

# Classification and Logistic Regression

---

Natasha Jaques



# Regression vs classification

- Regression: Given input  $x$ , predict continuous value  $y \in \mathbb{R}$
- Classification: Given input  $x$ , predict discrete value  $y \in \mathcal{Y}$

$x$

$y$



Temperature: 62F

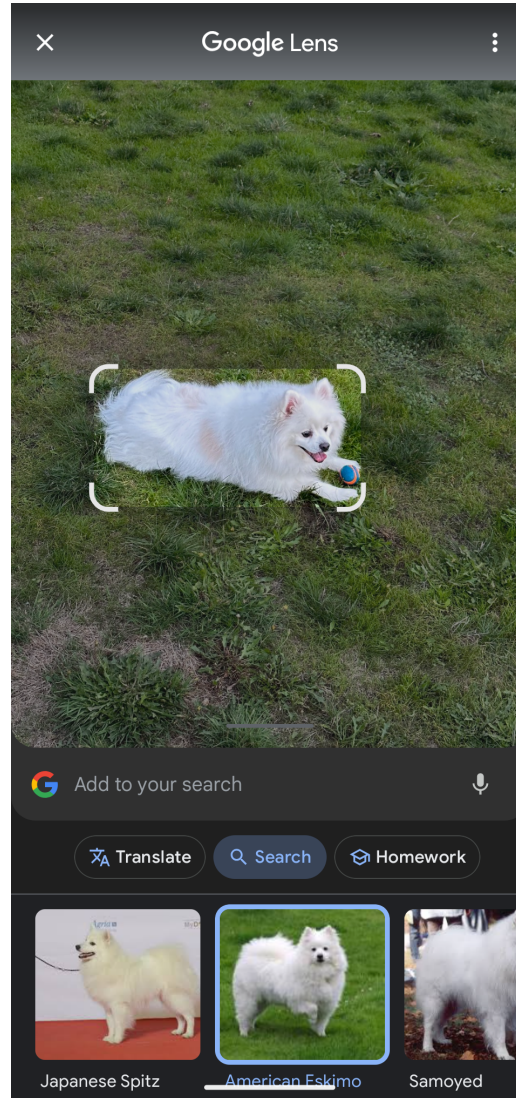
# Regression



# Classification

# Classification problems are ubiquitous

$x$



$y$

# Classification problems are ubiquitous



Healthy  
Pneumonia  
Pneumothorax  
Pleural Effusion  
...

$y$

# Text generation as classification

Enter text:

One, two,



3198 11 734 11

## Prediction

| # | probs  | next token ID | predicted next token |
|---|--------|---------------|----------------------|
| 0 | 39.71% | 1115          | three                |
| 1 | 16.97% | 290           | and                  |
| 2 | 7.55%  | 734           | two                  |
| 3 | 3.76%  | 1440          | four                 |
| 4 | 2.76%  | 393           | or                   |
| 5 | 2.18%  | 1936          | five                 |
| 6 | 1.57%  | 530           | one                  |
| 7 | 1.43%  | 345           | you                  |
| 8 | 1.15%  | 257           | a                    |
| 9 | 0.84%  | 3598          | seven                |

# Text generation as classification

Enter text:

One, two, three

---



3198 11 734 11 1115

## Prediction

| # | probs  | next token ID | predicted next token |
|---|--------|---------------|----------------------|
| 0 | 54.42% | 11            | ,                    |
| 1 | 5.45%  | 1399          | ...                  |
| 2 | 4.82%  | 13            | .                    |
| 3 | 4.51%  | 290           | and                  |
| 4 | 2.72%  | 986           | ...                  |
| 5 | 2.51%  | 25            | :                    |
| 6 | 1.50%  | 393           | or                   |
| 7 | 1.23%  | 3926          | ...                  |
| 8 | 0.85%  | 553           | ,                    |
| 9 | 0.84%  | 960           | —                    |

# Text generation as classification

Enter text:

One, two, three,

---



3198 11 734 11 1115 11

## Prediction

| # | probs  | next token ID | predicted next token |
|---|--------|---------------|----------------------|
| 0 | 46.44% | 1440          | four                 |
| 1 | 7.48%  | 290           | and                  |
| 2 | 7.31%  | 1936          | five                 |
| 3 | 2.66%  | 393           | or                   |
| 4 | 2.54%  | 2237          | six                  |
| 5 | 2.09%  | 1115          | three                |
| 6 | 1.86%  | 3863          | maybe                |
| 7 | 1.62%  | 345           | you                  |
| 8 | 1.23%  | 257           | a                    |
| 9 | 0.92%  | 530           | one                  |

# Text generation as classification

Enter text:

One, two, three, four



3198 11 734 11 1115 11 1440

## Prediction

| # | probs  | next token ID | predicted next token |
|---|--------|---------------|----------------------|
| 0 | 50.14% | 11            | ,                    |
| 1 | 6.66%  | 13            | .                    |
| 2 | 5.91%  | 1399          | ...                  |
| 3 | 3.15%  | 25            | :                    |
| 4 | 2.63%  | 290           | and                  |
| 5 | 2.58%  | 986           | ...                  |
| 6 | 1.42%  | 3926          | ...                  |
| 7 | 1.17%  | 553           | ,                    |
| 8 | 1.09%  | 960           | —                    |
| 9 | 1.08%  | 526           | ."                   |

# Classification

- Learn  $f: \mathcal{X} \rightarrow \mathcal{Y}$ 
  - $\mathcal{X} \subset \mathbb{R}^d$ : features
  - $\mathcal{Y} = \{1, \dots, k\}$ : target classes
- Expected loss:

- 0-1 loss function:

$$\ell(f(x), y) = \mathbf{1}\{f(x) \neq y\}$$



$$\mathbb{E}_{XY}[\mathbf{1}\{f(x) \neq y\}] = \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq y | X = x\}]]$$

# By tower rule

$$\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\} | X = x] = \sum_{i=1}^k P(Y = i | X = x) \mathbf{1}\{f(x) \neq i\}$$

# Expectation of indicator is just probability of that event

$$= \sum_{i \neq f(x)} P(Y = i | X = x)$$

$$= 1 - P(Y = f(x) | X = x)$$

# Classification

- Learn  $f: \mathcal{X} \rightarrow \mathcal{Y}$ 
  - $\mathcal{X} \subset \mathbb{R}^d$ : features
  - $\mathcal{Y} = \{1, \dots, k\}$ : target classes

- Expected loss:

$$\mathbb{E}_{\mathbf{X}\mathbf{Y}}[\mathbf{1}\{f(x) \neq y\}] = 1 - P(Y = f(x) | X = x)$$

- What is optimal predictor for minimizing this loss?
- **Bayes-Optimal Classifier:**

$$f^*(x) = \arg \max_{y \in \mathcal{Y}} P(Y = y | X = x)$$

# For any  $x$ , pick the  $y$  that maximizes true underlying probability... which you don't have

- 0-1 loss function:

$$\ell(f(x), y) = \mathbf{1}\{f(x) \neq y\}$$

Prediction

Label

# Goal is to minimize errors

# Bayes-optimal classifier

$$f^*(x) = \arg \max_{y \in \mathcal{Y}} P(Y = y | X = x) \quad \# \text{ Unknown probability, property of the true distribution}$$

In practice, we don't know  $P(Y = y | X = x)$ , but have  $n$  i.i.d. examples:

$$\left\{ (x_i, y_i) \right\}_{i=1}^n \quad \# \text{ Training data}$$

Suppose  $\mathcal{X}$  is discrete so that  $x \in \{1, 2, \dots, m\}$ . What is a natural estimator for  $P(Y = y | X = x)$ ?

$$\hat{f}(x) = \operatorname{argmax}_y \frac{\sum_{i=1}^n 1\{x_i = x, y_i = y\}}{\sum_{i=1}^n 1\{x_i = x\}}$$

What if  $\mathcal{X}$  is continuous? We need a model to explain observations!

