Over-confident predictions

Sewoong Oh

CSE/STAT 416
University of Washington

Maximum Likelihood Estimation

Galton board

- Which Gaussian distribution best explains the data?
- Mean and Variance?

Maximum Likelihood Estimation (MLE)

- MLE is used to find the **probabilistic model** that is most likely to have generated the observed data
- In works for both **continuous** data and **categorial** or **discrete** data
- As we are studying classification, we focus on discrete data
- Let's start with simplest case where there is no ***x*** and we just observe discrete valued ***y***
- We will learn how to use MLE to find the best model, via examples

Example: coin with unknown bias *p*

- Data: ***y1,y2***,….either H or T

$$100 \text{ trials and } 78 \text{ H's and } 22 \text{ T's}$$

- Model class:
  - Coin flip ***Y*** with probability ***p*** of being H, for some ***p*** in [0,1]

$$\mathbb{P}(\hat{y}_i = H) = p$$
$$\mathbb{P}(\hat{y}_i = T) = 1 - p$$

- Goal: find the model parameter ***p*** that is most likely to have generated the data
- There are many ways to do it
- MLE is on of the most popular, and principled way
- MLE finds the model parameter that maximizes the likelihood

$$\text{maximize}_p \;\; \mathbb{P}(\text{observed data} \,|\, p \text{ is ground truth})$$

- We assume each data is ***independently*** collected, so

$$\text{maximize}_p \;\; \prod_{i=1}^{N} \mathbb{P}(\hat{y}_i = y_i \,|\, p \text{ is ground truth})$$

- We typically take logarithmic function of the objective and normalize by ***N***

$$\text{maximize}_p \quad \frac{1}{N}\sum_{i=1}^{N} \log\left(\mathbb{P}(\hat{y}_i = y_i \,|\, p \text{ is ground truth})\right)$$

- log(P(xi|model)) is called ***likelihood*** , hence the name maximum likelihood estimation
- For the coin example, it is

$$\boxed{\text{maximize}_p \quad \frac{N_H}{N}\log p + \frac{N_T}{N}\log(1-p)}$$

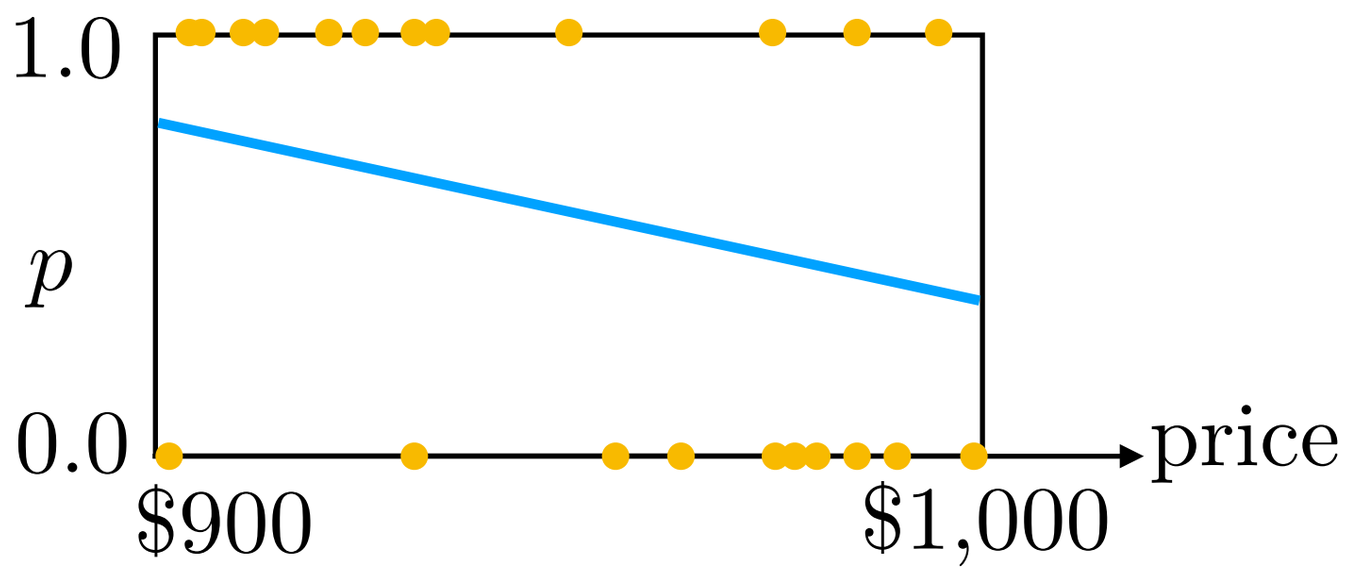
- The optimal solution is

$$p^* \;=\; \frac{N_H}{N}$$

- As the expected number of H is ***p***, we are matching the expectation, which makes sense

Example: demand curve estimation

- Data:
  - ***x1,x2***,… : price of the item (changing over time)
  - ***y1,y2***,… : 1 if customer bought it, 0 if customer did not



- Model class:
  - ***y*** is coin flip with bias $p = w_0 + w_1 x$

$$\mathbb{P}(\hat{y}_i = 1) = w_0 + w_1 x_i$$
$$\mathbb{P}(\hat{y}_i = 0) = 1 - w_0 - w_1 x_i$$

- What is the MLE?

$$\text{maximize}_{w_0, w_1} \quad \frac{1}{N} \sum_{i=1}^{N} \left\{ y_i \log(w_0 + w_1 x_i) + (1 - y_i) \log(1 - w_0 - w_2 x_i) \right\}$$

Probabilistic interpretation of logistic regression

- Data: ***x1,x2***,… in any dimension
  y1,y2,… in {+1,-1}
- Model:
  - Logistic model

$$\mathbb{P}(\hat{y}_i = -1) = \frac{1}{1 + e^{(w_0 + w_1 x[1] + w_2 x[2] + \cdots)}}$$

$$\mathbb{P}(\hat{y}_i = +1) = \frac{1}{1 + e^{-(w_0 + w_1 x[1] + w_2 x[2] + \cdots)}}$$

- What is the MLE?

$$w^T x = w_0 + w_1 x[1] + w_2 x[2] + \cdots$$

$$\text{maximize}_w \quad \frac{1}{N} \sum_{i=1}^{N} \log\left(\frac{1}{1 + e^{-y_i w^T x}}\right)$$

