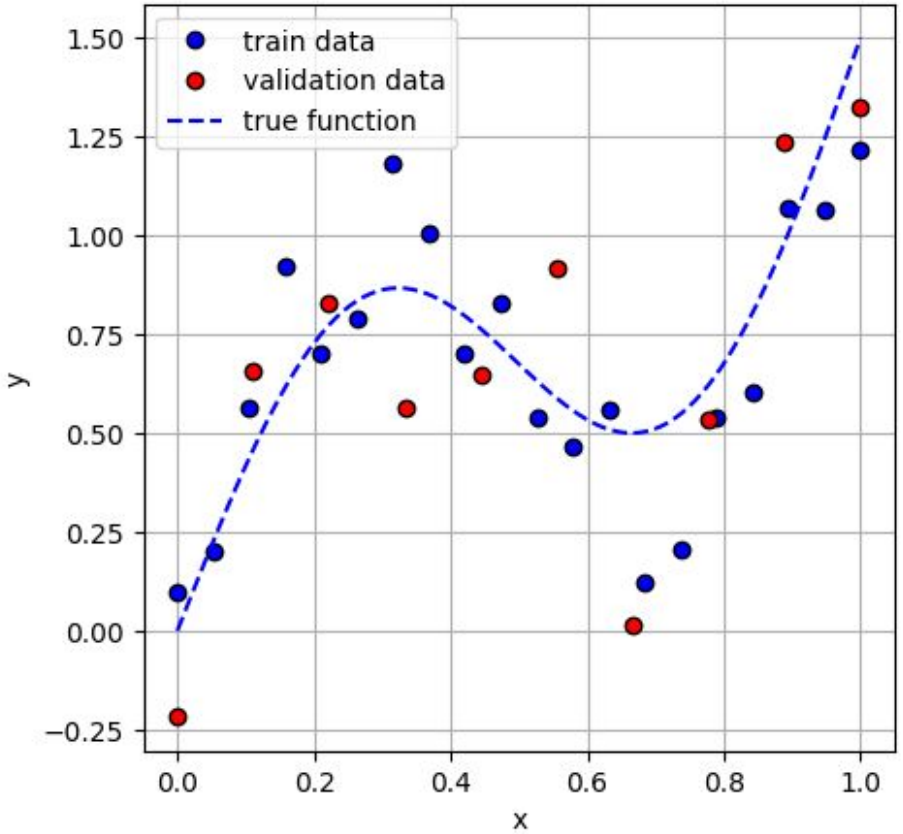


Overfitting and underfitting

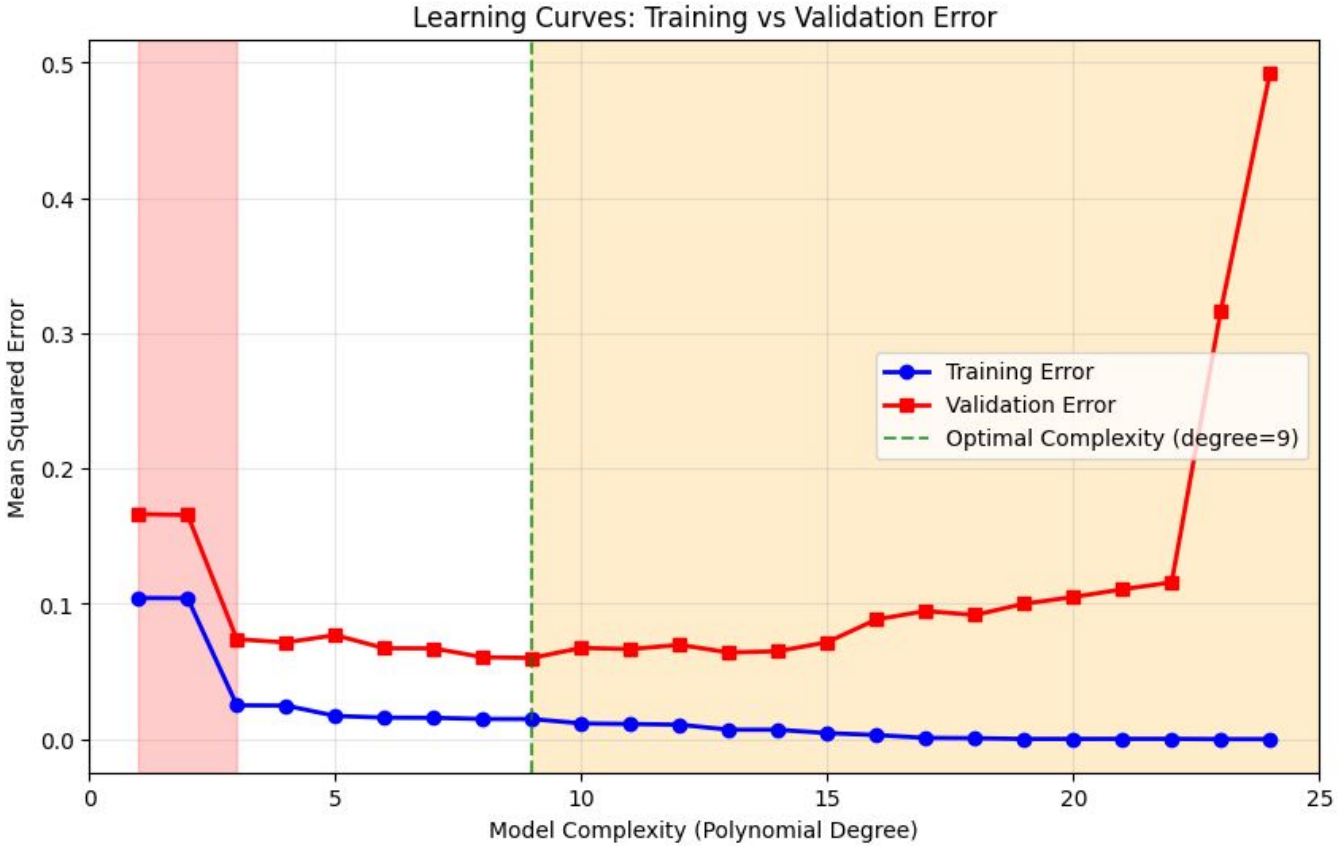


Validation

- We don’t have access to the true test loss in practice
- Leave some of the train data as a held-out validation set
- Use validation loss as a proxy for generalization



Model selection

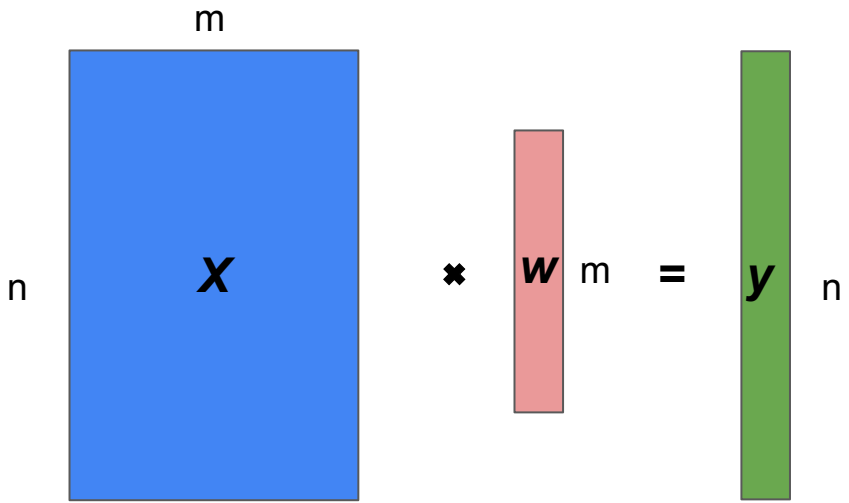


Connection to systems of linear equations

- We can view fitting a linear model as solving a system of linear equations

$$Xw = y \quad \Leftrightarrow \quad Ax = b$$

- What determines when we can solve a linear system?



Connection to systems of linear equations

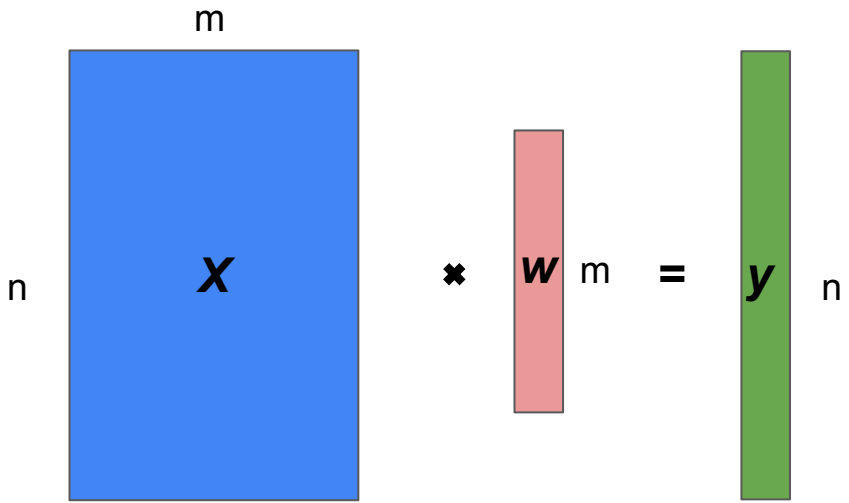
$$Xw = y \quad \Leftrightarrow \quad Ax = b$$

- If n > m and X full-rank then generally no solution
- Instead, we solve a least-squares problem

$$\min_w \|y - Xw\|^2$$

- Solution is given by the Moore–Penrose inverse

$$w^* = (X^T X)^{-1} X^T y$$



Connection to systems of linear equations

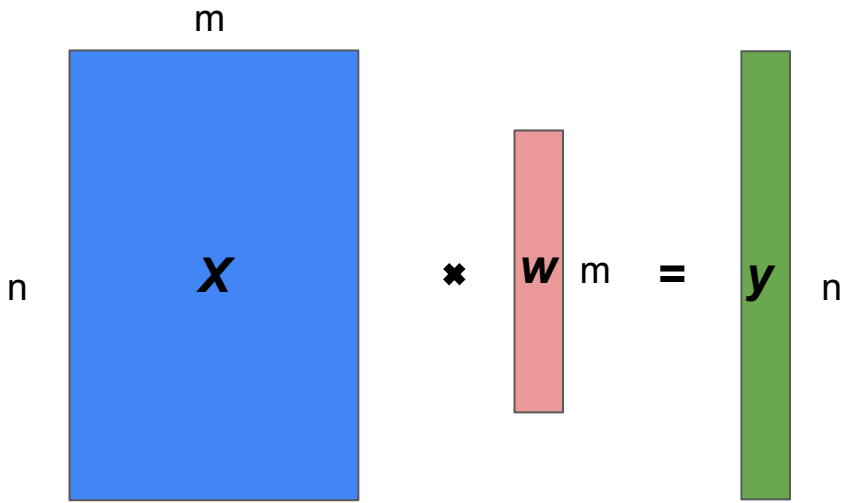
$$Xw = y \quad \Leftrightarrow \quad Ax = b$$

- If n > m and X full-rank then generally no solution
- Instead, we solve a least-squares problem

$$\min_w \|y - Xw\|^2$$

- Solution is given by the Moore–Penrose inverse

$$w^* = (X^T X)^{-1} X^T y$$



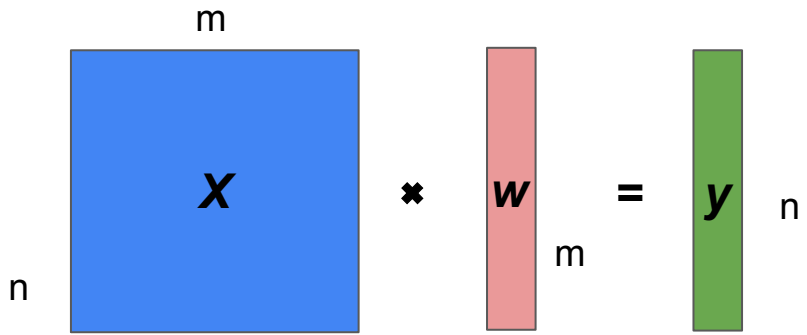
$$[(X^T X)^{-1} X^T] \cdot X = (X^T X)^{-1} (X^T X) = I.$$