# Maximum likelihood estimation for classification

$$f^*(x) = \operatorname{argmax}_y P(Y = y | X = x)$$

$$\hat{f}_w(x) = \operatorname{argmax}_y P_w(Y = y | X = x)$$

General MLE procedure:

1. Parameterize  $P_w(Y = y | X = x)$  as a function of  $w$

2. Learn  $w$  on i.i.d. training data  $\{(x_i, y_i)\}_{i=1}^n$  # Maximize likelihood of training data

$$\hat{w}_{\text{MLE}} = \operatorname{argmax}_w \prod_{i=1}^n P_w(y_i | x_i)$$

$$= \operatorname{argmax}_w \sum_{i=1}^n \log P_w(y_i | x_i)$$

# Modeling conditional probabilities

$$\hat{w}_{\text{MLE}} = \operatorname{argmax}_w \sum_{i=1}^n \log P_w(y_i | x_i)$$

- Recall linear regression:  $P_w(Y = y | X = x) = \frac{1}{\sqrt{2\pi}} e^{-(y - w^\top x)^2 / 2}$

# Why not keep doing this?

# A Gaussian won't focus probability mass on labels  $y=0$  and  $y=1$

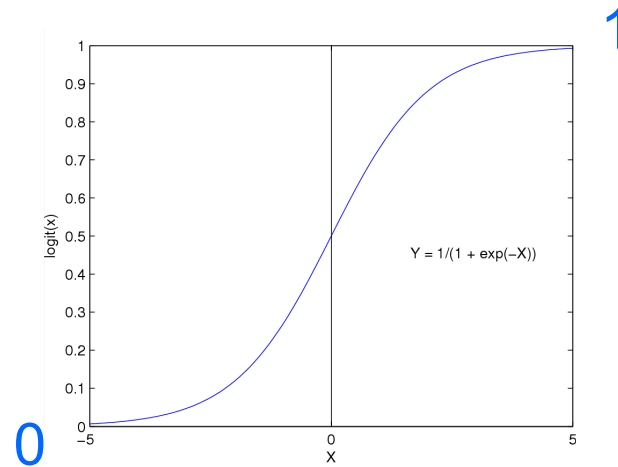
# Modeling conditional probabilities

$$\hat{w}_{\text{MLE}} = \operatorname{argmax}_w \sum_{i=1}^n \log P_w(y_i | x_i)$$

- Recall: prediction:  $E[Y | X = x] = w^\top x$
- Logistic regression uses a model specialized for (binary) classification:

$$P(Y = 1 | X = x) = \frac{1}{1 + e^{-w^\top x}}$$

$$P(Y = 0 | X = x) = \frac{e^{-w^\top x}}{1 + e^{-w^\top x}}$$



# Sigmoid function!

# Idea: pass  $w^\top x$  through sigmoid function

# How to make a binary prediction with this model? # `round(sigmoid( $w^\top x$ ))`

# Understanding the sigmoid

$$P(Y = 1 \mid X = x) = \frac{1}{1 + e^{-w^\top x}} = \sigma(w^\top x)$$

Trying to learn weights  $w$

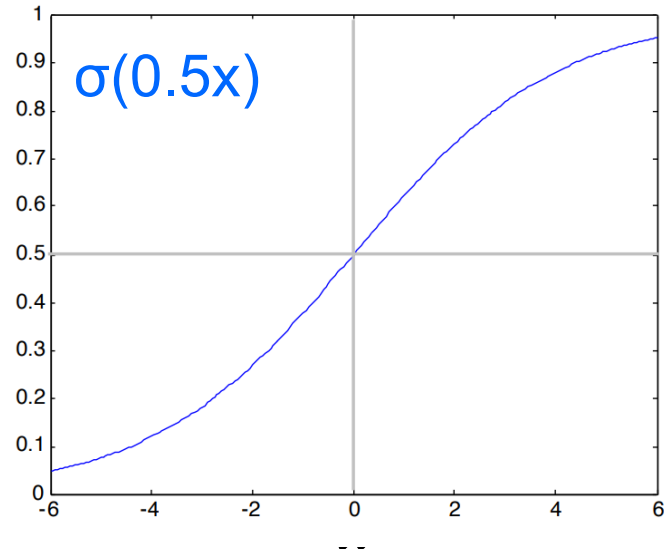
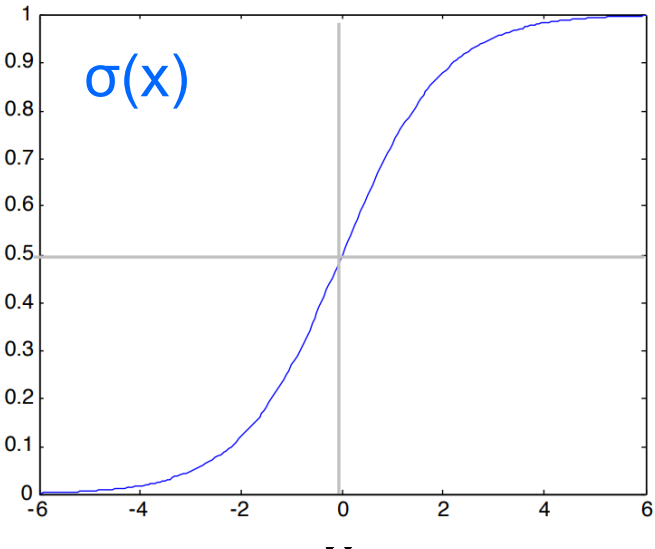
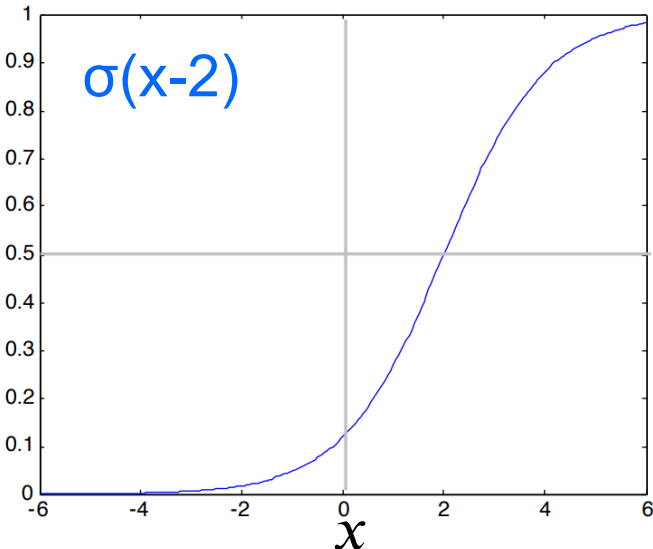
$$\sigma\left(w_0 + \sum_k w_k x_k\right) = \frac{1}{1 + e^{-w_0 + \sum_k w_k x_k}}$$

$w_0 = -2, w_1 = 1$

$w_0 = 0, w_1 = 1$

$w_0 = 0, w_1 = 0.5$

$\sigma(w^\top x)$



# Can use weights to shift # and stretch # the classification boundary

# Sigmoid for binary classification

# want to find **classification boundary**: where model switches from predicting 0  $\rightarrow$  1

- What's the shape of the decision rule  $P(Y = 1 | X) \geq P(Y = 0 | X)$ ?