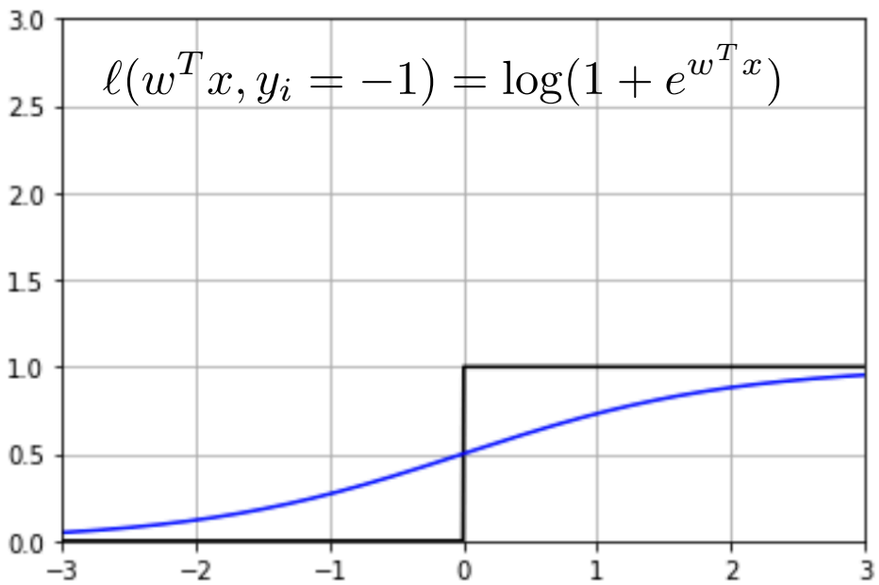
Logistic regression is MLE

- MLE written in a different form

$$\text{maximize}_w \quad \frac{1}{N}\sum_{i:y_i=-1}\log\left(\frac{1}{1+e^{w^T x}}\right)+\frac{1}{N}\sum_{i:y_i=1}\log\left(\frac{1}{1+e^{-w^T x}}\right)$$

- Notice that this is exactly the **logistic regression** without any regularizers and with ***k***=1

$$\text{minimize}_w \quad \frac{1}{N}\sum_{i:y_i=-1}\log\left(1+e^{w^T x}\right)+\kappa\frac{1}{N}\sum_{i:y_i=1}\log\left(1+e^{-w^T x}\right)$$

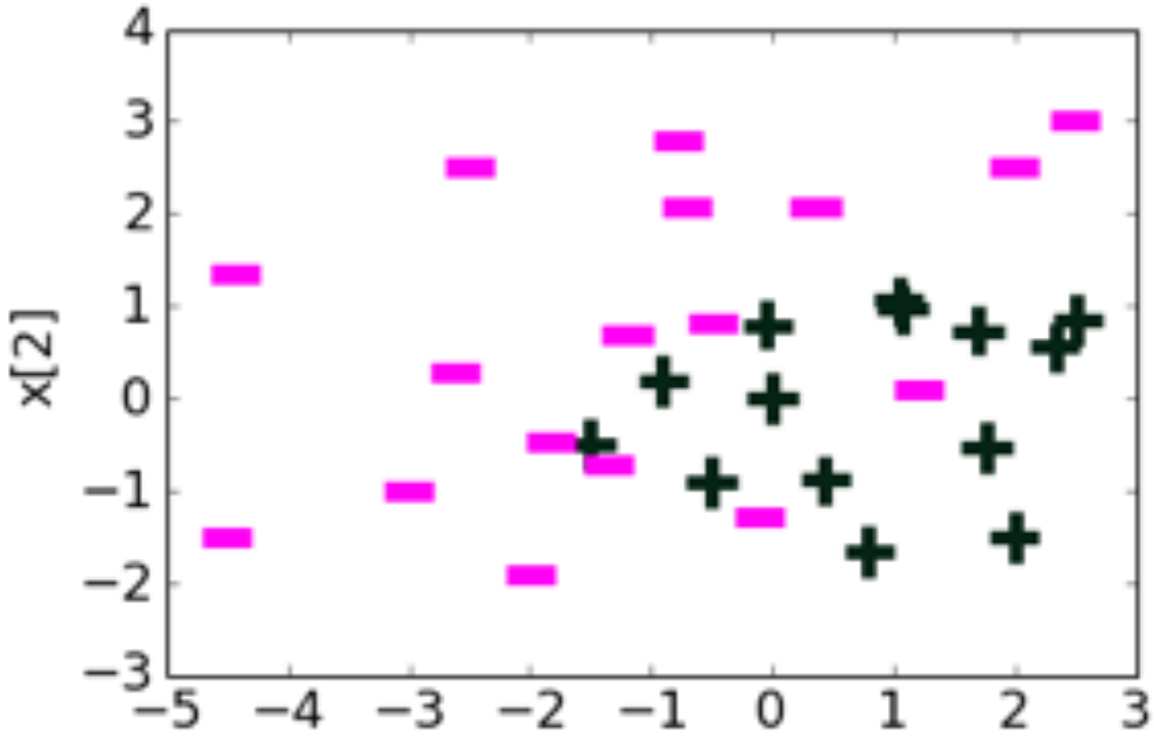


$w^T x$

$y_i = -1$

Overfitting in classification

Example: adding more polynomial features



- features

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

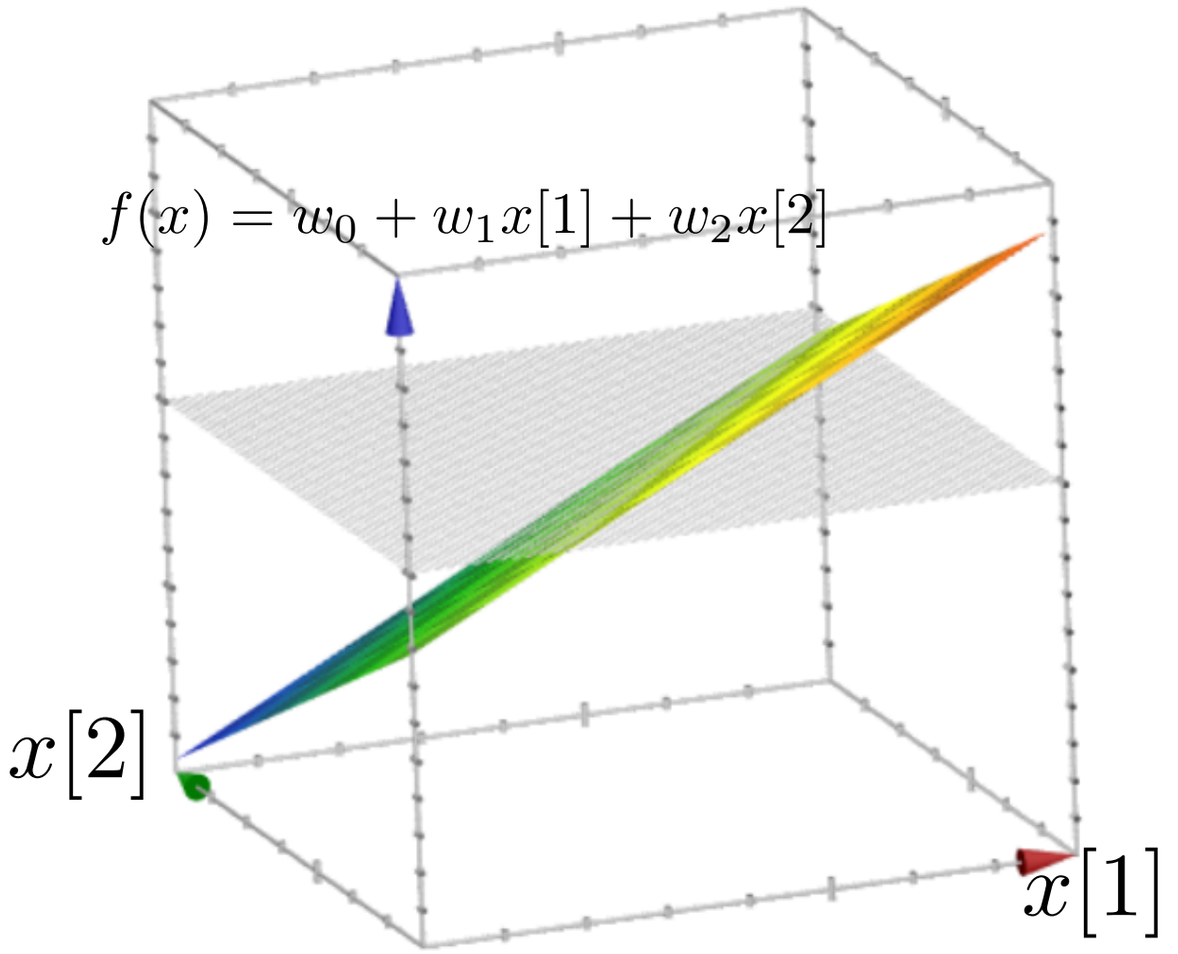
- data: ***x*** in 2-dimensions, ***y*** in {+1,-1}
- features: polynomials
- model: linear