Connection to systems of linear equations

$$Xw = y \quad \Leftrightarrow \quad Ax = b$$

- If n = m and X full-rank then unique solution
- We can fit the data perfectly in one unique way
- Pseudo-Inverse simplifies to

$$w^* = X^{-1} y$$



Connection to systems of linear equations

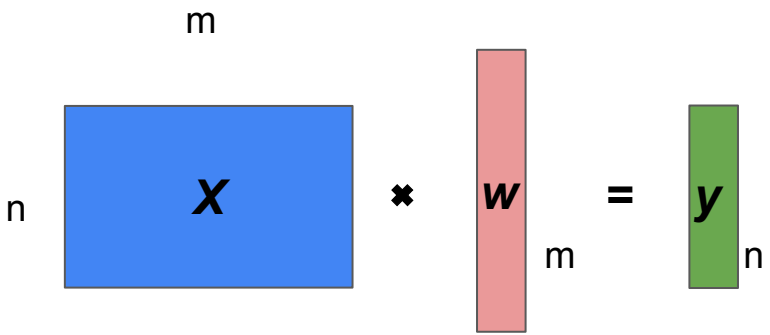
$$Xw = y \quad \Leftrightarrow \quad Ax = b$$

- If n < m and X full-rank then many solutions fit data perfectly
- Our formula no longer works…

$$w^* = (X^T X)^{-1} X^T y$$

- We can solve an optimization problem

$$\min_w \|w\|_2, \text{s.t. } Xw = y$$



Connection to systems of linear equations

$$Xw = y \quad \Leftrightarrow \quad Ax = b$$

- If n < m and X full-rank then many solutions fit data perfectly
- Our formula no longer works…

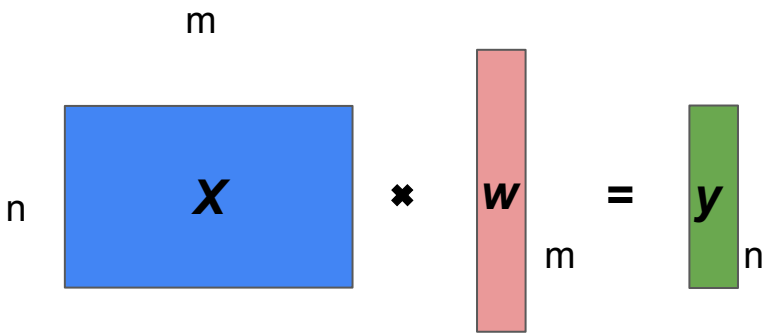
$$w^* = (X^T X)^{-1} X^T y$$

- We can solve an optimization problem

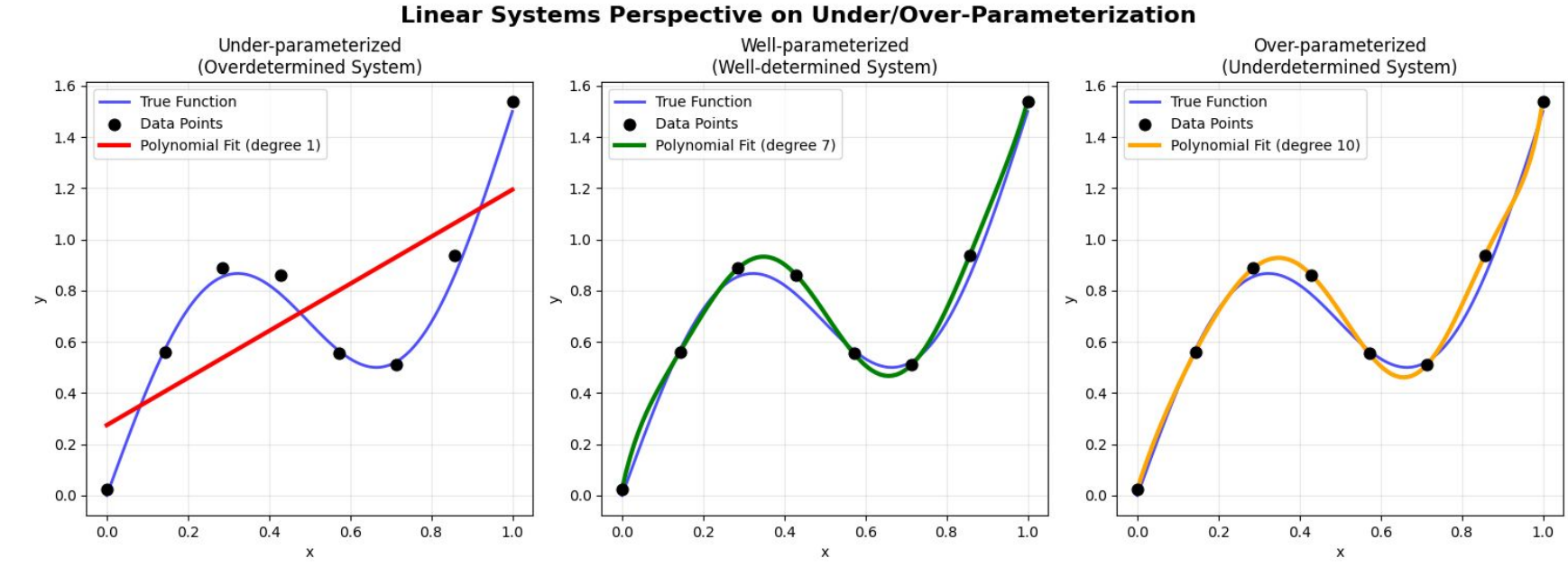
$$\min_w \|w\|_2 , \text{s.t. } Xw = y$$

- Turns out, the solution is still given by the pseudo-inverse (defined via SVD)!

$$w^* = X^+ y$$



Connection to systems of linear equations




```
Linear System: Ax = b

Equations (n): 8
Unknowns (m): 2
Rank of A: 2
Condition #: 3.89e+00

Residual ||Ax-b||: 0.7859

System Type:
• OVERDETERMINED (n > m)
• No exact solution
• Use least squares
• High bias, low variance
```

```
Linear System: Ax = b

Equations (n): 8
Unknowns (m): 8
Rank of A: 8
Condition #: 2.68e+05

Residual ||Ax-b||: 0.0000

System Type:
• WELL-DETERMINED (n ≈ m)
• Unique solution
• Balanced bias-variance
```

```
Linear System: Ax = b

Equations (n): 8
Unknowns (m): 11
Rank of A: 8
Condition #: 3.98e+04

Residual ||Ax-b||: 0.0000

System Type:
• UNDERDETERMINED (n < m)
• Infinite solutions
• Zero residual possible
• Low bias, high variance
```

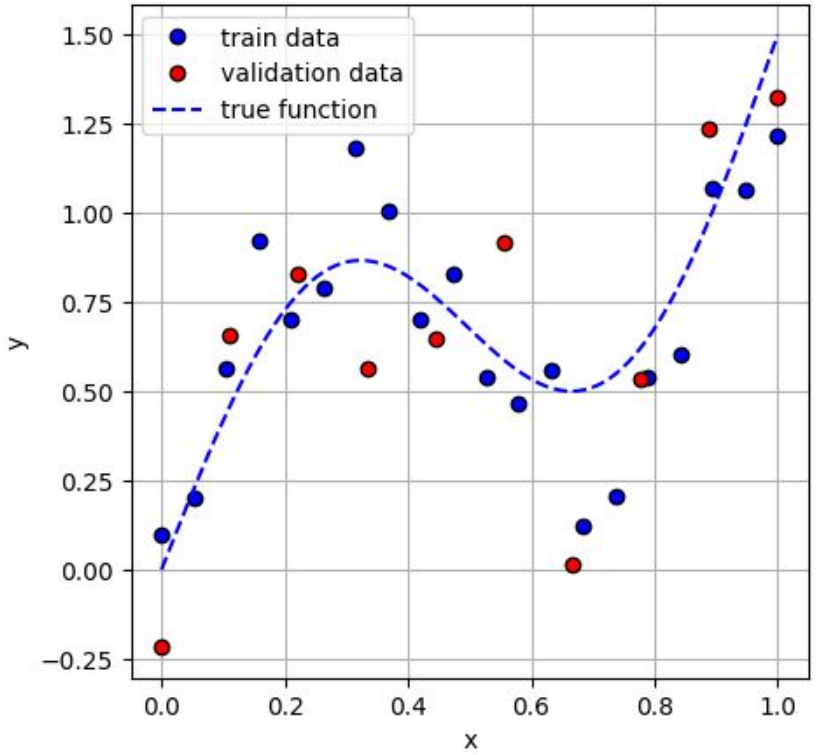
Connection to systems of linear equations

- Equations ↔ datapoints, variables ↔ features
- If we have many equations and few features, we will often underfit
  - When will we not underfit here?
- When we have more (≥) features than data, we will fit the data perfectly
  - In most cases, this would mean overfitting
- Pseudo-inverse gives a general form of optimal solution $w^* = X^+ y$