$$P(Y = 1 | x) = \frac{1}{1 + e^{-w^T x}}$$

$$P(Y = 0 | x) = \frac{e^{-w^T x}}{1 + e^{-w^T x}}$$

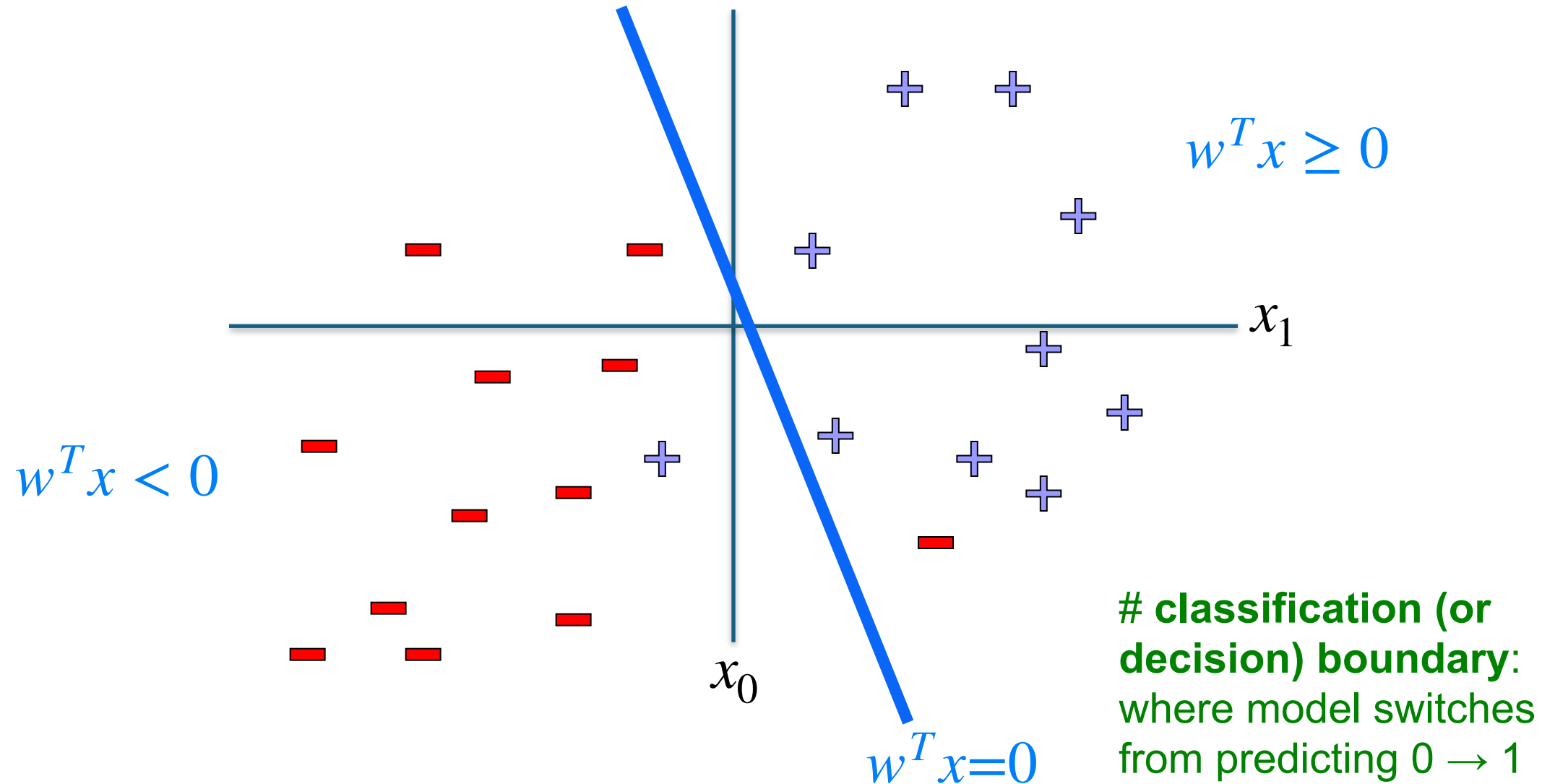
$$\frac{P(Y = 1 | x)}{P(Y = 0 | x)} = \frac{1}{e^{-w^T x}}$$

$$\begin{aligned} \log\left(\frac{P(Y = 1 | x)}{P(Y = 0 | x)}\right) &= \log \frac{1}{e^{-w^T x}} \\ &\quad \quad \quad \color{red}{1 \times d} \quad \color{red}{d \times 1} \\ &= w^T x \end{aligned}$$

**Decision rule:** is  $w^T x \geq 0$ ?

- Yes: predict  $y=1$
- No: predict  $y=0$

# Logistic regression – a linear classifier



# Logistic regression for binary classification

$$\hat{w}_{\text{MLE}} = \operatorname{argmax}_w \sum_{i=1}^n \log P_w(y_i | x_i)$$

# We've switched the labels for notation convenience

- We have i.i.d. training data  $\left\{ (x_i, y_i) \right\}_{i=1}^n$ ,  $x_i \in \mathbb{R}^d$ ,  $y_i \in \{-1, 1\}$
- $P_w(y_i | x_i) = \sigma(y_i w^\top x_i) = 1 / (1 + \exp(-y_i w^\top x_i))$
- $\ell_i(w) = -\log P_w(y_i | x_i) = \log(1 + \exp(-y_i w^\top x_i))$

# Huh?

$$y_i = 1 \quad \hat{y}_i = -1 \quad \rightarrow \quad \text{Loss} = 1$$

$$y_i = -1 \quad \hat{y}_i = 1 \quad \rightarrow \quad \text{Loss} = 1$$

$$\hat{w}_{\text{MLE}} = \operatorname{argmin}_w \sum_{i=1}^n \log(1 + \exp(-y_i w^\top x_i))$$

Logistic loss

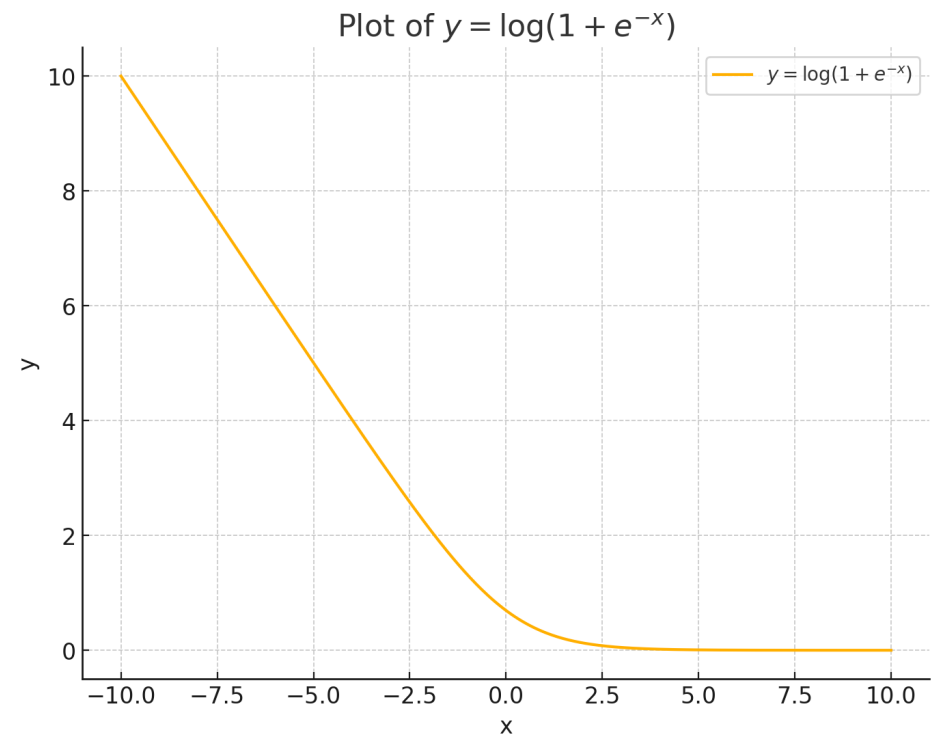
# Computing the logistic regression solution

$$\hat{w}_{\text{MLE}} = \operatorname{argmin}_w \sum_{i=1}^n \log(1 + \exp(-y_i w^\top x_i))$$