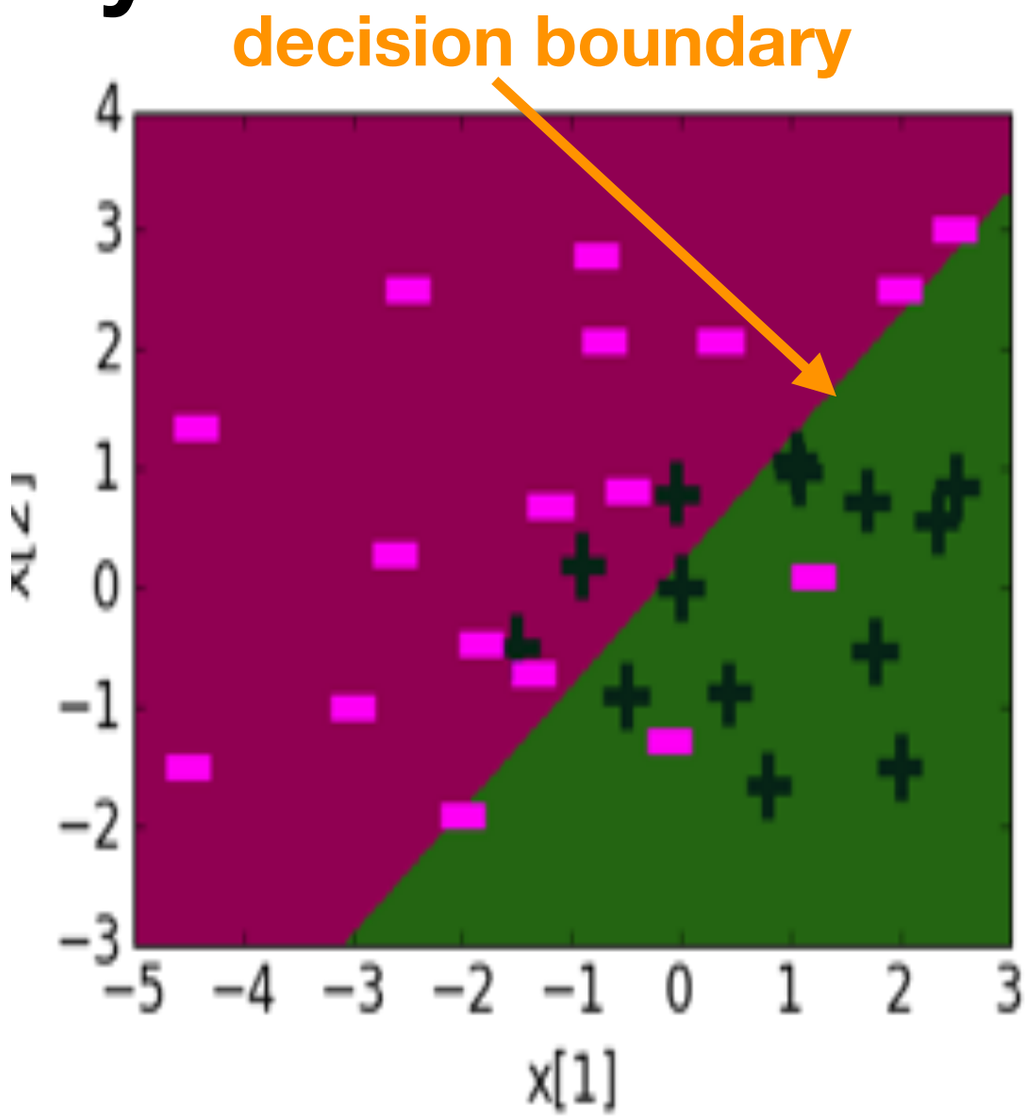
$$f(x) = w_0 h_0(x) + w_1 h_1(x) + w_2 h_2(x) + \cdots$$

- quality metric: logistic regression

$$\text{minimize}_w \quad \frac{1}{N} \sum_{i=1}^{N} \ell(\underbrace{f(x_i)}_{\hat{y}_i}, y_i) \qquad \ell(\hat{y}_i, y_i) = \log(1 + e^{-y_i \hat{y}_i})$$