Regularization
0.34

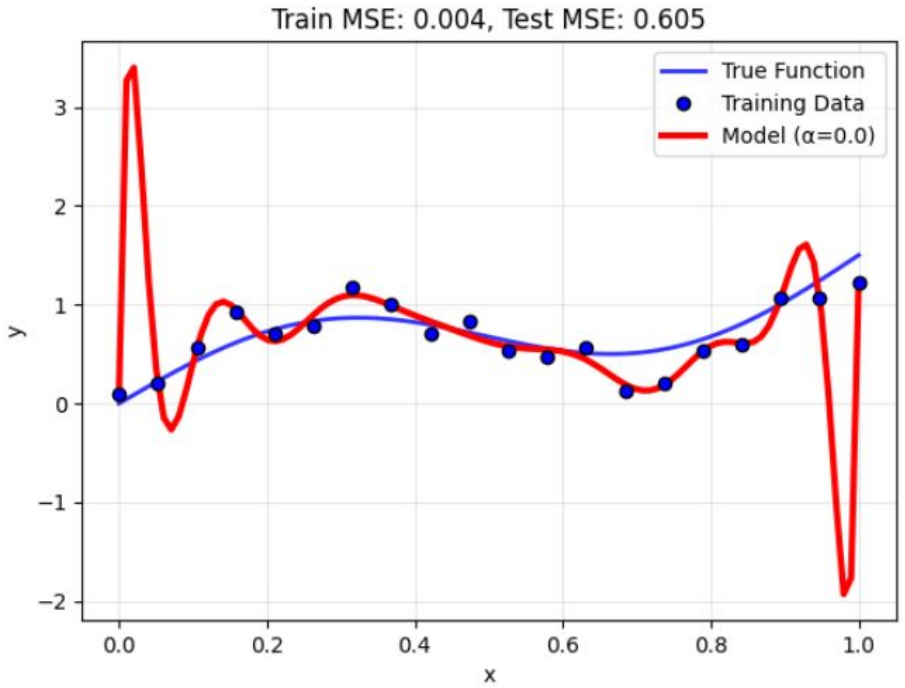
Regularization Motivation

- Go back to our regression problem



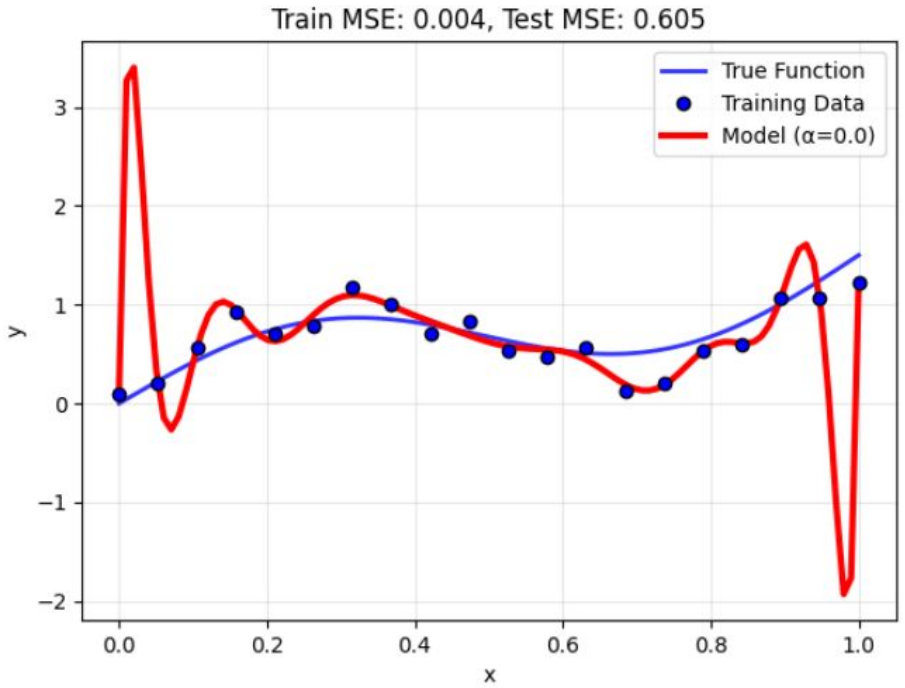
Regularization Motivation

- Go back to our regression problem
- Let’s fit a degree-15 polynomial



Regularization Motivation

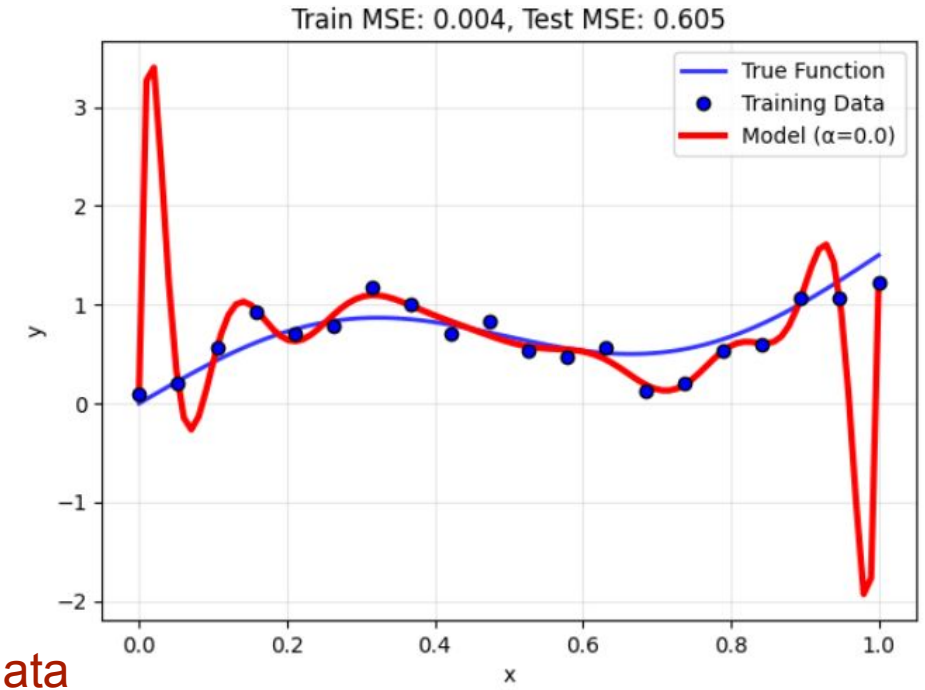
- Go back to our regression problem
- Let's fit a degree-15 polynomial
- We are not in the over-parameterized regime, but close to it
  - We are overfitting badly



9.9E-02 + 0.0E+00 x^1 + 5.6E+02 x^2 - 3.0E+04 x^3 + 6.7E+05 x^4 - 8.2E+06 x^5 + 6.4E+07 x^6 - 3.4E+08 x^7 + 1.3E+09 x^8 - 3.4E+09 x^9 + 6.6E+09 x^10 - 9.6E+09 x^11 + 1.0E+10 x^12 - 7.4E+09 x^13

Regularization Motivation

- Go back to our regression problem
- Let's fit a degree-15 polynomial
- We are not in the over-parameterized regime, but close to it
  - We are overfitting badly

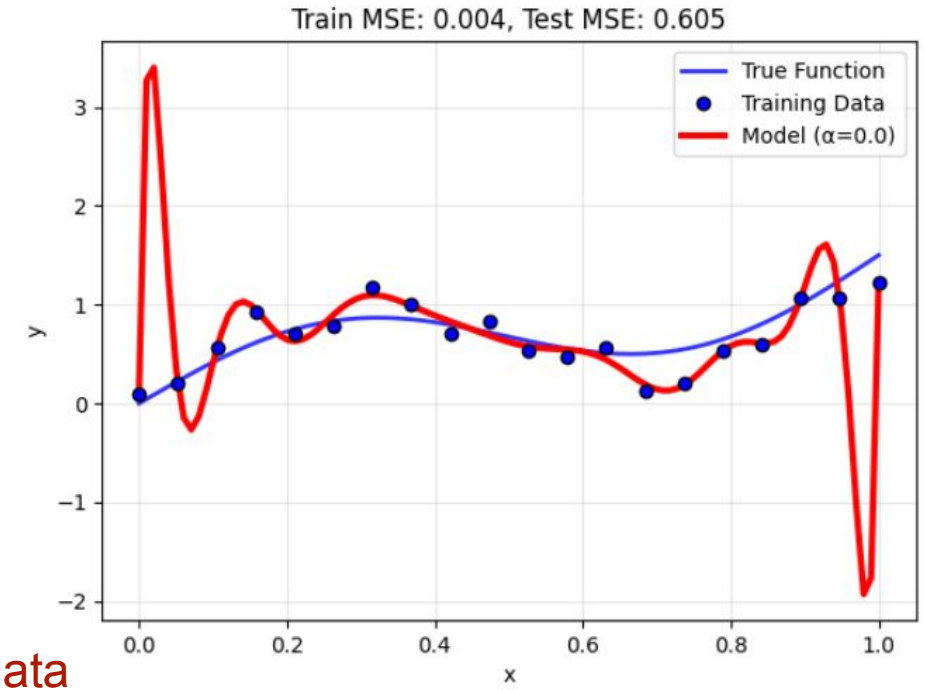


The model has to use big weights to fit the data

9.9E-02 + 0.0E+00 x^1 + 5.6E+02 x^2 - 3.0E+04 x^3 + 6.7E+05 x^4 - 8.2E+06 x^5 + 6.4E+07 x^6 - 3.4E+08 x^7 + 1.3E+09 x^8 - 3.4E+09 x^9 + 6.6E+09 x^10 - 9.6E+09 x^11 + 1.0E+10 x^12 - 7.4E+09 x^13

Regularization Motivation

- We change features a bit and get a lot of change in the predictions
- This happens when the weights are large
- What can we do to prevent large weights?



The model has to use big weights to fit the data

9.9E-02 + 0.0E+00 x^1 + 5.6E+02 x^2 - 3.0E+04 x^3 + 6.7E+05 x^4 - 8.2E+06 x^5 + 6.4E+07 x^6 - 3.4E+08 x^7 + 1.3E+09 x^8 - 3.4E+09 x^9 + 6.6E+09 x^10 - 9.6E+09 x^11 + 1.0E+10 x^12 - 7.4E+09 x^13

Regularization Motivation

What can we do to prevent large weights?

$$\min_{w} \|Xw - y\|^2 + \lambda \|w\|^2$$

We add a *regularization* term that penalizes large weights!

- We trade off fitting the data vs the norm of the weights.
- We can choose different forms of regularization.

Regularization

$$\min_{w} \|Xw - y\|^2 + \lambda \|w\|^2$$

Regularization

$$\min_{w} \|Xw - y\|^2 + \lambda \|w\|^2$$

$$J(w) = (Xw - y)^T (Xw - y) + \lambda w^T w$$

$$\begin{aligned}(Xw - y)^T (Xw - y) &= (w^T X^T - y^T)(Xw - y) \\ &= w^T X^T Xw - w^T X^T y - y^T Xw + y^T y \\ &= w^T X^T Xw - 2w^T X^T y + y^T y\end{aligned}$$

$$J(w) = w^T (X^T X + \lambda I) w - 2w^T X^T y + y^T y$$

- $\frac{\partial}{\partial w}(w^T A w) = 2Aw$ for symmetric matrix A
- $\frac{\partial}{\partial w}(a^T w) = a$ for vector a

$$\frac{\partial J(w)}{\partial w} = 2(X^T X + \lambda I) w - 2X^T y$$

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

Regularization

$$\min_{w} \|Xw - y\|^2 + \lambda \|w\|^2$$

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

- When λ is 0, recover our standard LR solution
- When λ > 0, w* always exists, $(X^T X + \lambda I)$ is always invertible
- What happens when λ → ∞?

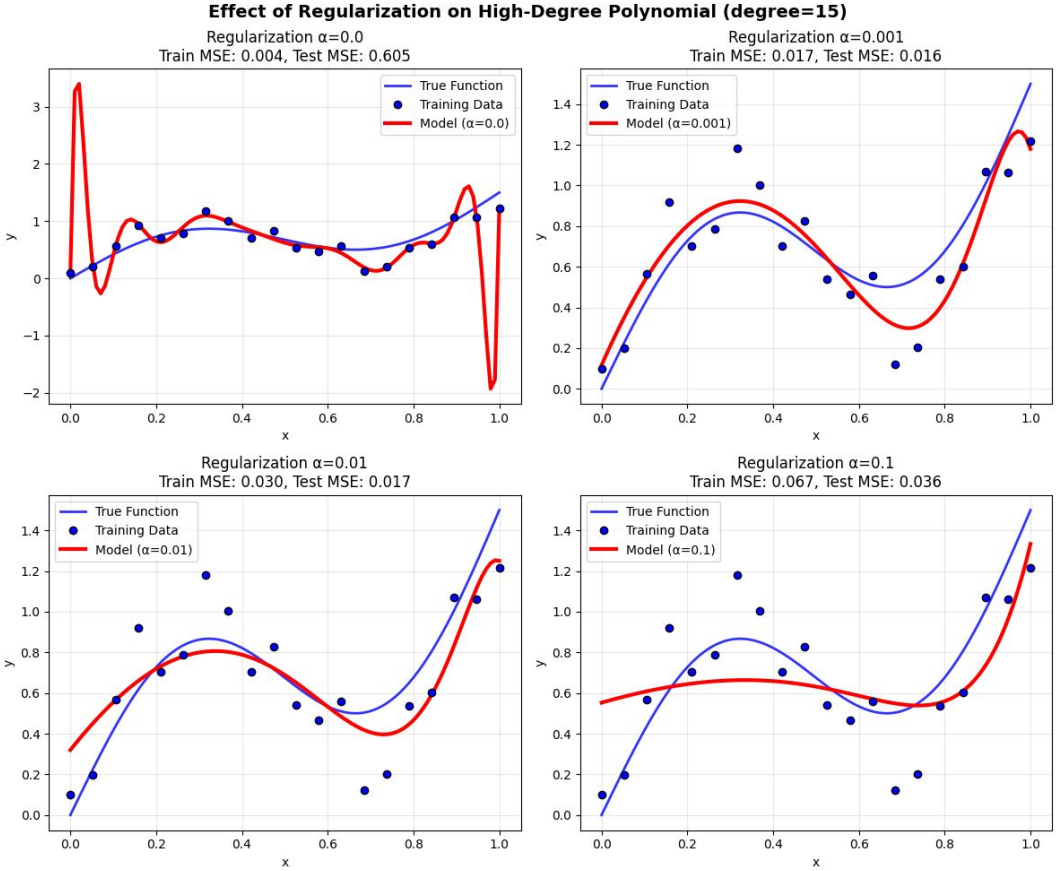
Regularization

$$\min_{w} \|Xw - y\|^2 + \lambda \|w\|^2$$

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

- When λ is 0, recover our standard LR solution
- When λ > 0, w* always exists, $(X^T X + \lambda I)$ is always invertible
- What happens when λ → ∞?
  - w* → 0

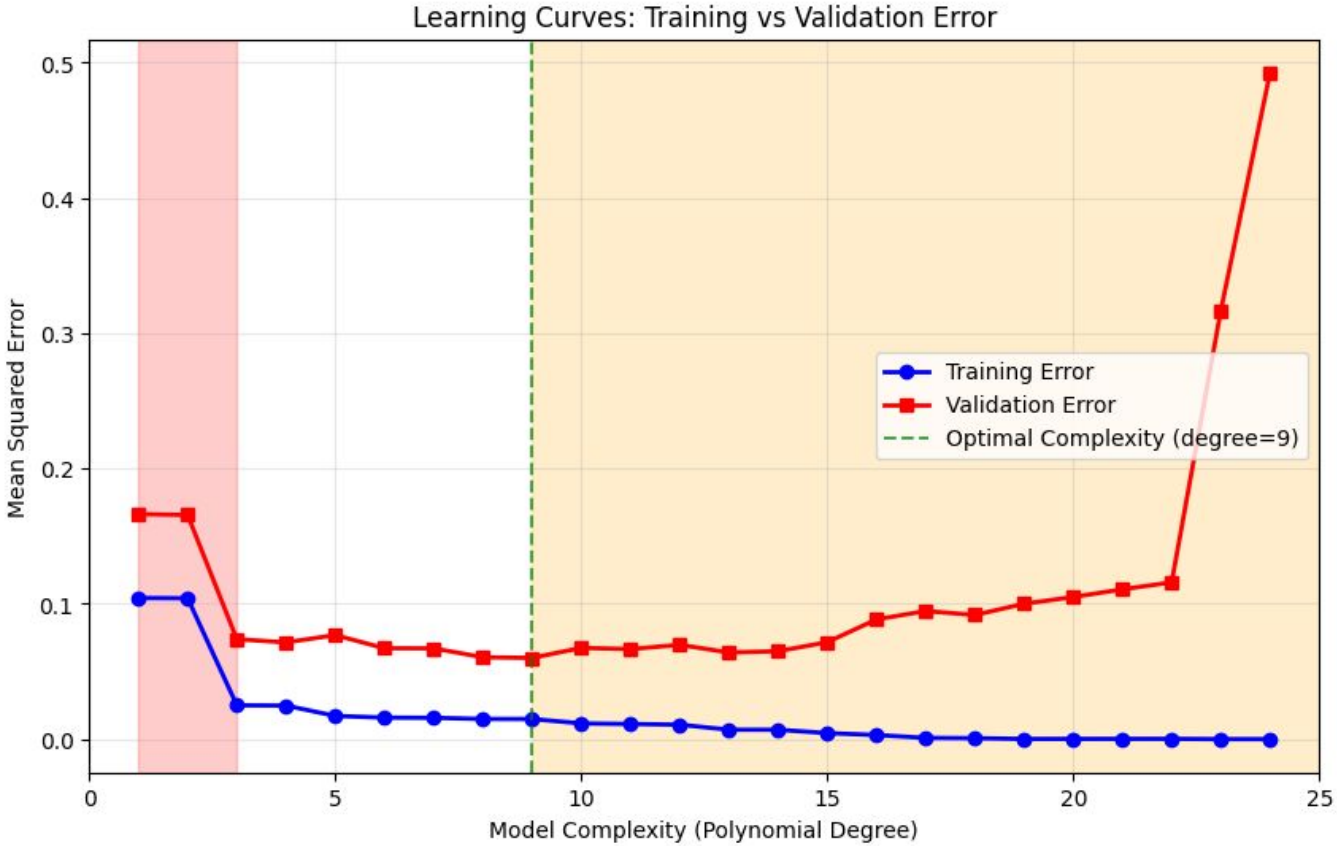
Regularization



- Bigger λ → simpler solution
- When λ is too big, we get *underfitting*
- We now have a way of going from underfitting to overfitting without changing the model family!