# Computing the logistic regression solution

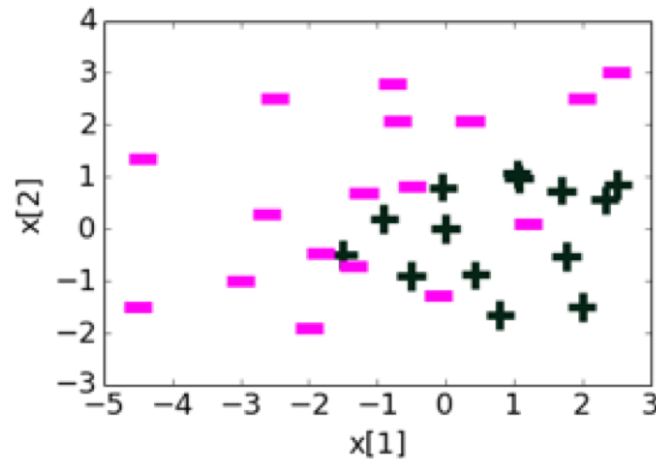
$$\hat{w}_{\text{MLE}} = \operatorname{argmin}_w \sum_{i=1}^n \log(1 + \exp(-y_i w^\top x_i))$$

- No closed form...
- But, this is a smooth convex problem!
- Optimize via gradient descent



# Example: Adding more polynomial features

- $x \in \mathbb{R}^2$ ,  $y \in \{-1, +1\}$
- Features: Polynomials
- Model: Linear on polynomial features

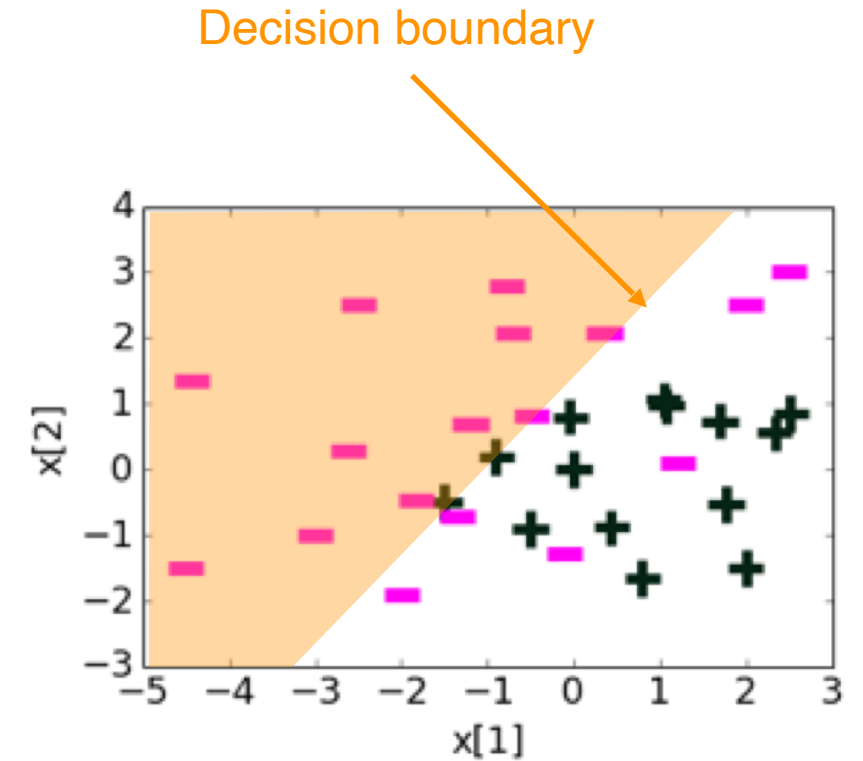
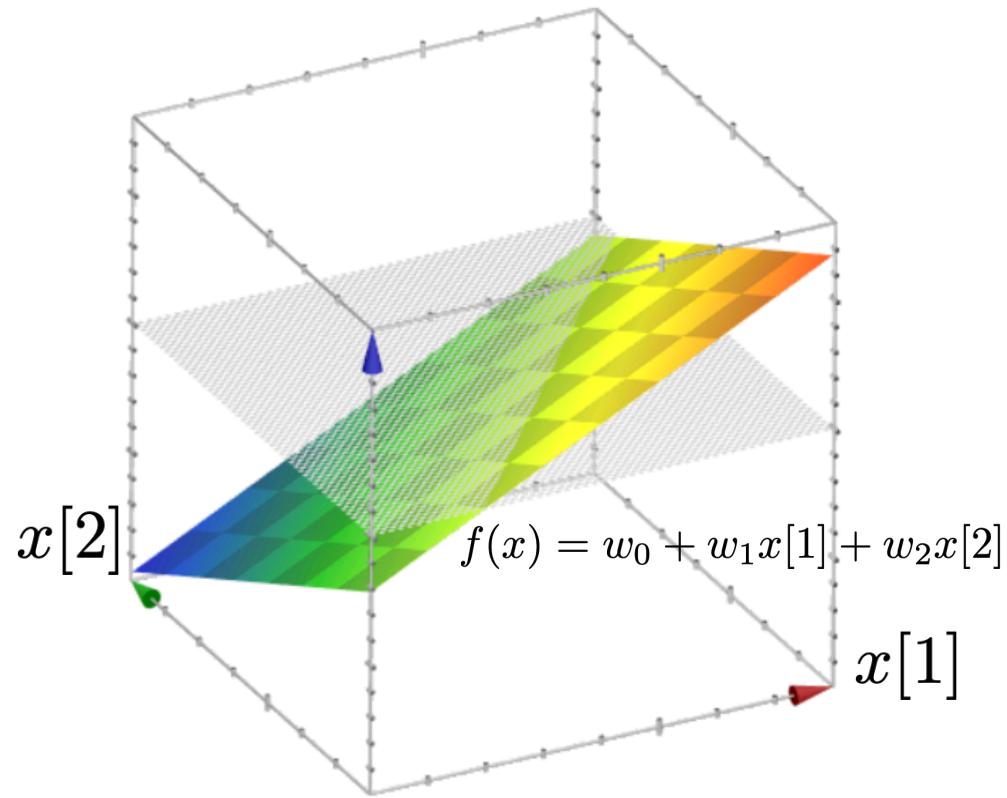


Polynomial features

$$\begin{bmatrix} h_0(x) = 1 \\ h_1(x) = x[1] \\ h_2(x) = x[2] \\ h_3(x) = x[1]^2 \\ h_4(x) = x[2]^2 \\ \vdots \end{bmatrix}$$