Learned decision boundary



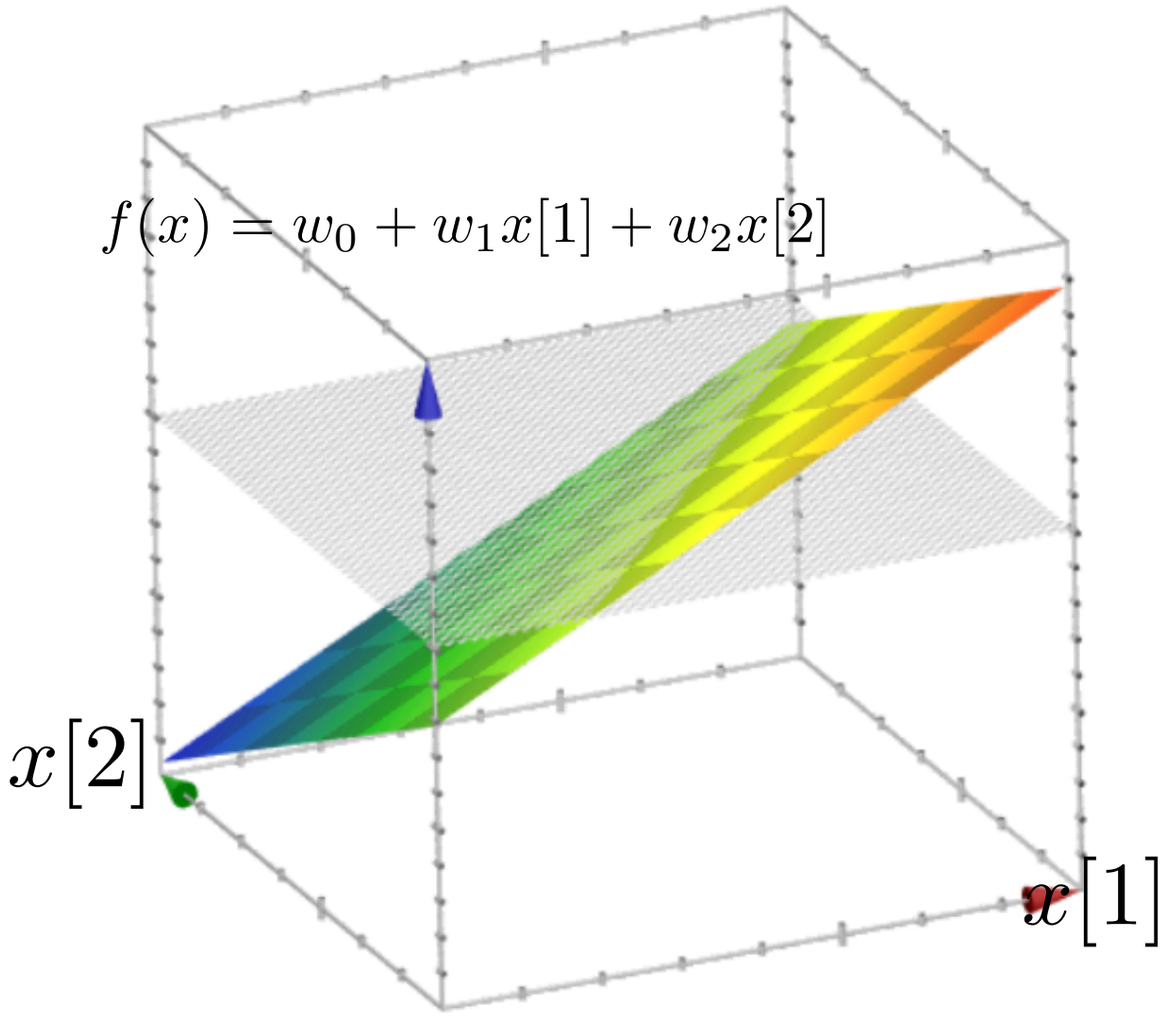


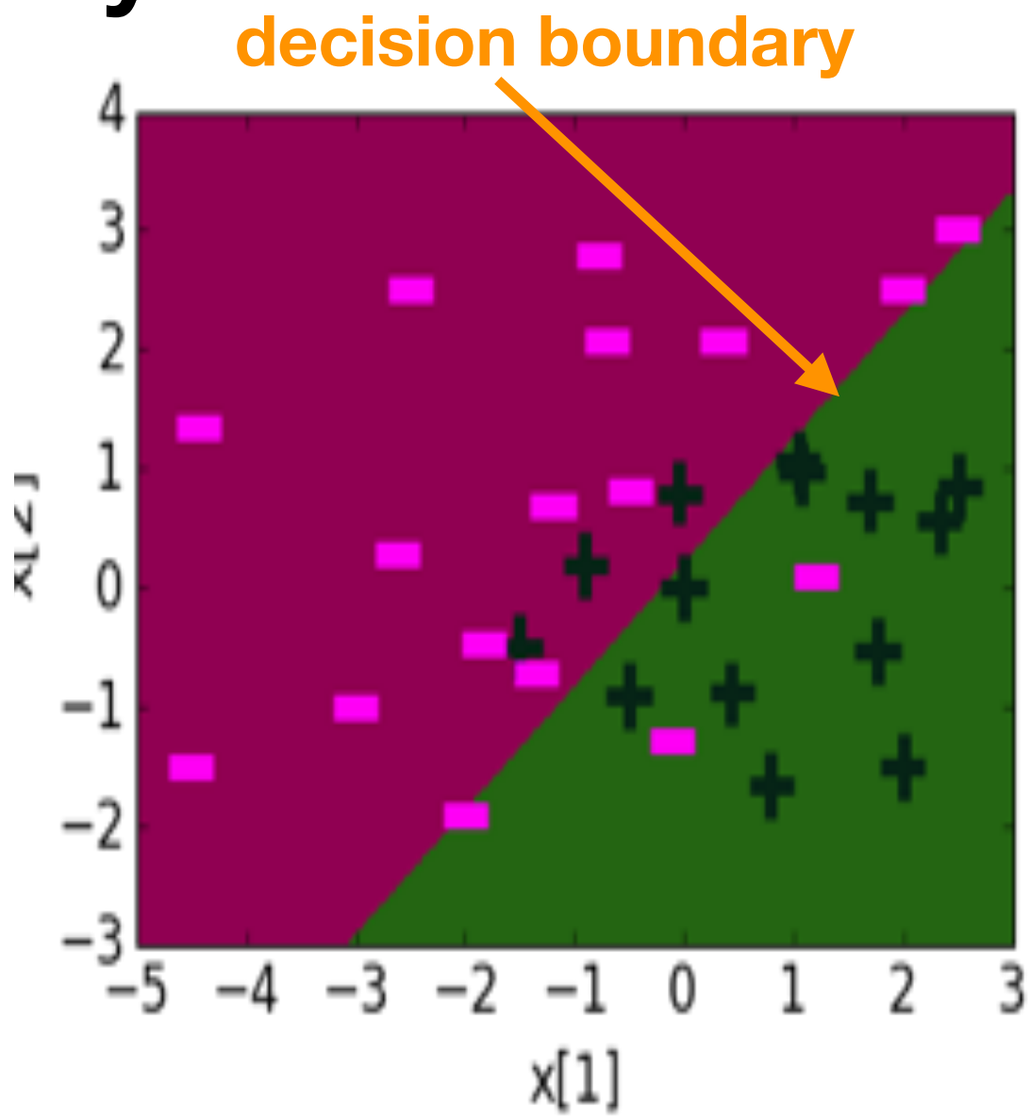
3-d view

Feature	Value	Coefficient
$h_0(x)$	1	0.23
$h_1(x)$	x[1]	1.12
$h_2(x)$	x[2]	-1.07

- Simple **regression** models had **smooth** predictors
- Simple **classifier** models have **smooth** decision boundaries