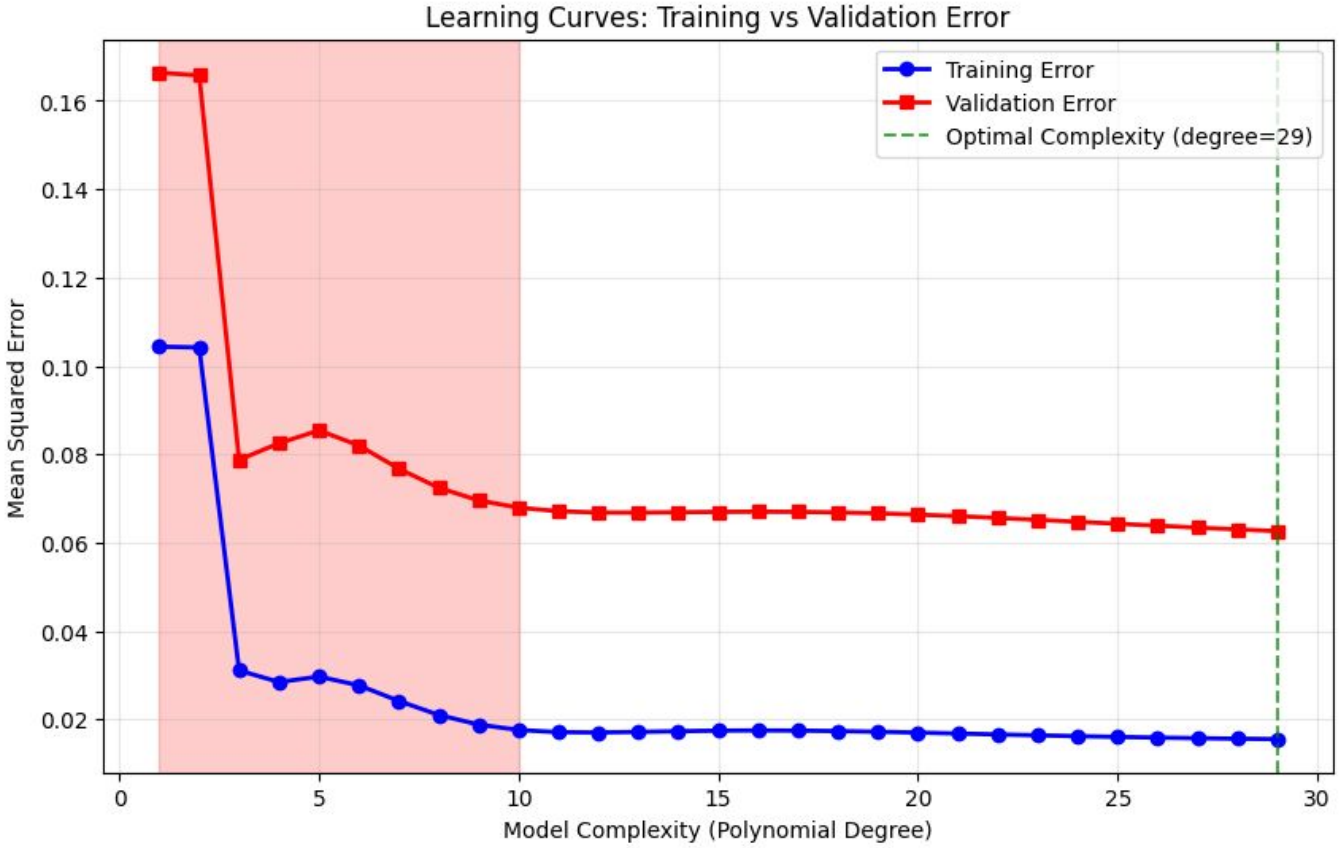
Model selection

- No regularization
- Overfitting



Model selection

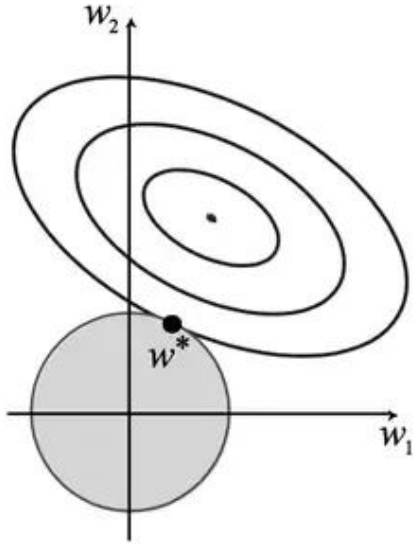
- With regularization
- No overfitting!
- Strength of regularization should be tuned via validation loss



Regularization geometry

$$\min_w \|Xw - y\|^2 + \lambda \|w\|^2$$

- Let's consider $\min \|Xw - y\|^2, \text{ s.t. } \|w\| < \alpha.$
- On the level set of the loss, we will find the point with the lowest norm.
- L2-regularization is also known as weight decay*, it shrinks weights towards the origin
- L2-regularized linear regression is also known as Ridge regression

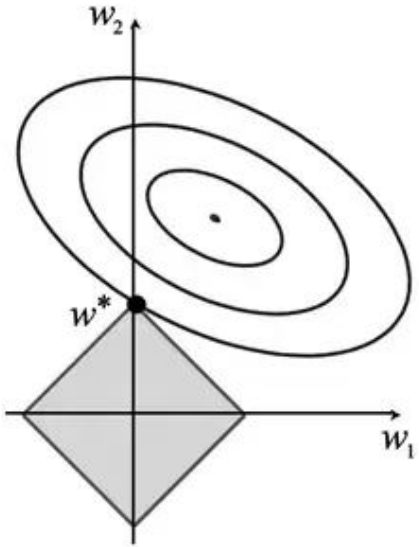


*Strictly speaking, weight decay is not identical to L2 reg, but in practice it often is.

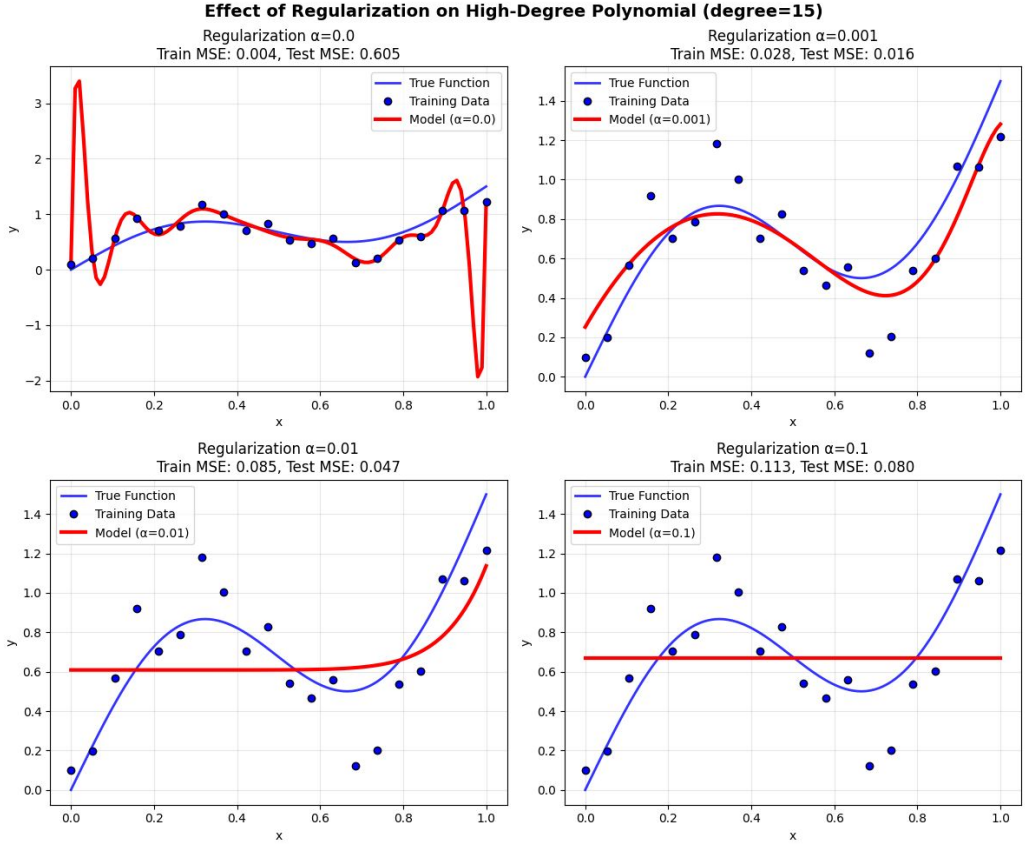
Regularization geometry

$$\min \|Xw - y\|^2 + \lambda \|w\|_1 \qquad \min \|Xw - y\|^2, \text{ s.t. } \|w\|_1 < \alpha.$$

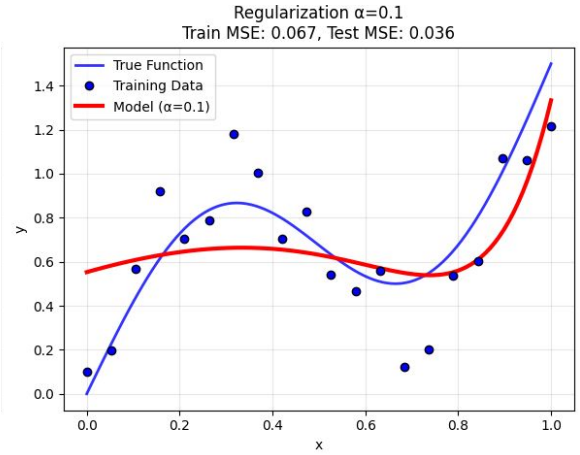
- The level set of L1-norm is a polyhedron.
- The minimizer will end up in a vertex of the polyhedron.
- L1-regularization produces sparse solutions!
- L1 reg is also known as LASSO (least absolute shrinkage and selection operator)
- We don't have a closed-form solution for LASSO!
  - Need to use gradient descent or fancy custom methods



Regularization geometry

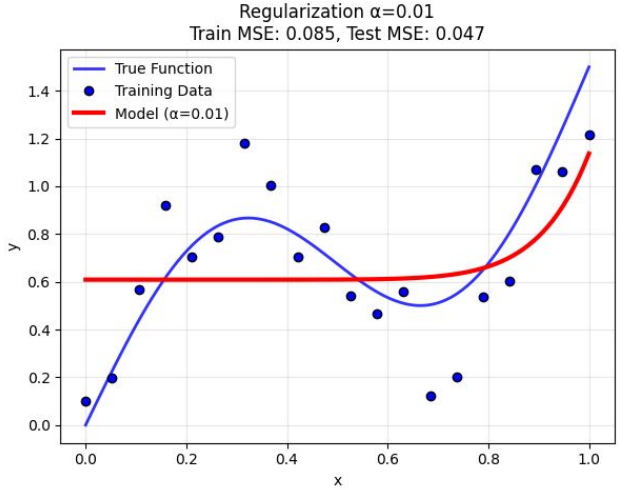


Regularization geometry



Ridge regression chooses low weights

```
5.5E-01 + 0.0E+00 x^1 + 6.0E-01 x^2 - 6.0E-01 x^3
- 5.3E-01 x^4 - 2.2E-01 x^5 + 5.0E-02 x^6 +
2.2E-01 x^7 + 3.0E-01 x^8 + 3.2E-01 x^9 + 3.0E-01
x^10 + 2.4E-01 x^11 + 1.8E-01 x^12 + 1.0E-01 x^13
+ 2.1E-02 x^14 - 5.9E-02 x^15 - 1.4E-01 x^16
```