$$f(x) = w_0 h_0(x) + w_1 h_1(x) + w_2 h_2(x) + \dots$$

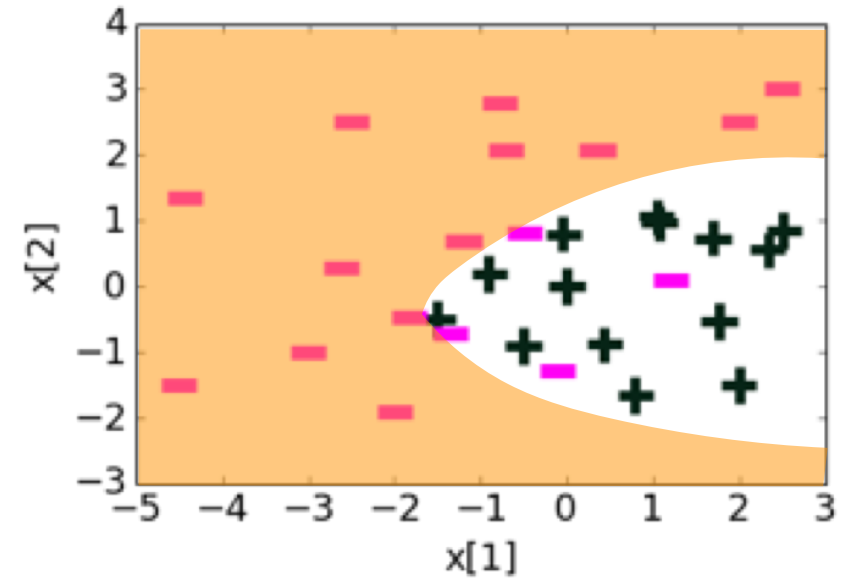
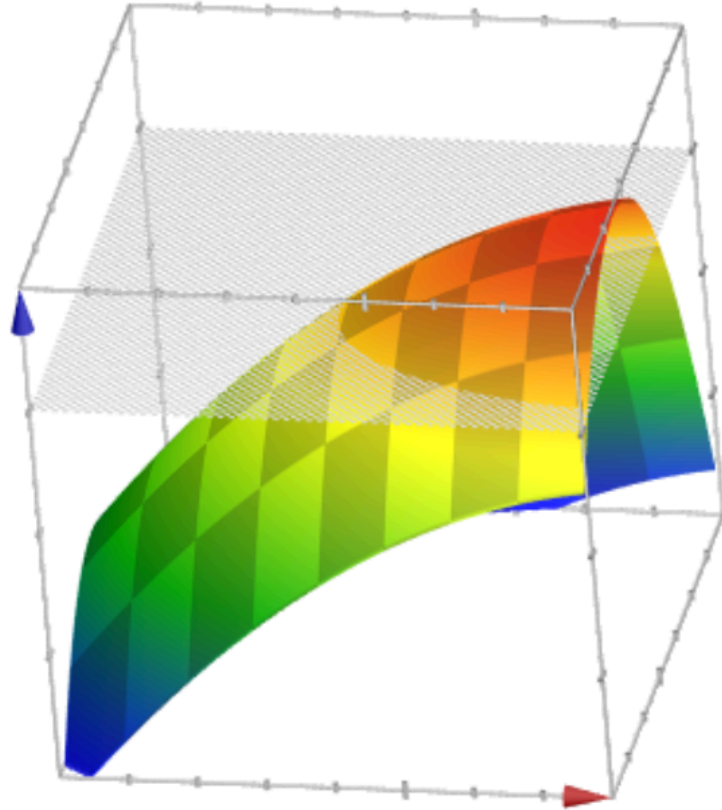
# Linear features



- Simple regression models → smooth predictors
- Simple classifier models → smooth decision boundaries

| Feature  | Value  | Coefficient |
|----------|--------|-------------|
| $h_0(x)$ | 1      | 0,23        |
| $h_1(x)$ | $x[1]$ | 1,12        |
| $h_2(x)$ | $x[2]$ | -1,07       |

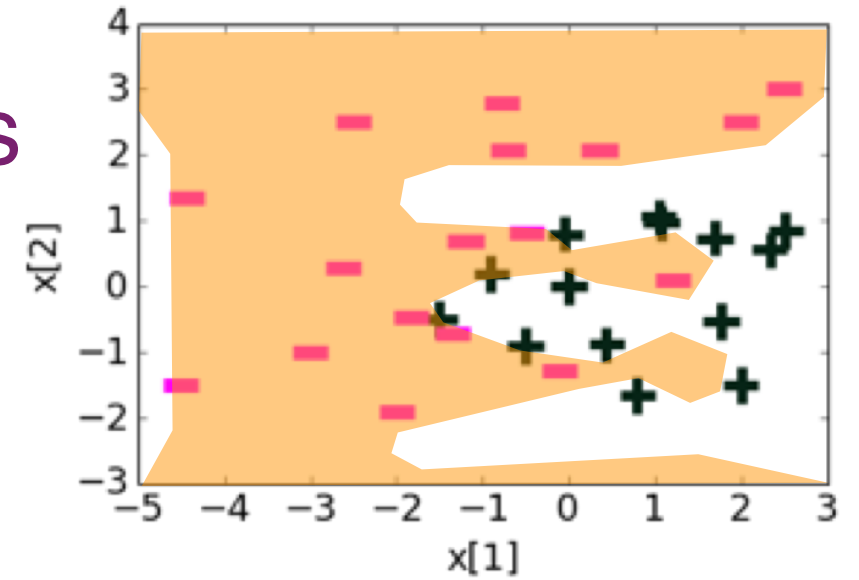
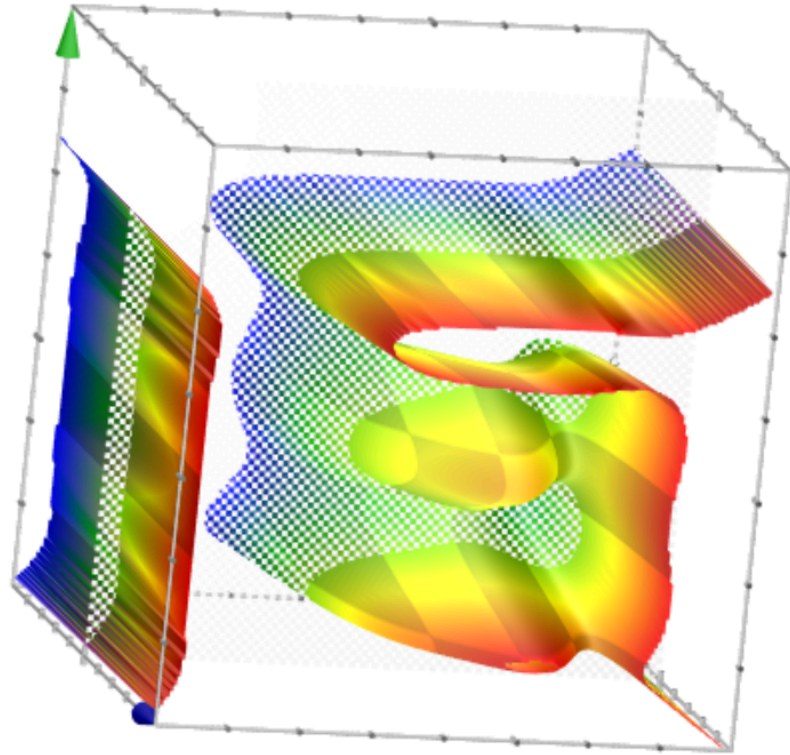
# Quadratic features



- Adding more features gives more complex models
- Decision boundary becomes more complex

| Feature  | Value      | Coefficient |
|----------|------------|-------------|
| $h_0(x)$ | 1          | 1,68        |
| $h_1(x)$ | $x[1]$     | 1,39        |
| $h_2(x)$ | $x[2]$     | -0,59       |
| $h_3(x)$ | $(x[1])^2$ | -0,17       |
| $h_4(x)$ | $(x[2])^2$ | -0,96       |
| $h_5(x)$ | $x[1]x[2]$ | Omitted     |

# Higher-degree polynomial features



- Overfitting leads to poor generalization

| Feature     | Value      | Coefficient learned |
|-------------|------------|---------------------|
| $h_0(x)$    | 1          | 21.6                |
| $h_1(x)$    | $x[1]$     | 5.3                 |
| $h_2(x)$    | $x[2]$     | -42.7               |
| $h_3(x)$    | $(x[1])^2$ | -15.9               |
| $h_4(x)$    | $(x[2])^2$ | -48.6               |
| $h_5(x)$    | $(x[1])^3$ | -11.0               |
| $h_6(x)$    | $(x[2])^3$ | 67.0                |
| $h_7(x)$    | $(x[1])^4$ | 1.5                 |
| $h_8(x)$    | $(x[2])^4$ | 48.0                |
| $h_9(x)$    | $(x[1])^5$ | 4.4                 |
| $h_{10}(x)$ | $(x[2])^5$ | -14.2               |
| $h_{11}(x)$ | $(x[1])^6$ | 0.8                 |
| $h_{12}(x)$ | $(x[2])^6$ | -8.6                |