Learned decision boundary



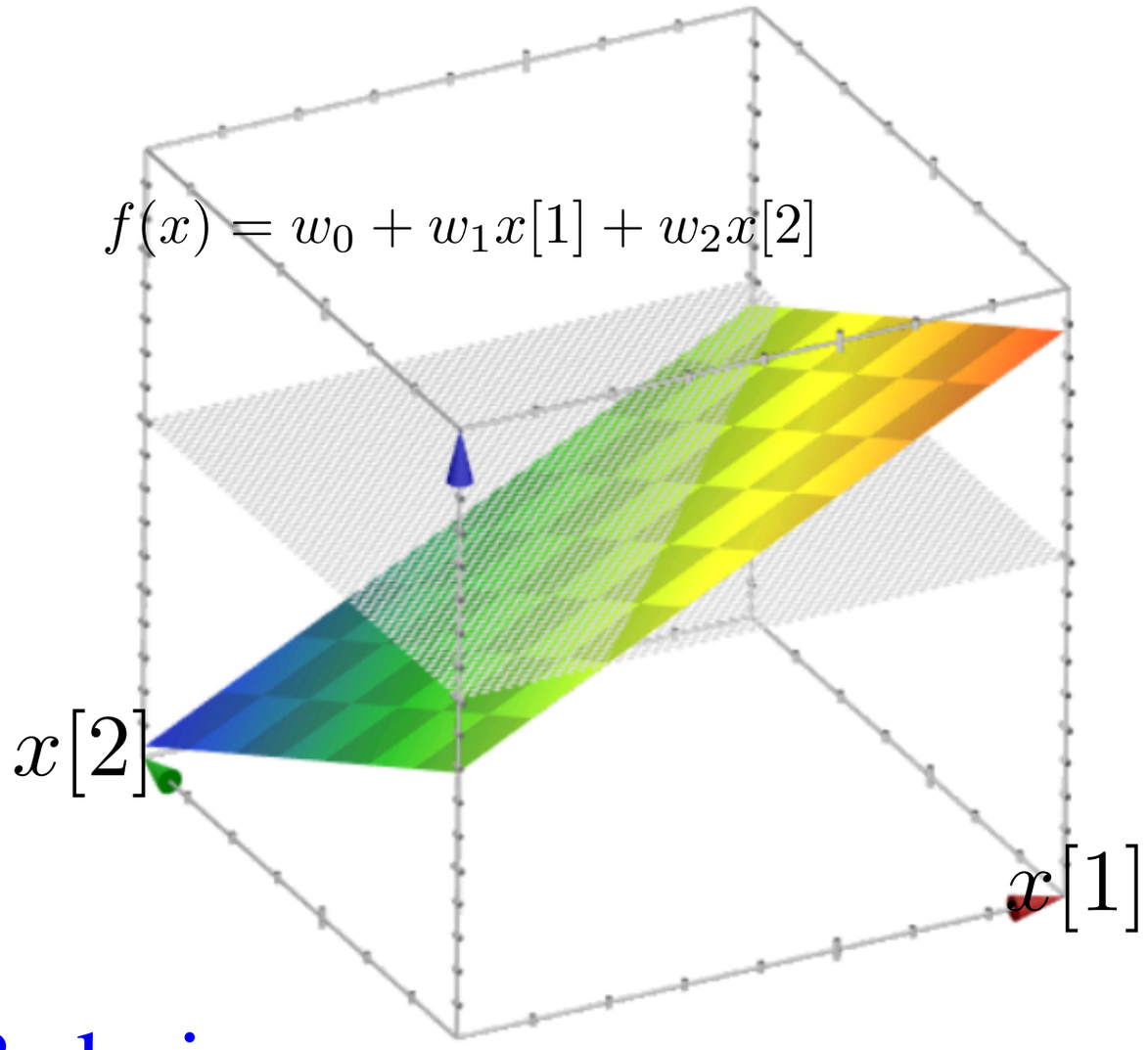


3-d view

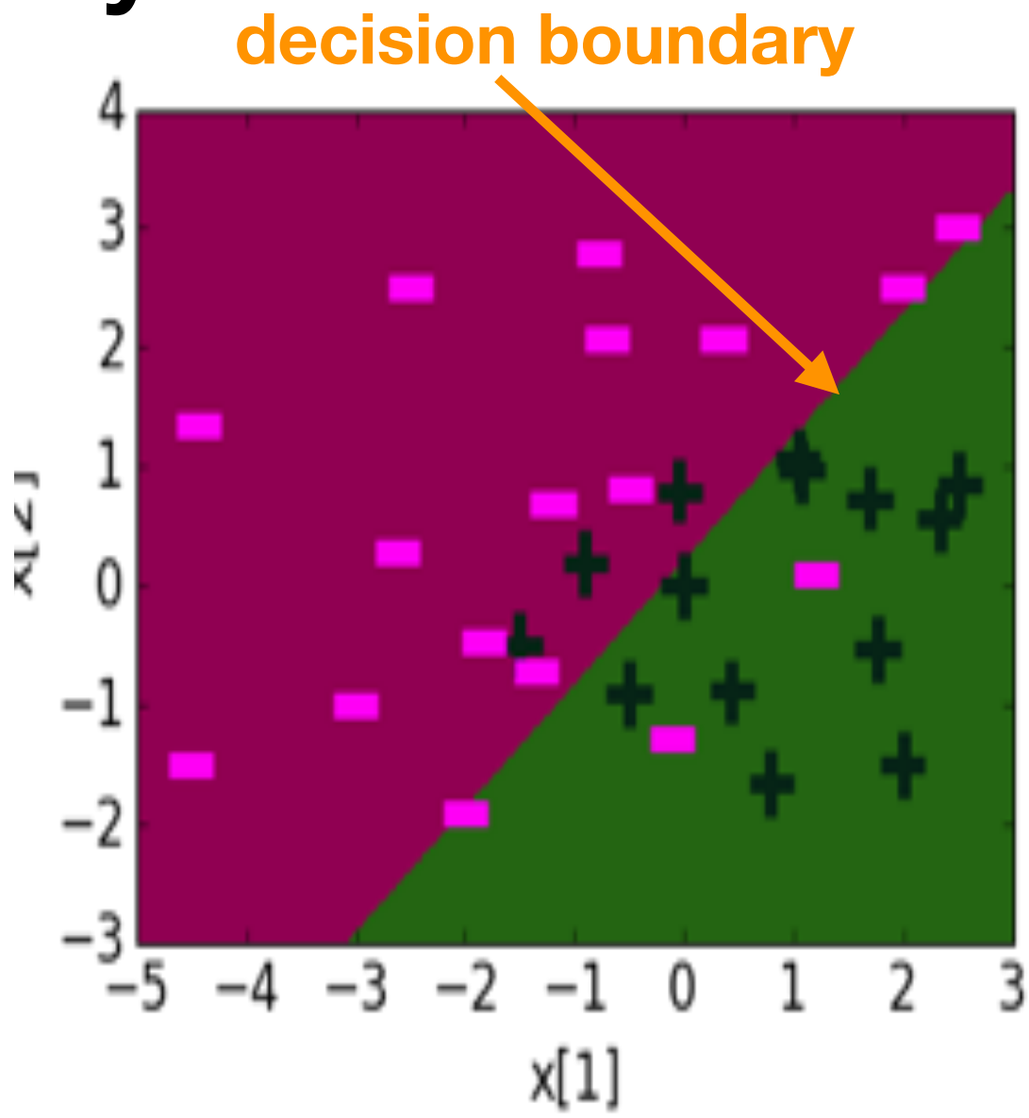
Feature	Value	Coefficient
$h_0(x)$	1	0.23
$h_1(x)$	x[1]	1.12
$h_2(x)$	x[2]	-1.07

- Simple **regression** models had **smooth** predictors
- Simple **classifier** models have **smooth** decision boundaries