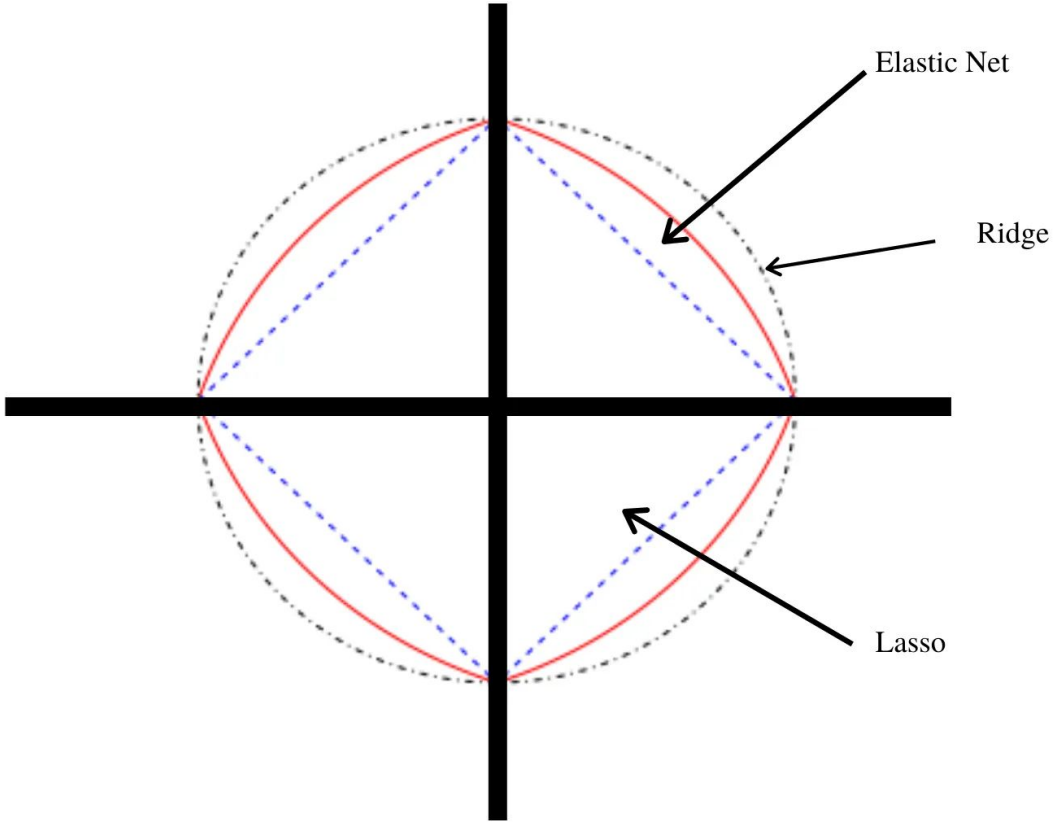
Lasso fit is choosing a sparse solution

```
6.1E-01 + 4.6E-01 x^11 + 7.2E-02 x^12
```

Regularization geometry

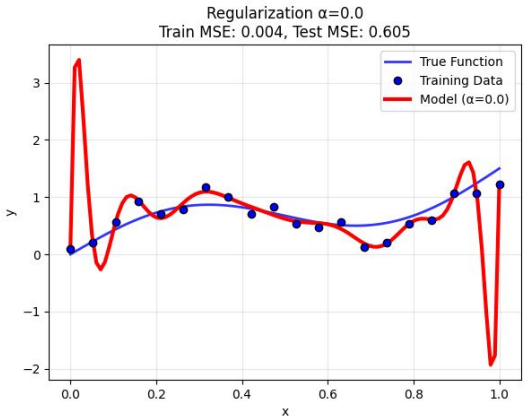
Elastic Net: mix L1 + L2

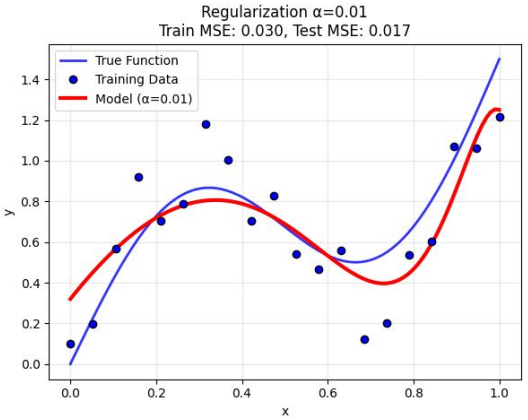
$$\min_{w} \|Xw - y\|^2 + \lambda_1 \|w\|_2^2 + \lambda_2 \|w\|_1$$



Regularization: Summary

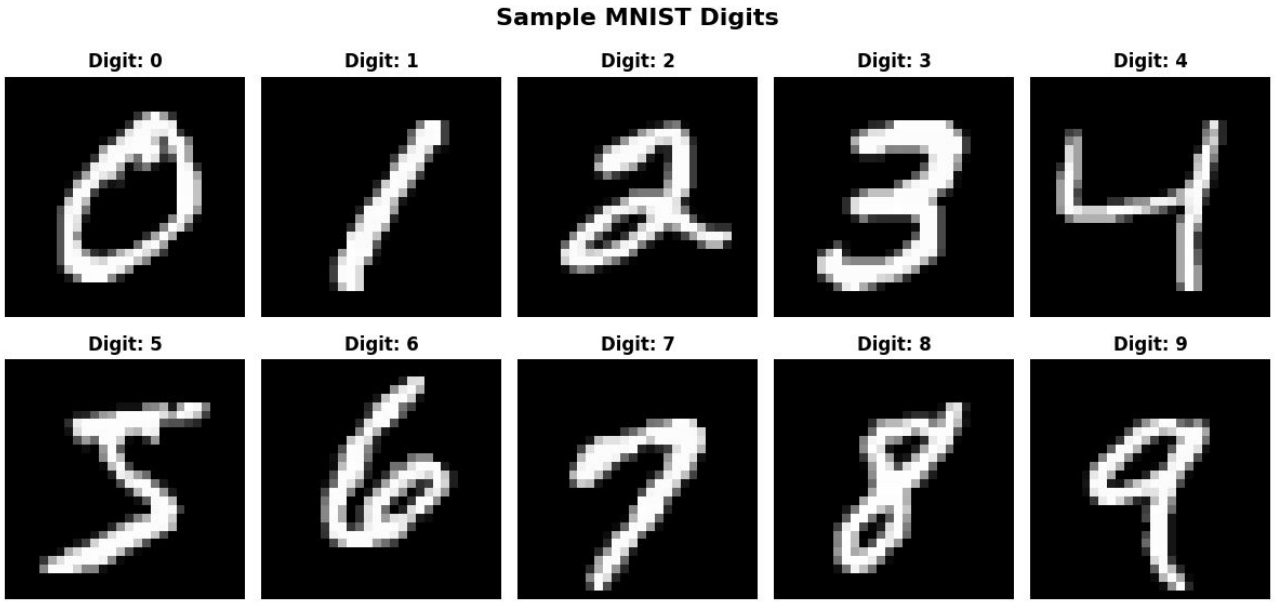
- Regularization is a core idea in ML
  - L2 regularization is used everywhere, from HFT stock price models to LLMs
- Train flexible models without overfitting
- Strength of the regularization is a key parameter that needs to be tuned via (cross-)validation
- L1 regularization can do automatic feature selection






Logistic Regression
0.34

Classification

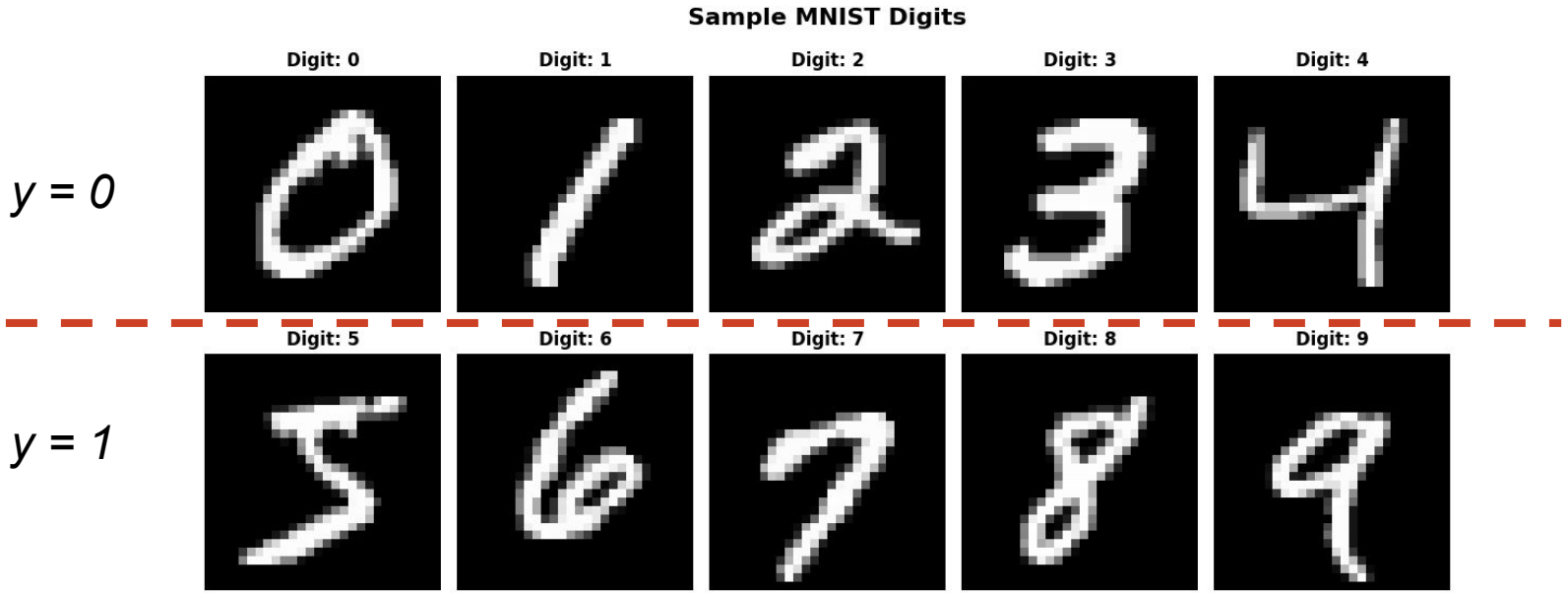


Classification:

$$x \in \mathbb{R}^{28^2}, \quad y \in \{0, 1, \dots, 9\}$$

- Inputs: arbitrary, e.g. images
- Labels: discrete set

Classification



y = 0

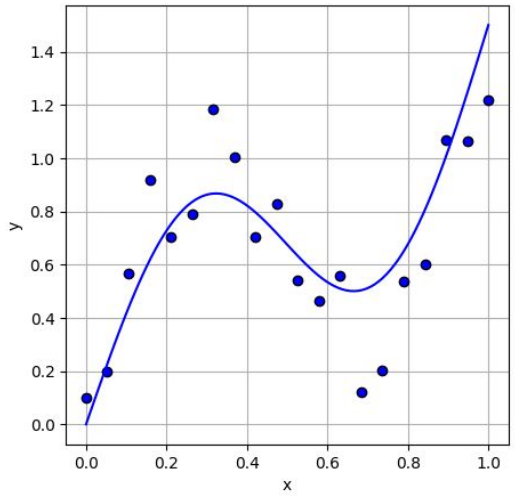
y = 1

Binary Classification:

$$x \in \mathbb{R}^{28^2}, \quad y \in \{0, 1\}$$

- Inputs: arbitrary, e.g. images
- Labels: two classes

Likelihood



Form of z can be arbitrary

Distribution of y | z can be arbitrary

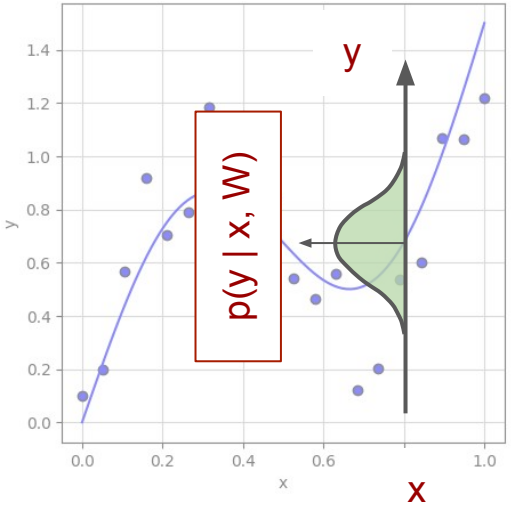
$$z = Xw, \quad p(y|z) = N(z, I)$$

$$\log p(y|X, w) = \log p(y|z) = \log N(z, 1) =$$

$$-\frac{1}{2}\|z - y\|^2 + C = -\frac{1}{2}\|Xw - y\|^2 + C.$$

- We can think of a model as defining the distribution of labels *y* given the parameters *w* of the model and inputs *X*