# Overfitting

$$\hat{w}_{\text{MLE}} = \operatorname{argmin}_w \sum_{i=1}^n \log(1 + \exp(-y_i w^\top x_i))$$

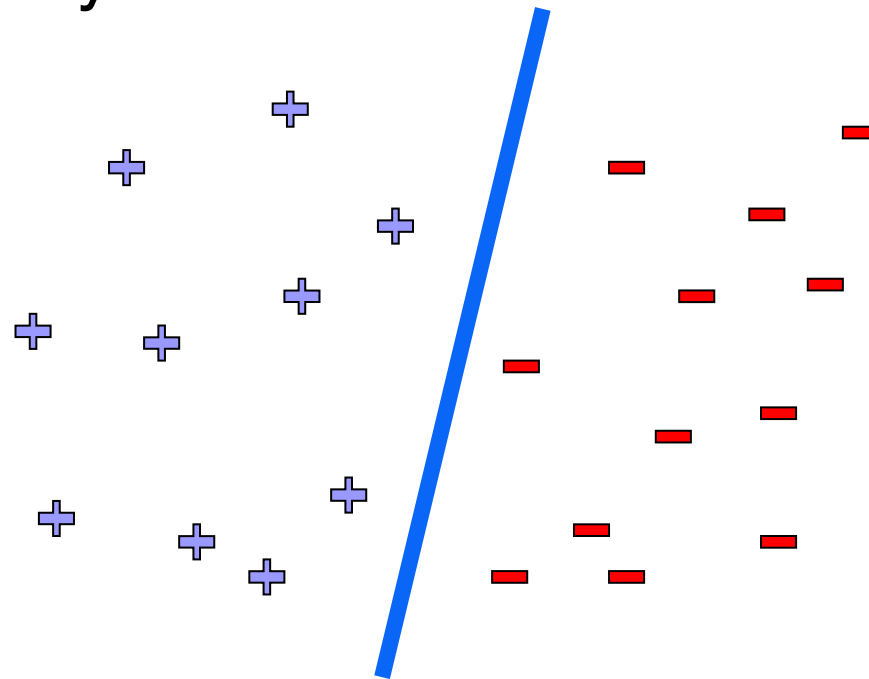
$y_i \in \{-1, 1\}$

# If we've classified  $x_i$  correctly, then we can further minimize loss by increasing magnitude of  $w$ !

- When would this badly overfit?

#  $w^\top x \geq 0 \rightarrow$  predict  $y=1$

# Trying to get  $w^\top x$  to match  $\operatorname{sign}(y)$



# Overfitting and linear separability

$$\hat{w}_{\text{MLE}} = \operatorname{argmin}_w \sum_{i=1}^n \log(1 + \exp(-y_i w^\top x_i))$$

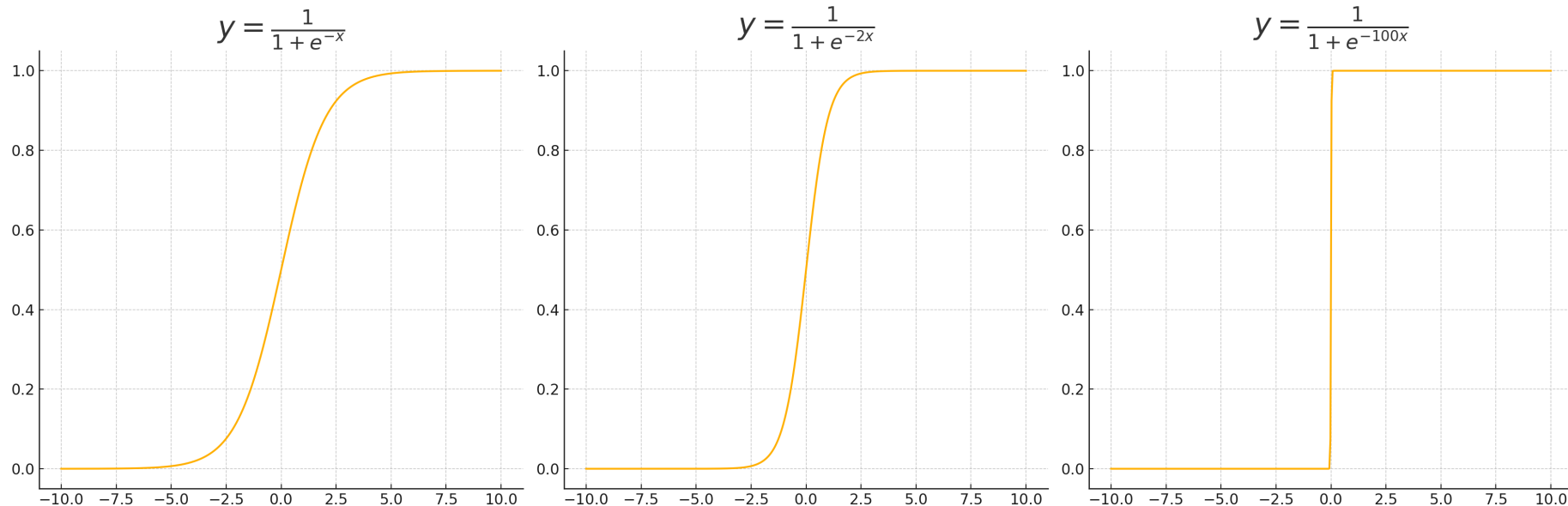
# Why do we not like a large  $w$ ?

# **Large  $w \rightarrow$  high variance**

(Small change in input leads to drastic change in prediction)

(Confidently incorrect)

- When data is linearly separable,  $\|w\| \rightarrow \infty$



# So how to fix?

# Regularized logistic regression

$$\hat{w}_{\text{reg}} = \operatorname{argmin}_w \sum_{i=1}^n \log(1 + \exp(-y_i w^\top x_i))$$

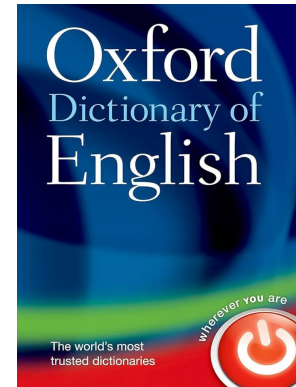
$+ \lambda \|w\|_2^2$  Ridge  
 $+ \lambda \|w\|_1$  LASSO

When data is linearly separable given the model class, this prevents:

$$\|w\| \rightarrow \infty$$

# Multi-class classification

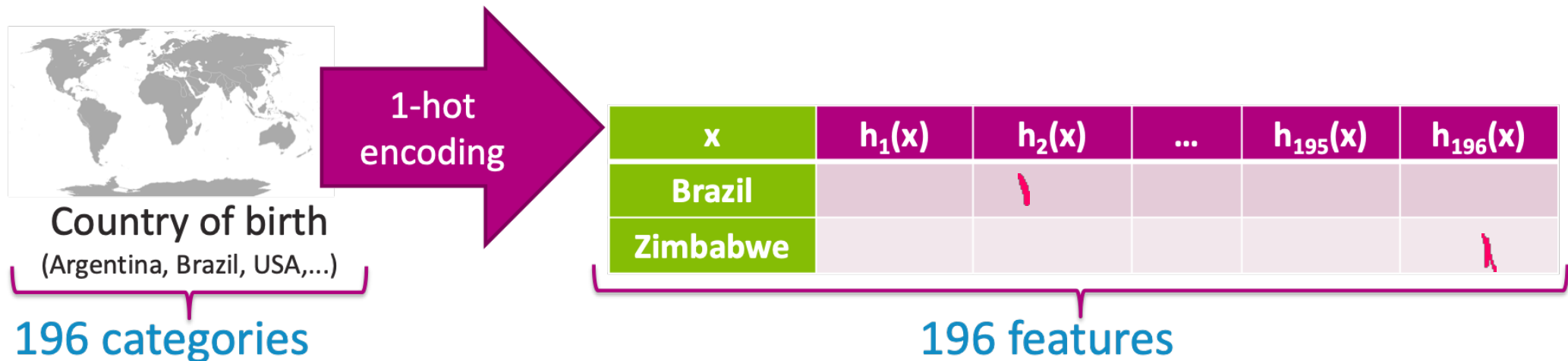
- So far: binary  $y \in \{-1, +1\}$
- In general:  $y \in \{c_1, c_2, \dots, c_k\}$
- $c_j$ 's are called classes or labels