Learned decision boundary



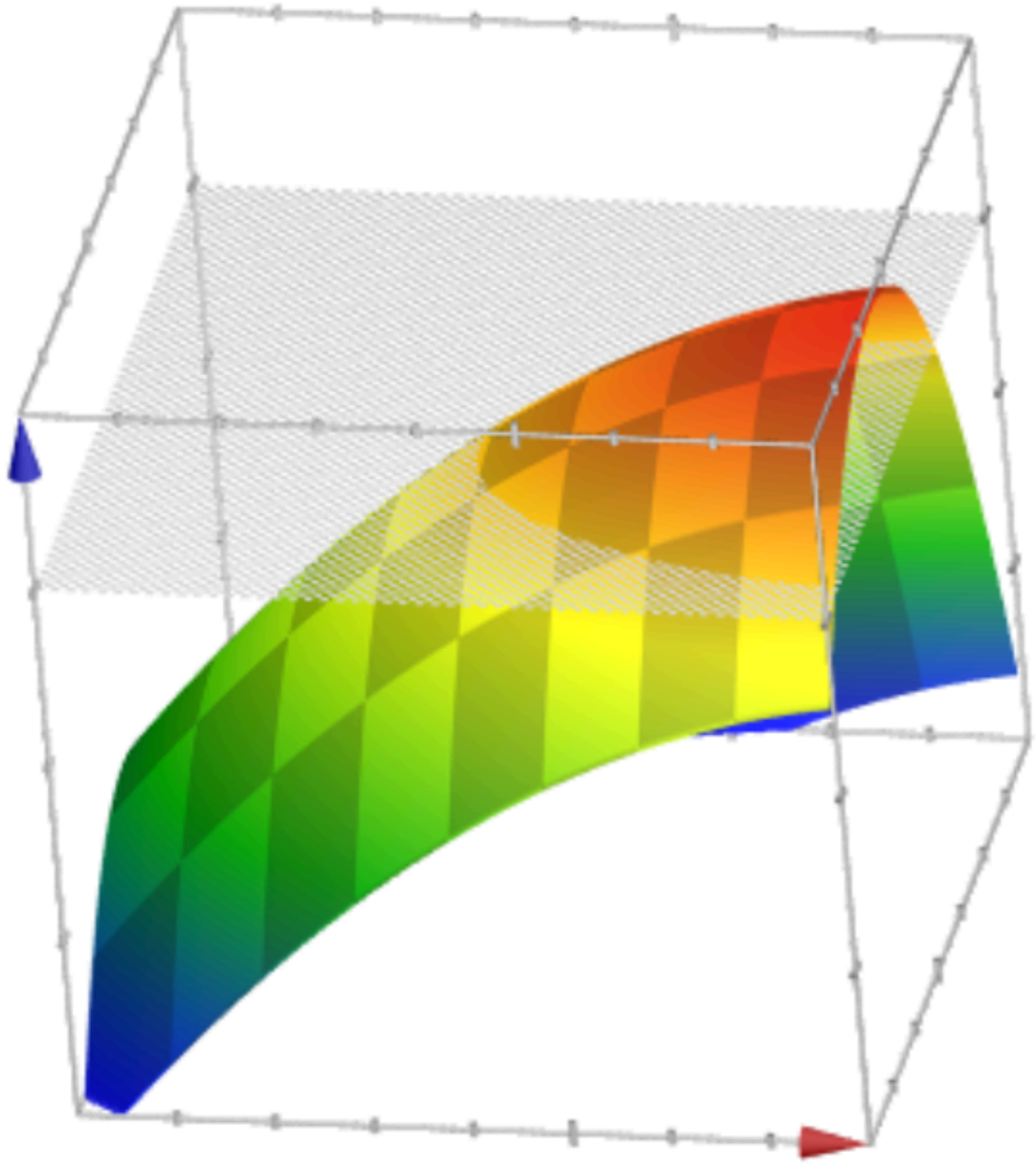
[3-d view]

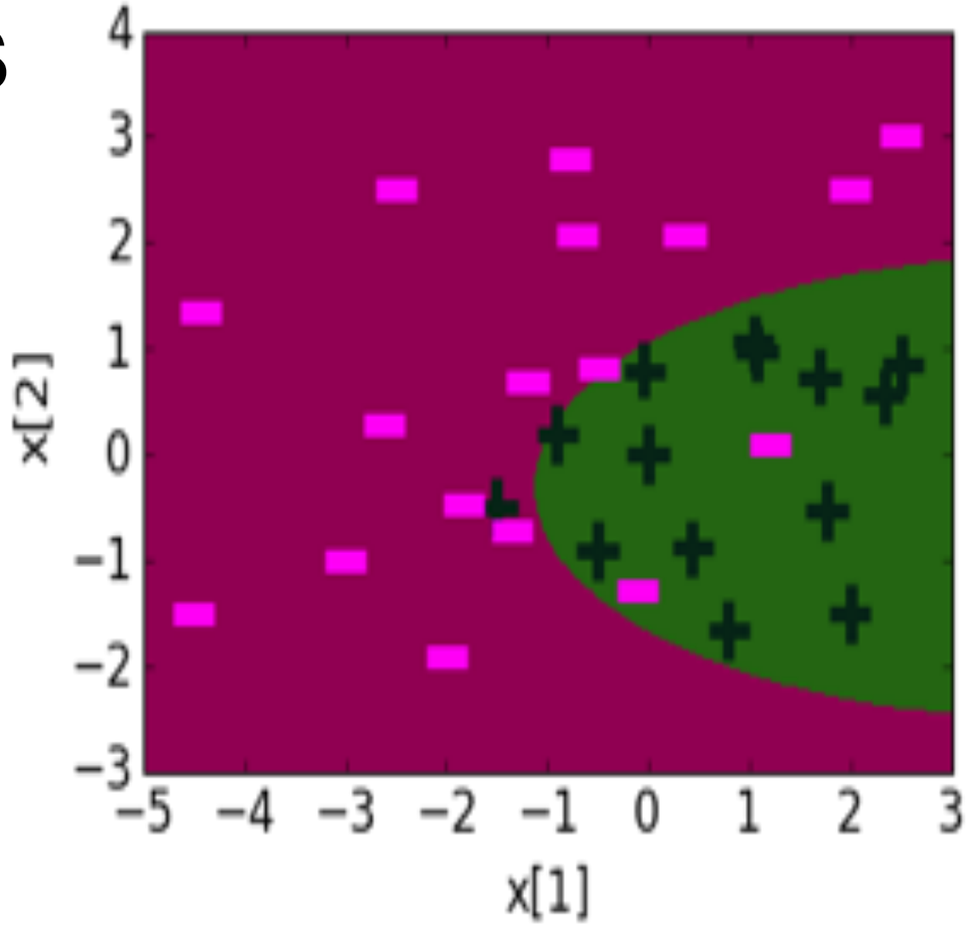


Feature	Value	Coefficient
$h_0(x)$	1	0.23
$h_1(x)$	x[1]	1.12
$h_2(x)$	x[2]	-1.07

- Simple **regression** models had **smooth** predictors
- Simple **classifier** models have **smooth** decision boundaries