Likelihood



Form of z can be arbitrary

Distribution of y | z can be arbitrary

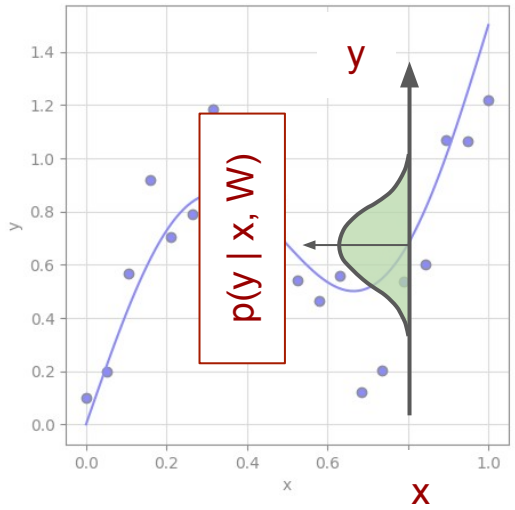
$$z = Xw, \quad p(y|z) = N(z, I)$$

$$\log p(y|X, w) = \log p(y|z) = \log N(z, 1) =$$

$$-\frac{1}{2}\|z - y\|^2 + C = -\frac{1}{2}\|Xw - y\|^2 + C.$$

- We can think of a model as defining the distribution of labels *y* given the parameters *w* of the model and inputs *X*

Likelihood



Form of z can be arbitrary

Distribution of y | z can be arbitrary

$$z = Xw, \quad p(y|z) = N(z, I)$$

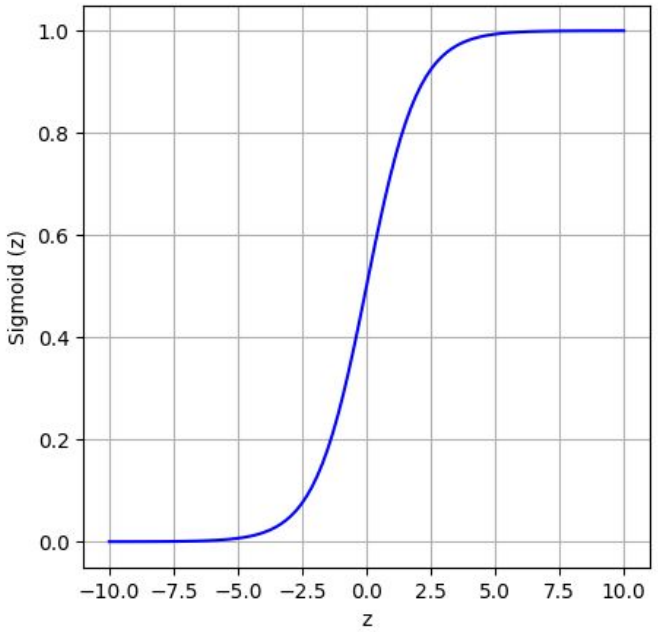
$$\log p(y|X, w) = \log p(y|z) = \log N(z, 1) =$$

$$-\frac{1}{2}\|z - y\|^2 + C = -\frac{1}{2}\|Xw - y\|^2 + C.$$

- We can think of a model as defining the distribution of labels *y* given the parameters *w* of the model and inputs *X*
- This is a powerful way of defining losses
  - Regularization = prior

Classification Loss

$$p(y_i|z_i) = \sigma(z_i) \qquad \sigma(z_i) = \frac{1}{1 + e^{-z_i}}$$



- Likelihood given a score (logit) is the sigmoid (logistic) function

$$z = Xw$$

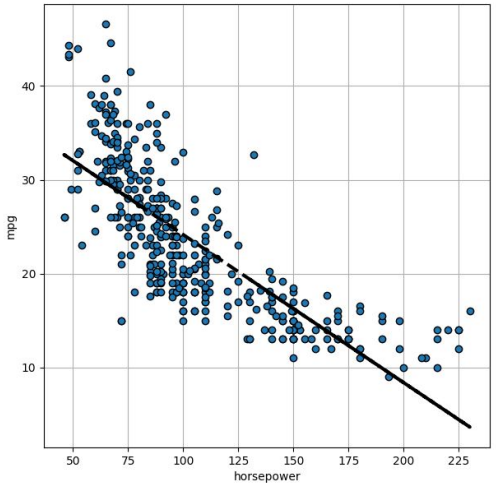
- If the logit *z* is linear, we get logistic regression

$$p(y_i|X_i, w) = \sigma(X_i w),$$

$$L(w) = \log(p(y|X, w)) = \sum_{i=1}^{N} \log\left(\sigma(X_i w)\right).$$

Classification Loss

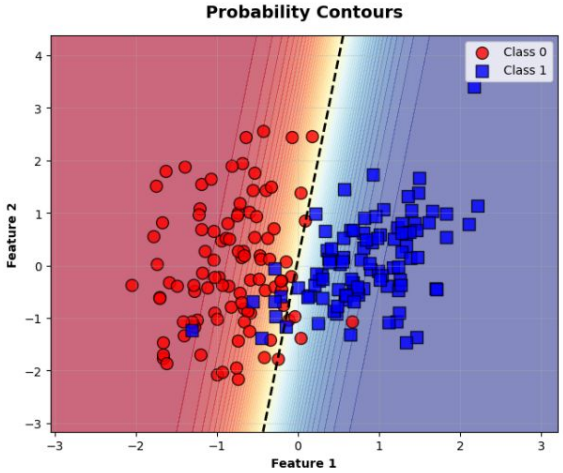
Linear Regression



$$\log p(y|X,w) = \log p(y|z) = \log N(z,1) =$$

$$-\frac{1}{2}\|z-y\|^2 + C = -\frac{1}{2}\|Xw-y\|^2 + C.$$

Logistic Regression

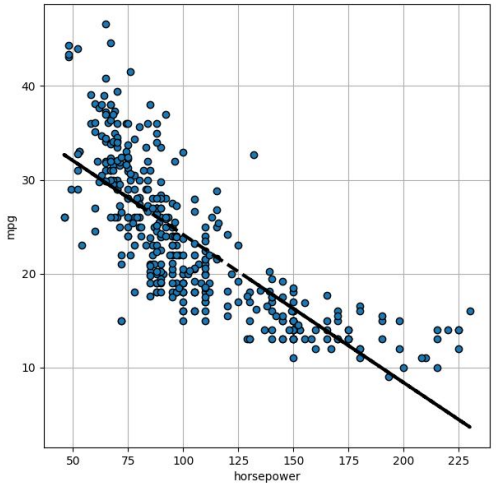


$$p(y_i|X_i,w) = \sigma(X_i w),$$

$$L(w) = \log(p(y|X,w)) = \sum_{i=1}^{N} \log\left(\sigma(X_i w)\right).$$

Classification Loss

Linear Regression

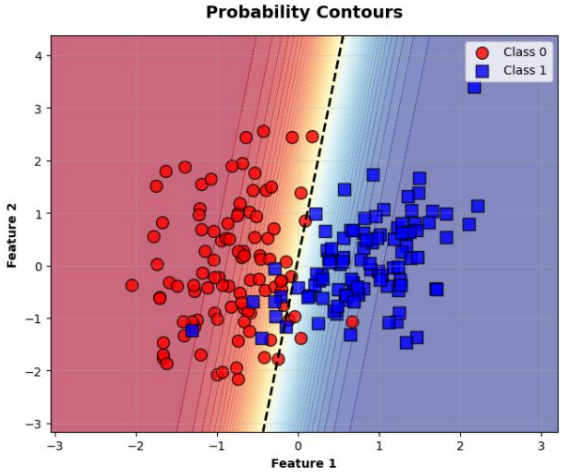


$$\log p(y|X,w) = \log p(y|z) = \log N(z,1) =$$

$$-\frac{1}{2}\|z-y\|^2 + C = -\frac{1}{2}\|Xw-v\|^2 + C.$$

$$+\lambda\|w\|^2$$

Logistic Regression



$$p(y_i|X_i,w) = \sigma(X_i w),$$

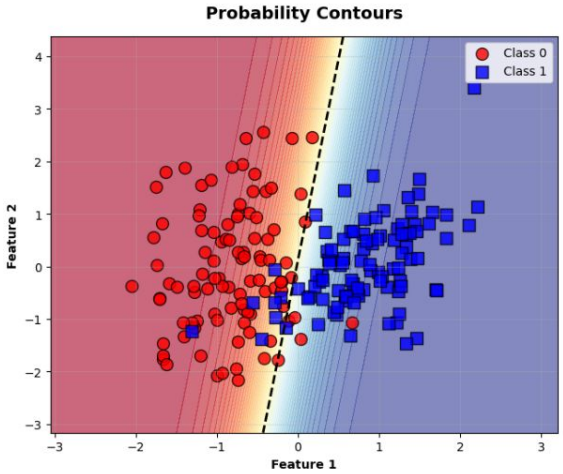
$$L(w) = \log(p(y|X,w)) = \sum_{i=1}^{N} \log\left(\sigma(X_i w)\right).$$

$$+\lambda\|w\|^2$$

Classification Loss

- Unlike linear regression, we cannot solve logistic regression in a closed form
- We will use gradient descent to minimize the loss

Logistic Regression

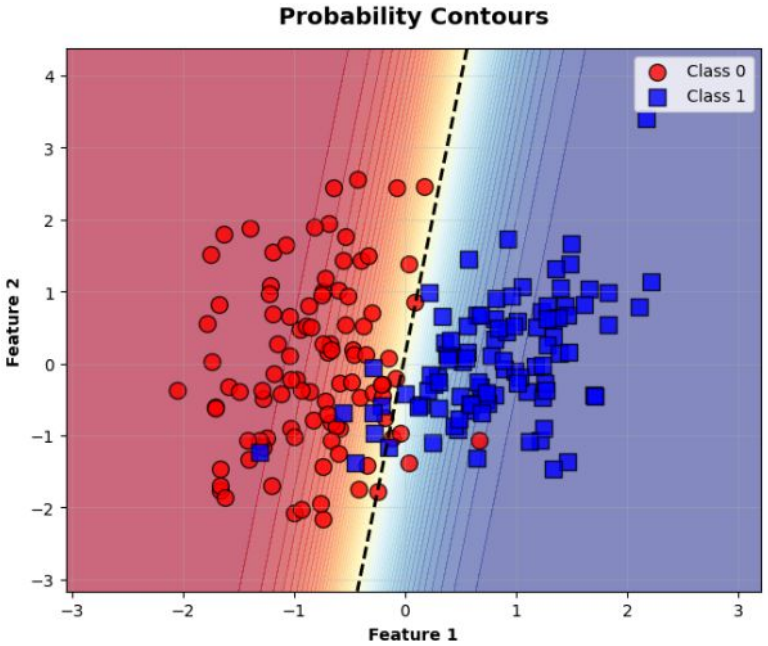


$$p(y_i|X_i, w) = \sigma(X_i w),$$

$$L(w) = \log(p(y|X, w)) = \sum_{i=1}^{N} \log\left(\sigma(X_i w)\right).$$

$$+ \lambda \|w\|^2$$

Logistic regression geometry



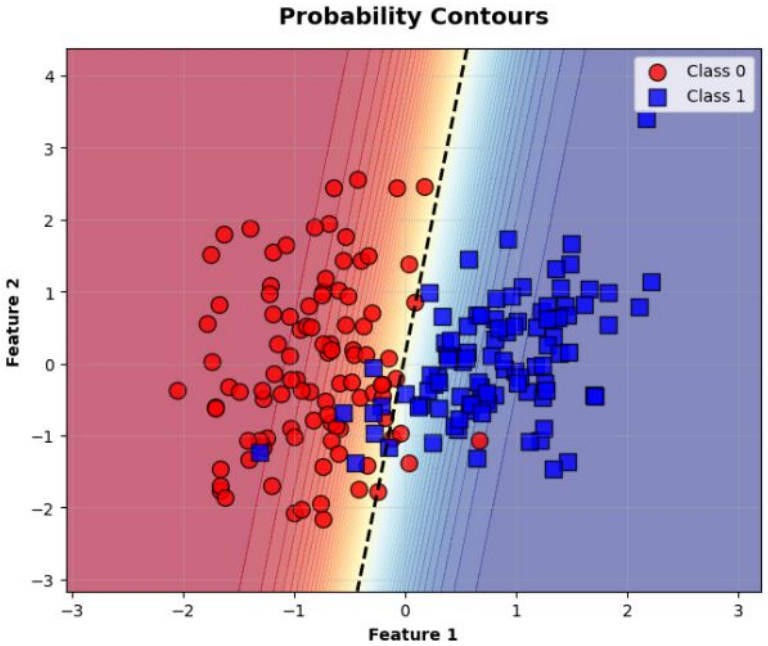
$$p(y_i|X_i, w) = \sigma(X_i w),$$

- The decision boundary ($p = 0.5$) is a linear subspace

$$xw = 0$$

- The decision boundary goes through $x=0$
- How can we separate the decision boundary from 0?
- How can we make the decision boundary non-linear?