- A k-class classifier predicts  $y$  given  $x$

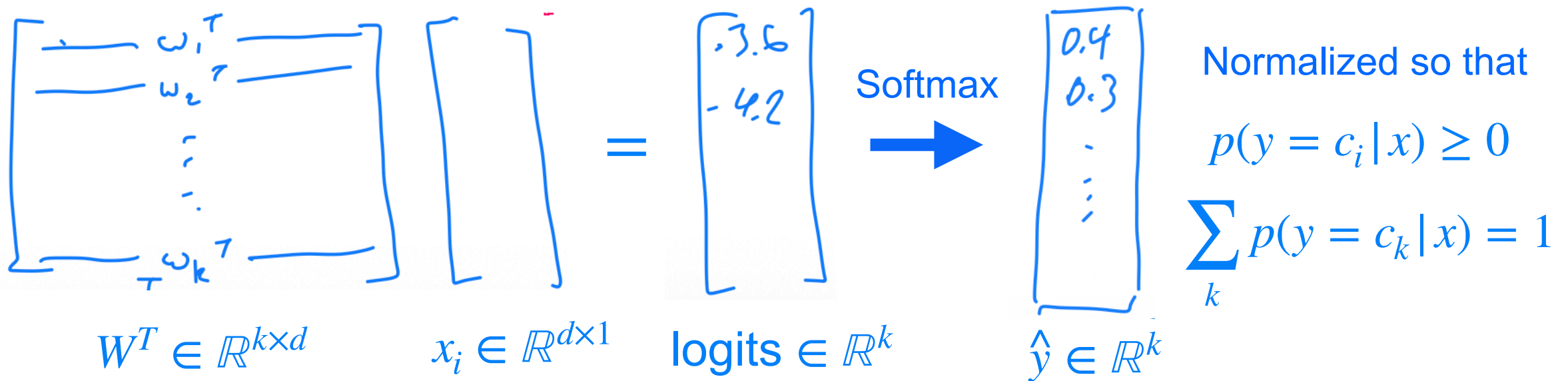
# Encoding categorical labels $c_j$

- For optimization, we need to embed raw  $c_j$  into real-valued vectors
- We typically use one-hot embeddings (a.k.a. one-hot encodings)
  - Each class is a standard basis vector in  $\mathbb{R}^k$



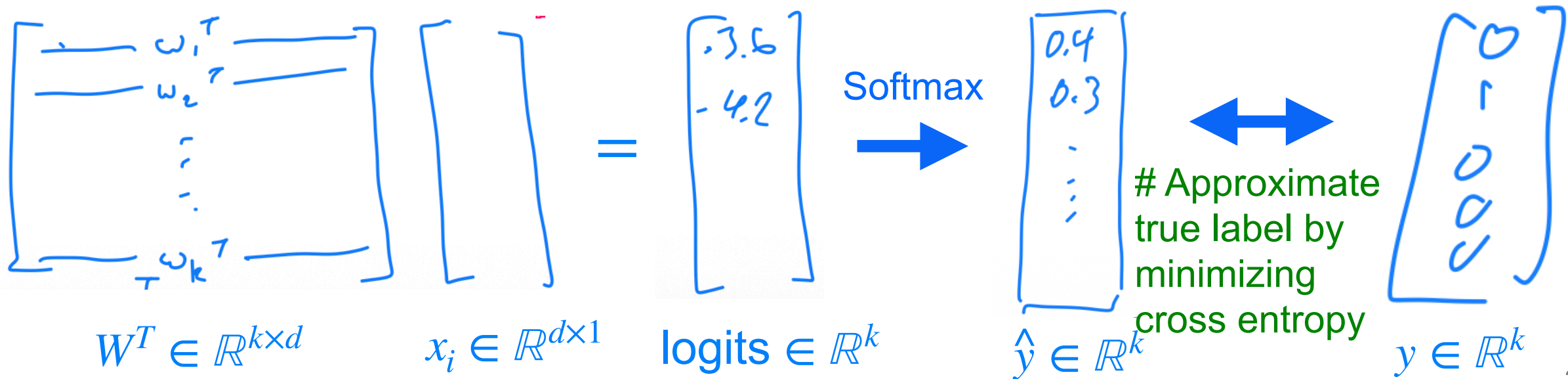
# Multi-class logistic regression (aka softmax classification)

- Data: features  $x_i \in \mathbb{R}^d$ , categorical  $y \in \{c_1, \dots, c_k\}$  # d features, k classes
- One-hot encoding  $\in \mathbb{R}^k$  s.t.  $y = [1, 0, 0, \dots]$  implies  $y = c_1$
- Model: linear prediction  $\hat{y}_i = f(x_i) = \text{softmax}(w^\top x) \in \mathbb{R}^k$
- Parameter matrix  $w \in \mathbb{R}^{d \times k}$



# Multi-class logistic regression (aka softmax classification)

- Data: features  $x_i \in \mathbb{R}^d$ , categorical  $y \in \{c_1, \dots, c_k\}$  # d features, k classes
- One-hot encoding  $\in \mathbb{R}^k$  s.t.  $y = [1, 0, 0, \dots]$  implies  $y = c_1$
- Model: linear prediction  $\hat{y}_i = f(x_i) = \text{softmax}(w^\top x) \in \mathbb{R}^k$
- Parameter matrix  $w \in \mathbb{R}^{d \times k}$  # How to train?



# Softmax classification

without loss of generality, set  $k = 2$ ,  $w_1 = 0$

2 classes

$k$  classes

# Sigmoid

Conditional probabilities

# Softmax

$$P(y = 1 | x) = \frac{1}{1 + e^{-w^\top x}} = \frac{e^{w^\top x}}{1 + e^{w^\top x}}$$

$$P(y = -1 | x) = \frac{1}{1 + e^{w^\top x}}$$

$$P(y = c_j | x) = \frac{e^{w_j^\top x}}{\sum_{j'} e^{w_{j'}^\top x}}$$

MLE

$$\operatorname{argmax}_w \sum_{i=1}^n \log \left( \frac{1}{1 + e^{-y_i w^\top x_i}} \right)$$

# Logistic loss

$$\operatorname{argmax}_w \sum_{i=1}^n \sum_{j=1}^k 1\{y_i = c_j\} \log \left( \frac{e^{w_j^\top x_i}}{\sum_{j'=1}^k e^{w_{j'}^\top x_i}} \right)$$

# Binary cross entropy

# Regression and classification

- ML paradigm: define prediction  $f_w(x)$  and loss  $\ell(f_w(x), y)$
- Then optimize:

$$\hat{w} = \operatorname{argmin}_w \sum_{i=1}^n \ell(f_w(x_i), y_i)$$

- Squared error loss:  $\ell(f_w(x), y) = (y - f_w(x))^2$
- Logistic loss:  $\ell(f_w(x), y) = \log(1 + \exp(-yf_w(x)))$