Adding quadratic features

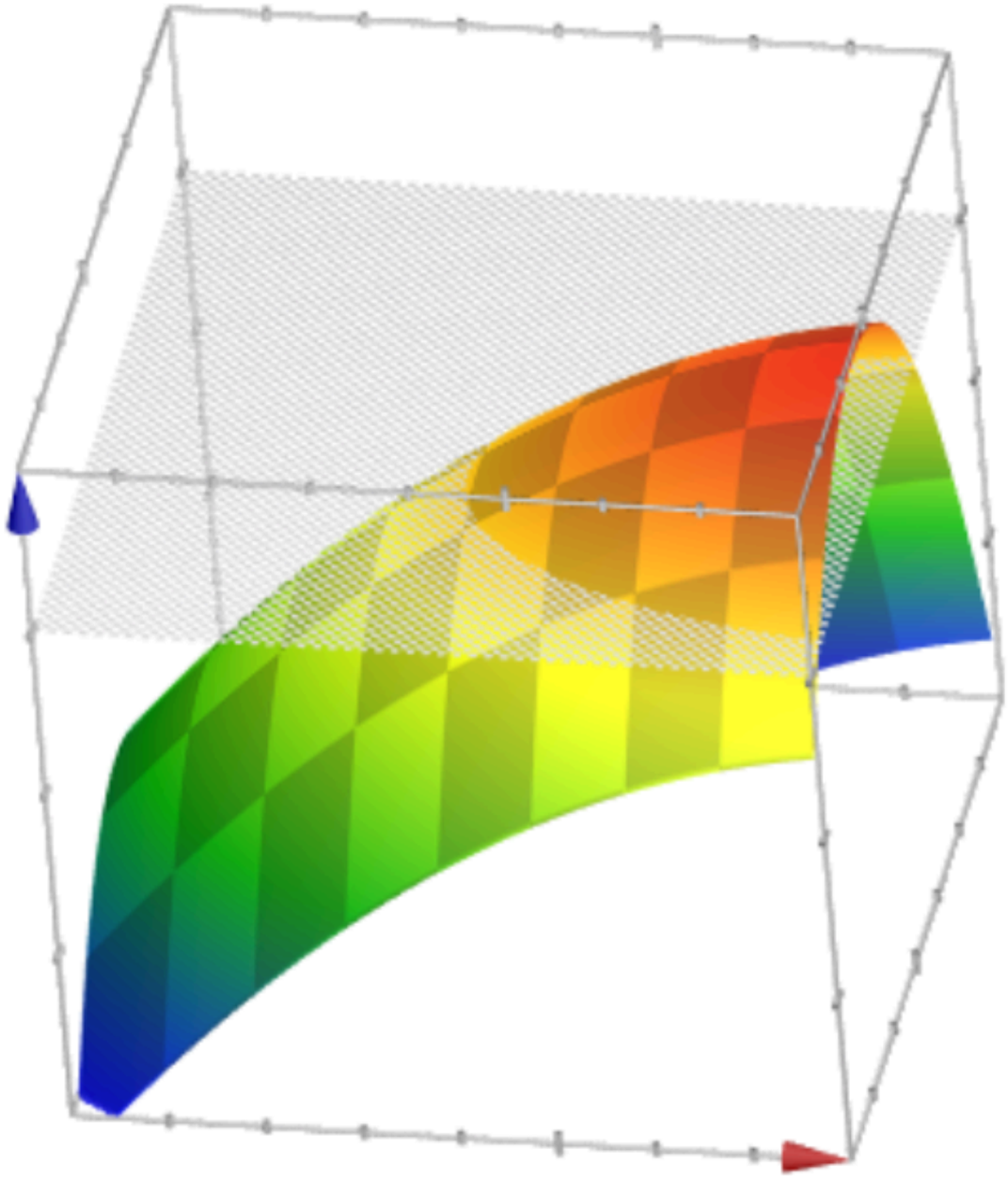


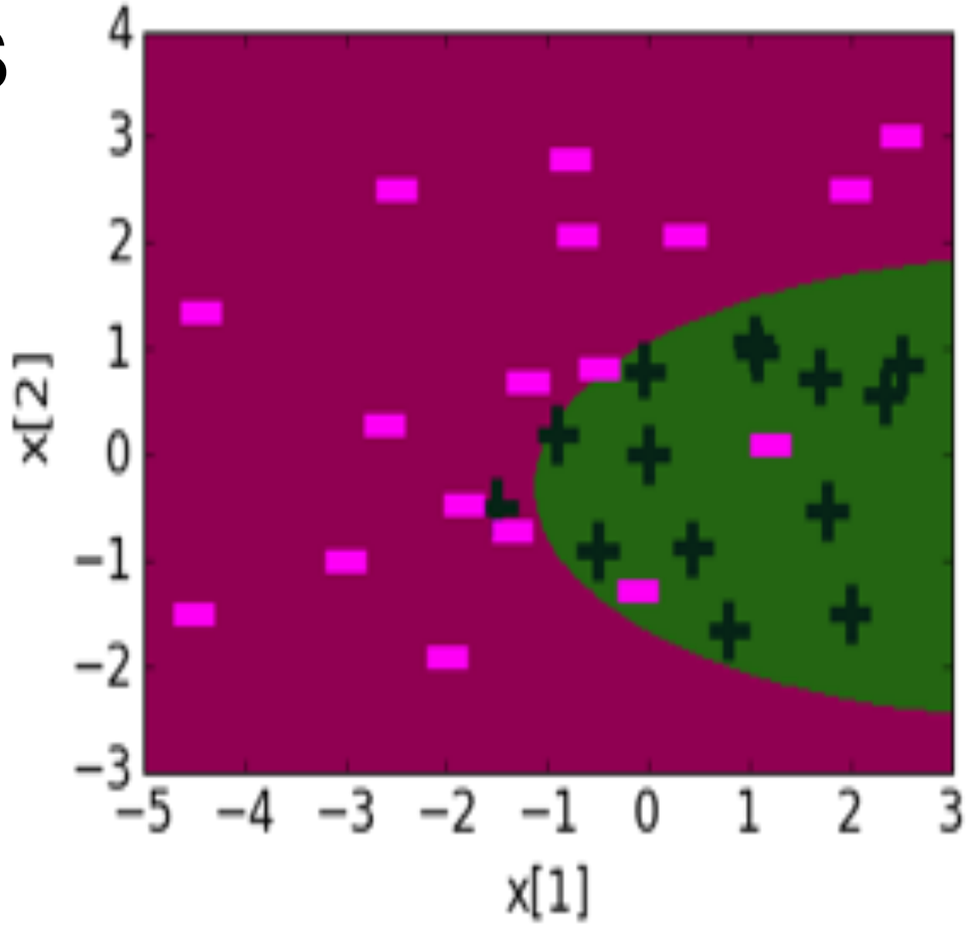


Feature	Value	Coefficient
$h_0(x)$	1	1.68
$h_1(x)$	x[1]	1.39
$h_2(x)$	x[2]	-0.59
$h_3(x)$	$(x[1])^2$	-0.17
$h_4(x)$	$(x[2])^2$	-0.96
$h_5(x)$	x[1]x[2]	Omitted