Logistic regression geometry



$$p(y_i|X_i, w) = \sigma(X_i w),$$

- The decision boundary ($p = 0.5$) is a linear subspace

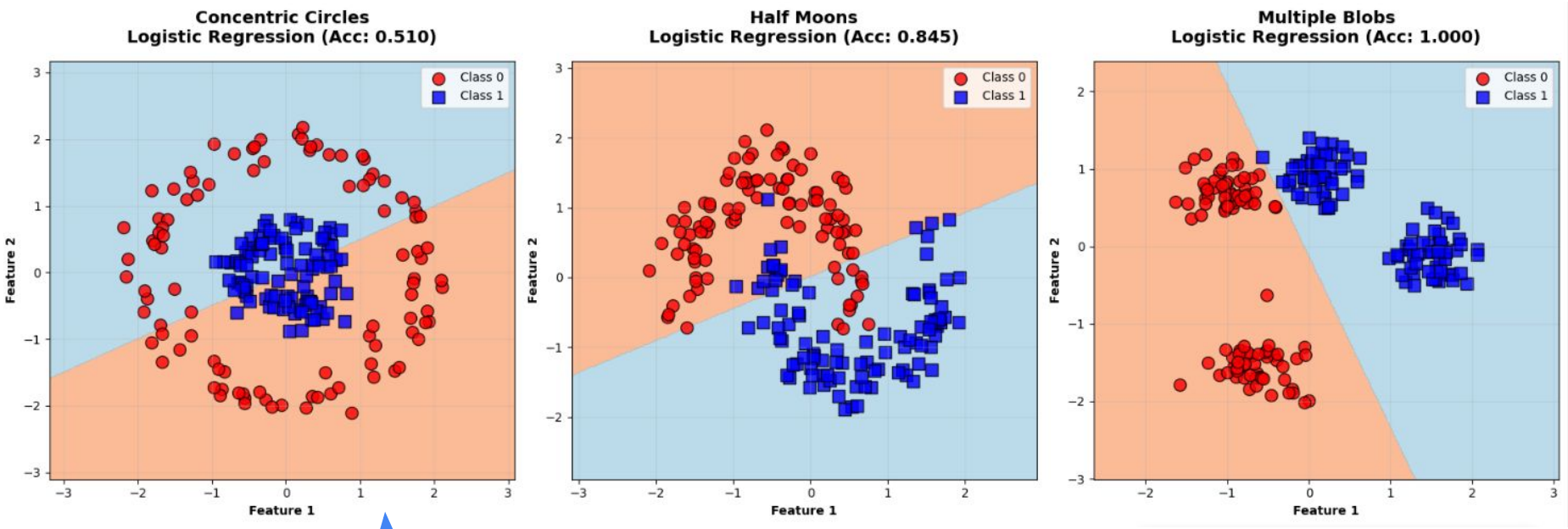
$$xw = 0$$

- The decision boundary goes through $x=0$
- How can we separate the decision boundary from 0?

  Add a bias term
- How can we make the decision boundary non-linear?

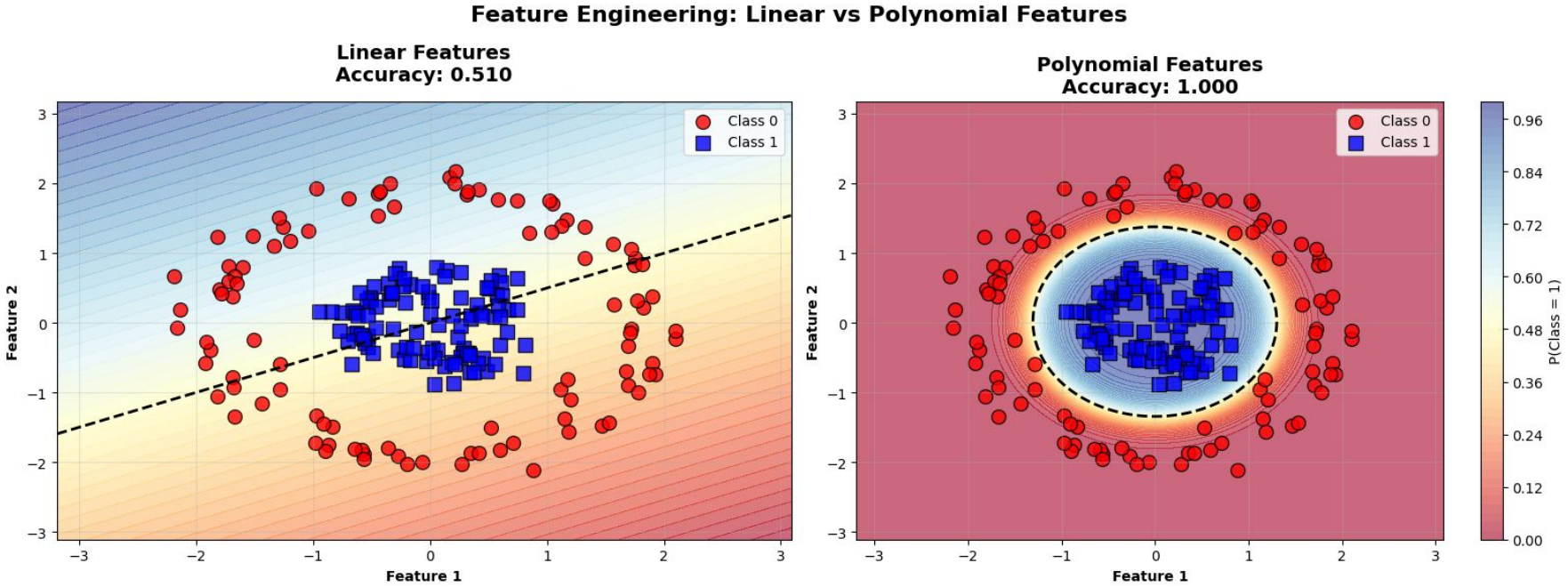
  Use non-linear features

Logistic regression geometry



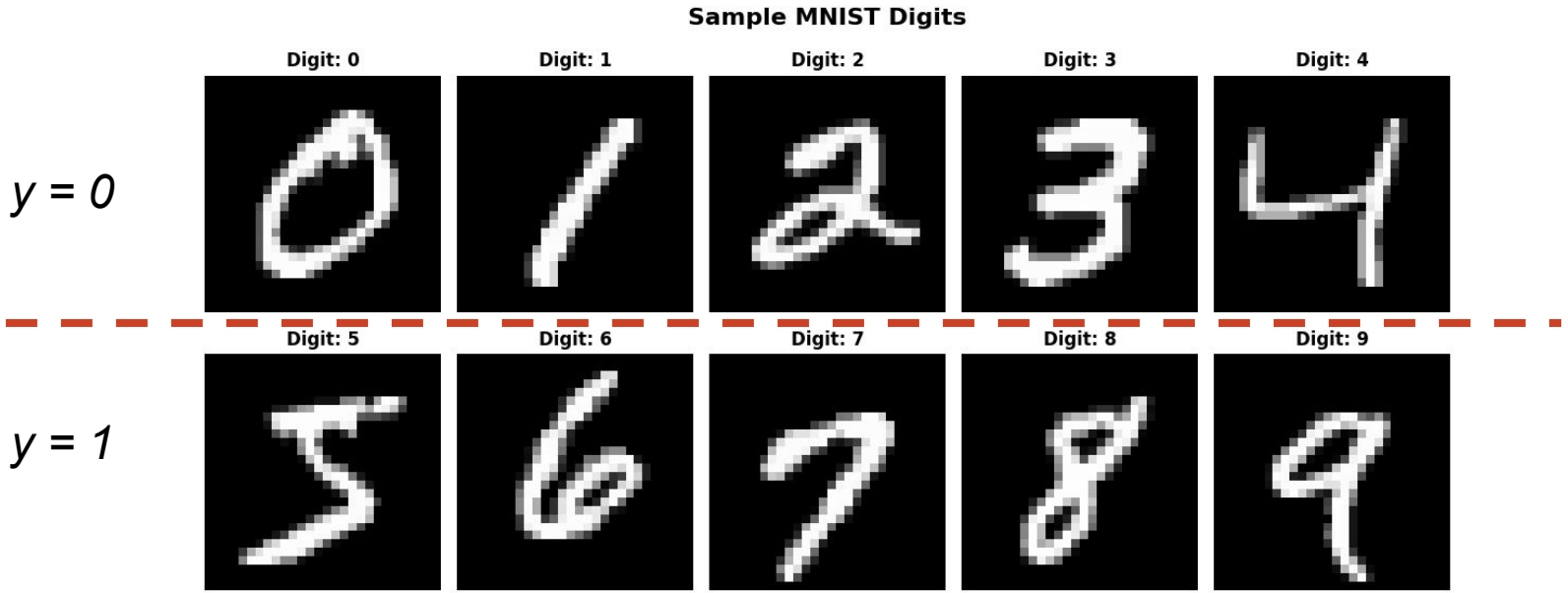
What features would work well here?

Logistic regression geometry



With degree-2 polynomial features, we can achieve perfect accuracy.

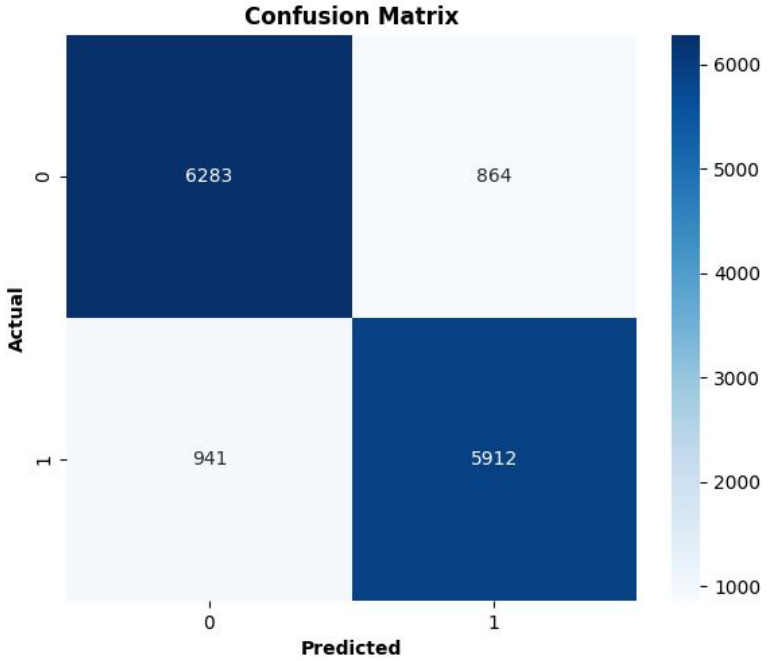
Logistic regression: MNIST



$$x \in \mathbb{R}^{28^2}, \quad y \in \{0, 1\}$$

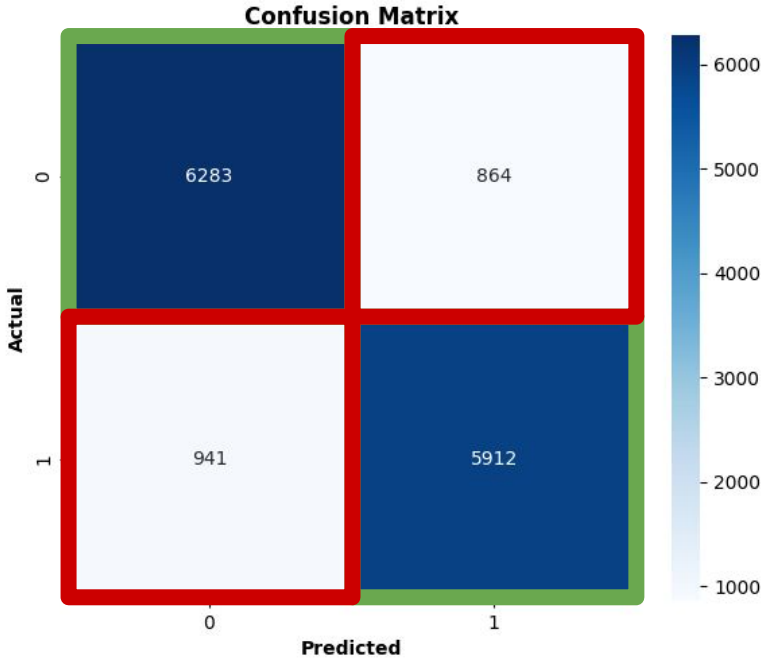
Binary classification accuracy with logistic regression: 87.1%