# Regression and classification

- Can we treat classification as a regression problem?
  - # Yes: could regress to a continuous output  $y$  and round to a valid label
  - # Is this a good idea? No, no smooth relationship between label values
- Can we treat regression as a classification problem?
  - # Yes: classify which “bin” the true number falls into:  $c_1 = [0-10]$ ,  $c_2 = [11-20]$ , ...
  - # Is this a good idea? You lose some precision, but it actually can be. Sometimes easier to optimize. Can win a kaggle competition this way. Look into “distributional RL”

# Regression and classification



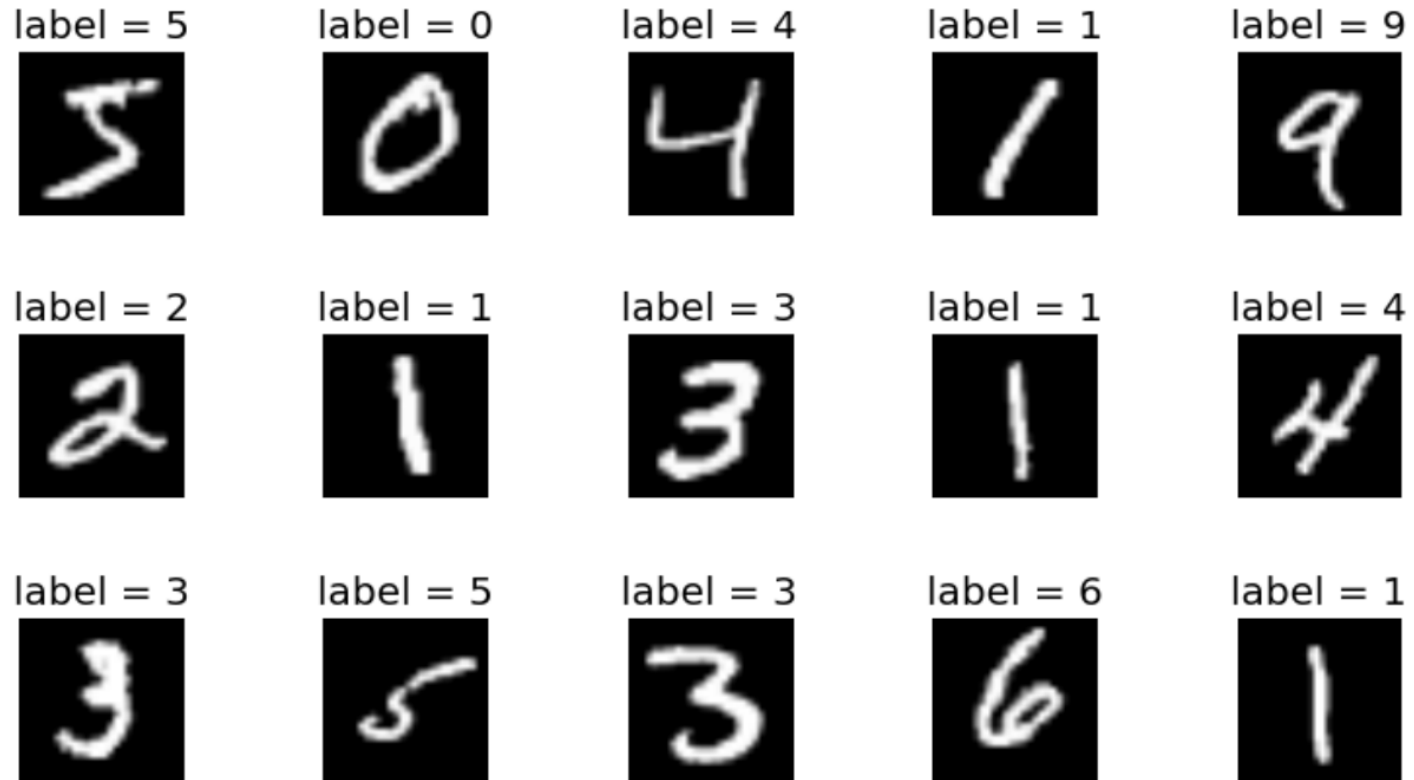
→ Temperature: 62F

# Regression and classification

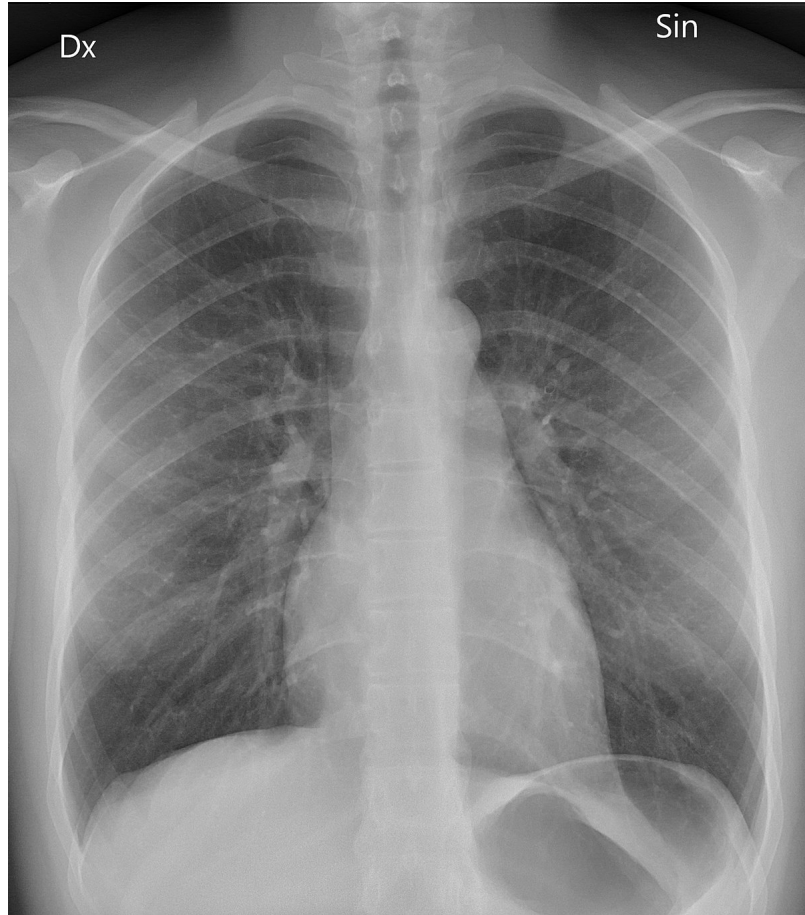


98105

# Regression and classification



# Multi-class vs multi-label classification



# What if multiple labels can be true at the same time?



Healthy  
Pneumonia  
Pneumothorax  
Pleural Effusion  
...

# Can you still use softmax? No

# Can use multiple sigmoids

# Summary

- Classification problems (where  $y$  is categorical) are everywhere
- Regression losses are not typically appropriate
- Logistic regression: model conditional probability  $P(y | x)$  as sigmoid (then apply usual MLE machinery)
- Softmax classification: generalized to  $k > 2$  classes
- Regularization is still important

# Recap: The ML pipeline

1. Define the **task** (what type of data, what type of eval metrics?)
2. Collect and preprocess **data** # Feature engineering, polynomial degree
3. Choose **model** family/parameterization # So far,  $w^T x$
4. Choose **training loss**
4. For each choice of hyperparameters:
  - **Optimize** model (minimize loss) on training data
  - **Evaluate** on validation data
5. Pick best hyperparameters according to validation performance
6. **Evaluate** final model on test data # Keep this held out

# The ML pipeline

1. Task
2. Data
3. Model family
4. Training loss
4. Optimize
5. Pick hyperparameters
6. Evaluate