Logistic regression: MNIST

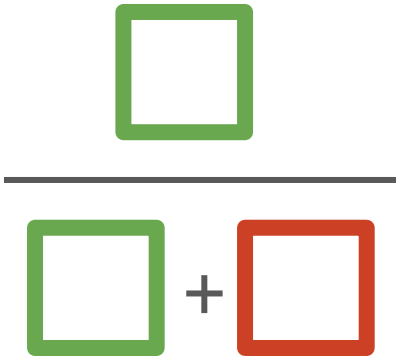


- Confusion matrix: predicted vs actual label distribution
- Accuracy: fraction of correct answers

Logistic regression: MNIST



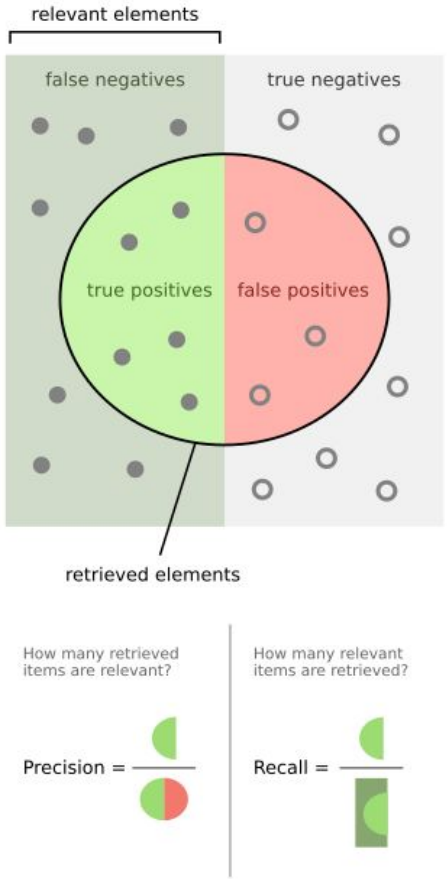
- Confusion matrix: predicted vs actual label distribution
- Accuracy: fraction of correct answers



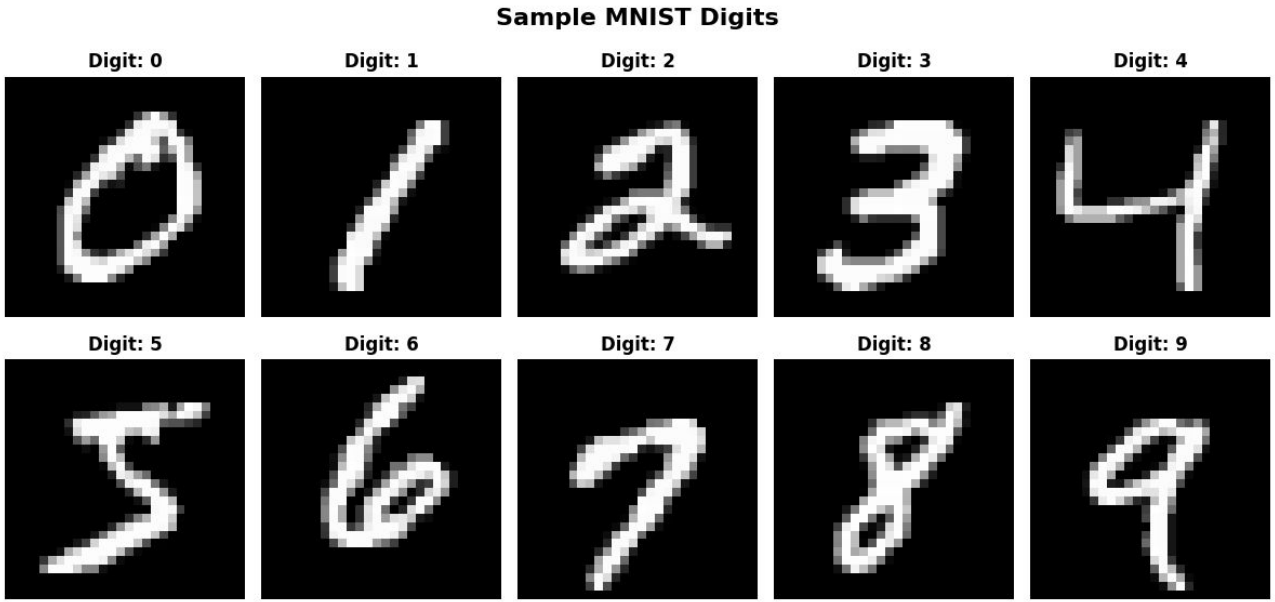
Classification metrics

Other metrics for classification:

- Precision
- Recall
- $F_1 = \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$



Multiclass classification



Classification:

$$x \in \mathbb{R}^{28^2}, \quad y \in \{0, 1, \dots, 9\}$$

- Inputs: arbitrary, e.g. images
- Labels: discrete set

How to define a likelihood?

Multiclass classification

Binary Classification

$$y \in \{0, 1\}$$

$$z \in \mathbb{R}$$

$$p(y \mid z) = \sigma(z)$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Multiclass Classification

$$y \in \{0, 1, \ldots, C - 1\}$$

$$z \in \mathbb{R}^C$$

$$p(y \mid z) = \text{softmax}(z)$$

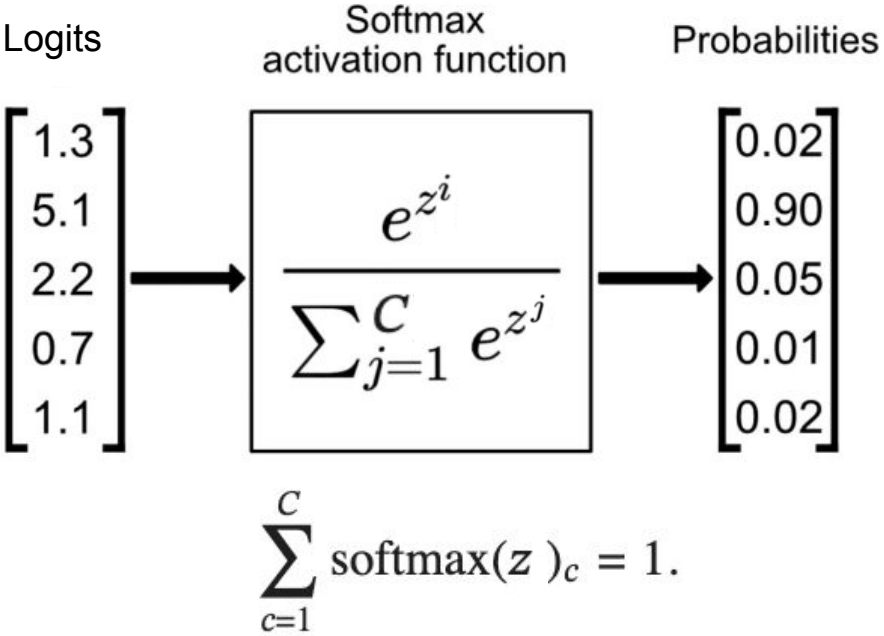
Multiclass classification

Multiclass Classification

$$y \in \{0, 1, \ldots, C-1\}$$

$$z \in \mathbb{R}^C$$

$$p(y \mid z) = \text{softmax}(z)$$



Multiclass classification

Multiclass Classification

$$y \in \{0, 1, \ldots, C-1\}$$

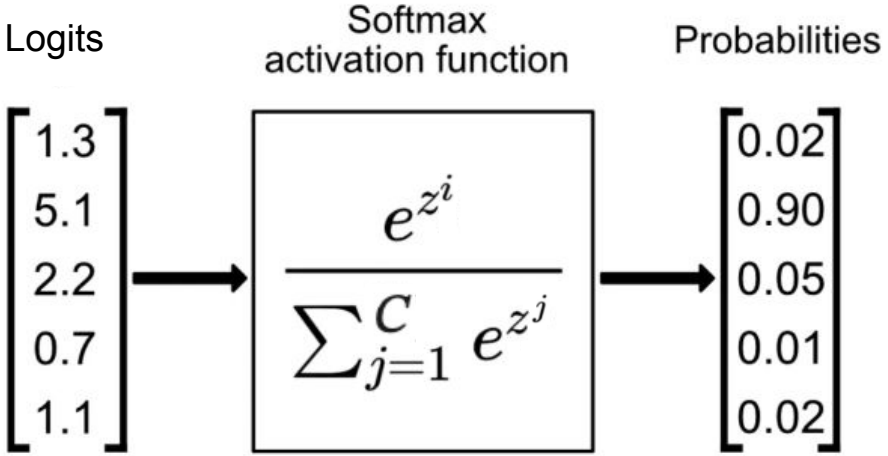
$$z \in \mathbb{R}^C$$

$$p(y \mid z) = \text{softmax}(z)$$

$$z_i = X_i w, \quad p(y_i \mid z_i) = \text{softmax}(z_i)_{y_i}$$

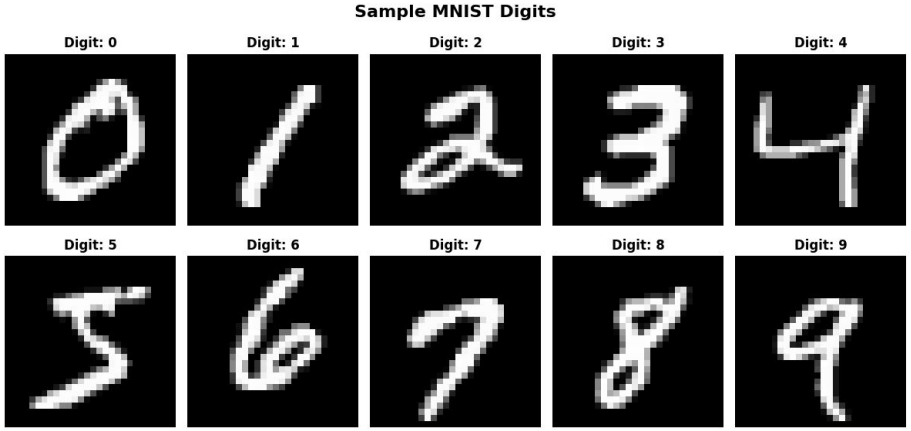
$$X_i \in \mathbb{R}^d, w \in \mathbb{R}^{d \times C}, z_i \in \mathbb{R}^C,$$

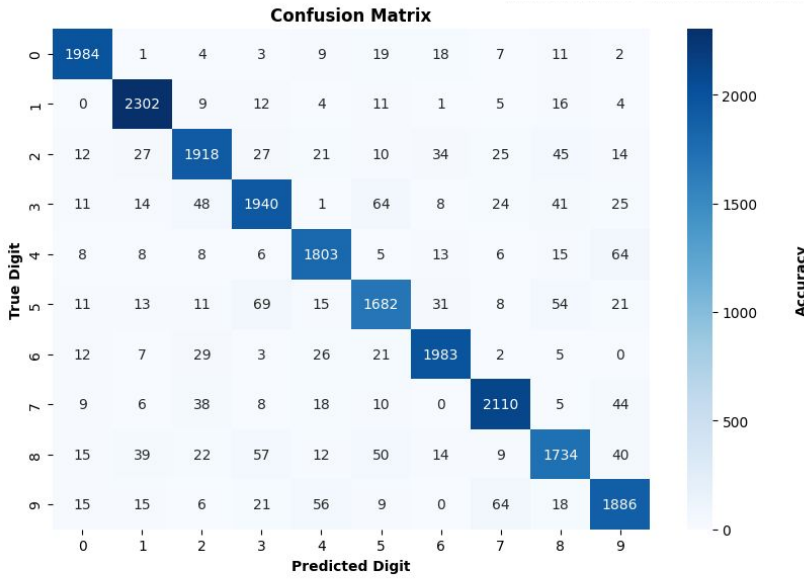
$$y_i \in \{0, 1, \ldots, C\}.$$



$$\sum_{c=1}^{C} \text{softmax}(z)_c = 1.$$

Multiclass classification





Multiclass Classification Accuracy: 92%

Multiclass classification