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

Adding quadratic features

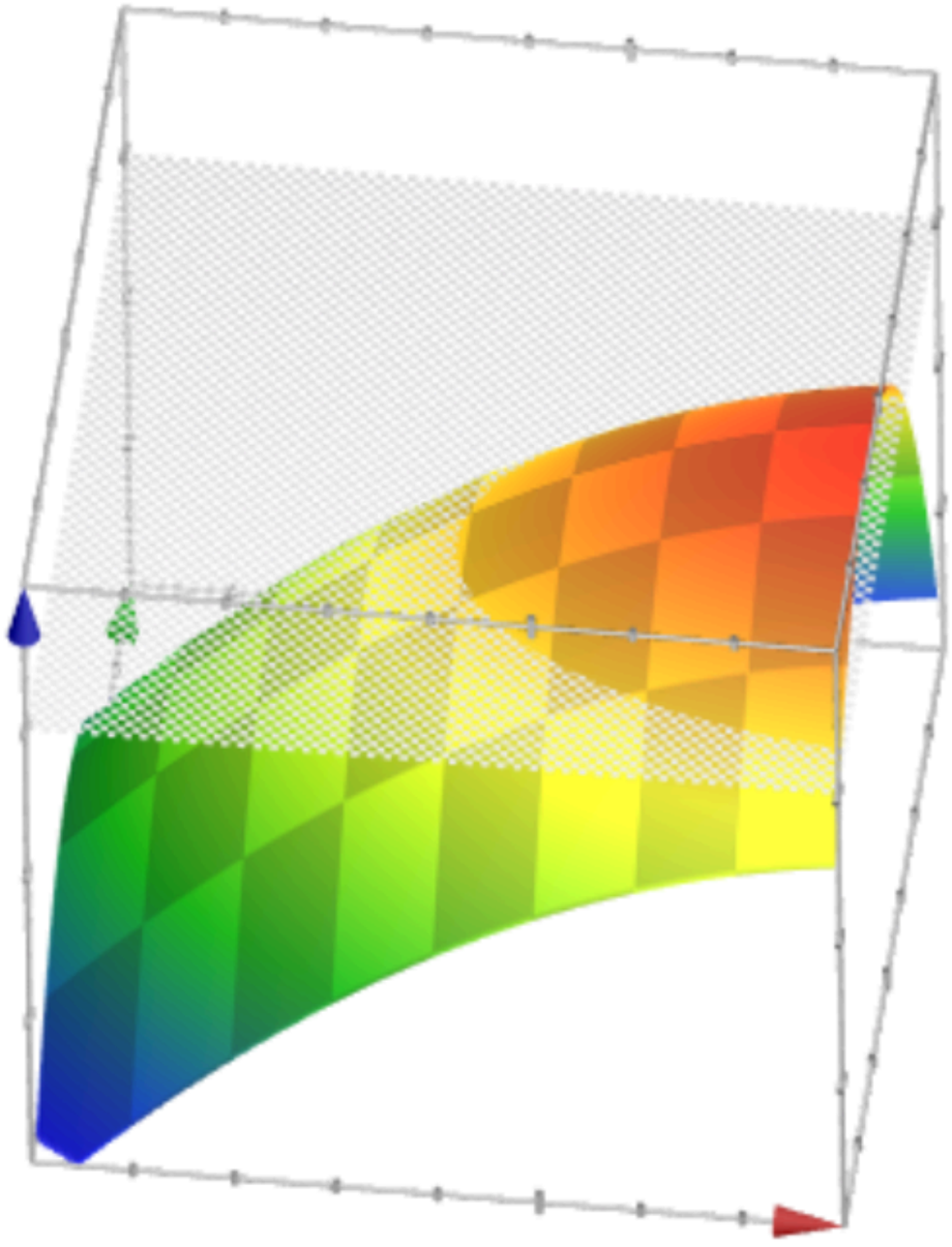


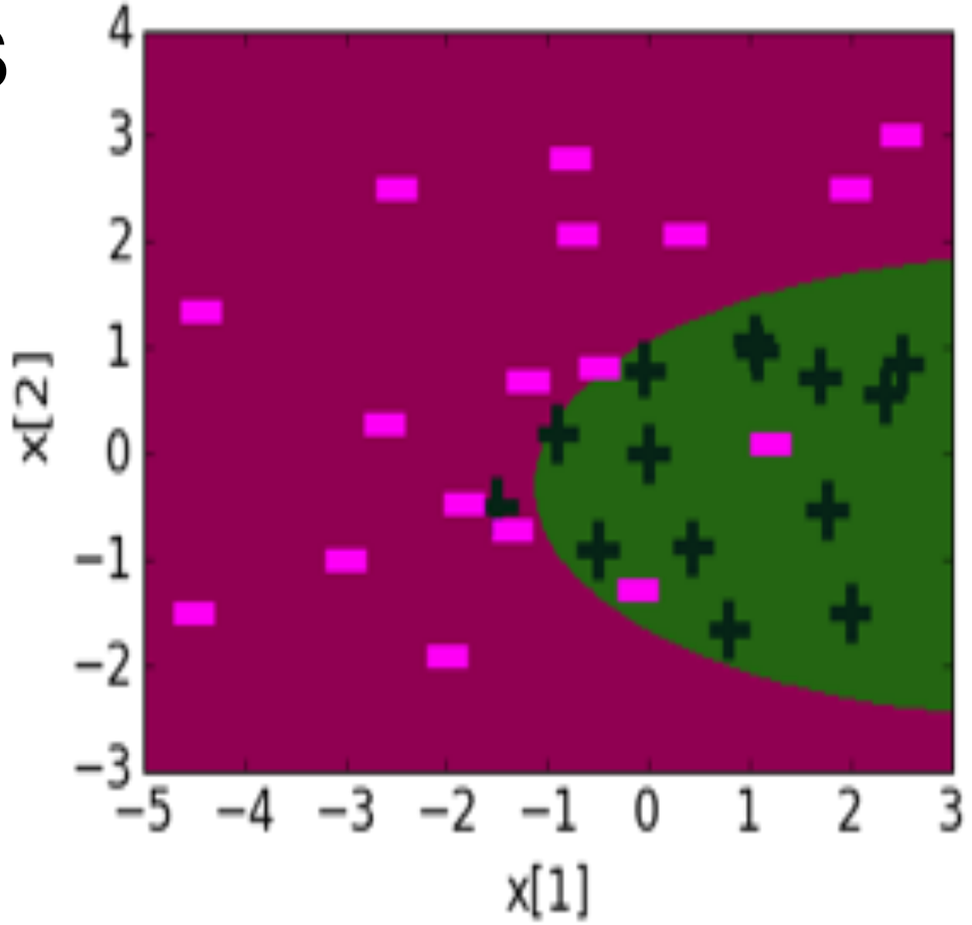


Feature	Value	Coefficient
$h_0(x)$	1	1.68
$h_1(x)$	x[1]	1.39
$h_2(x)$	x[2]	-0.59
$h_3(x)$	$(x[1])^2$	-0.17
$h_4(x)$	$(x[2])^2$	-0.96
$h_5(x)$	x[1]x[2]	Omitted

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

Adding quadratic features

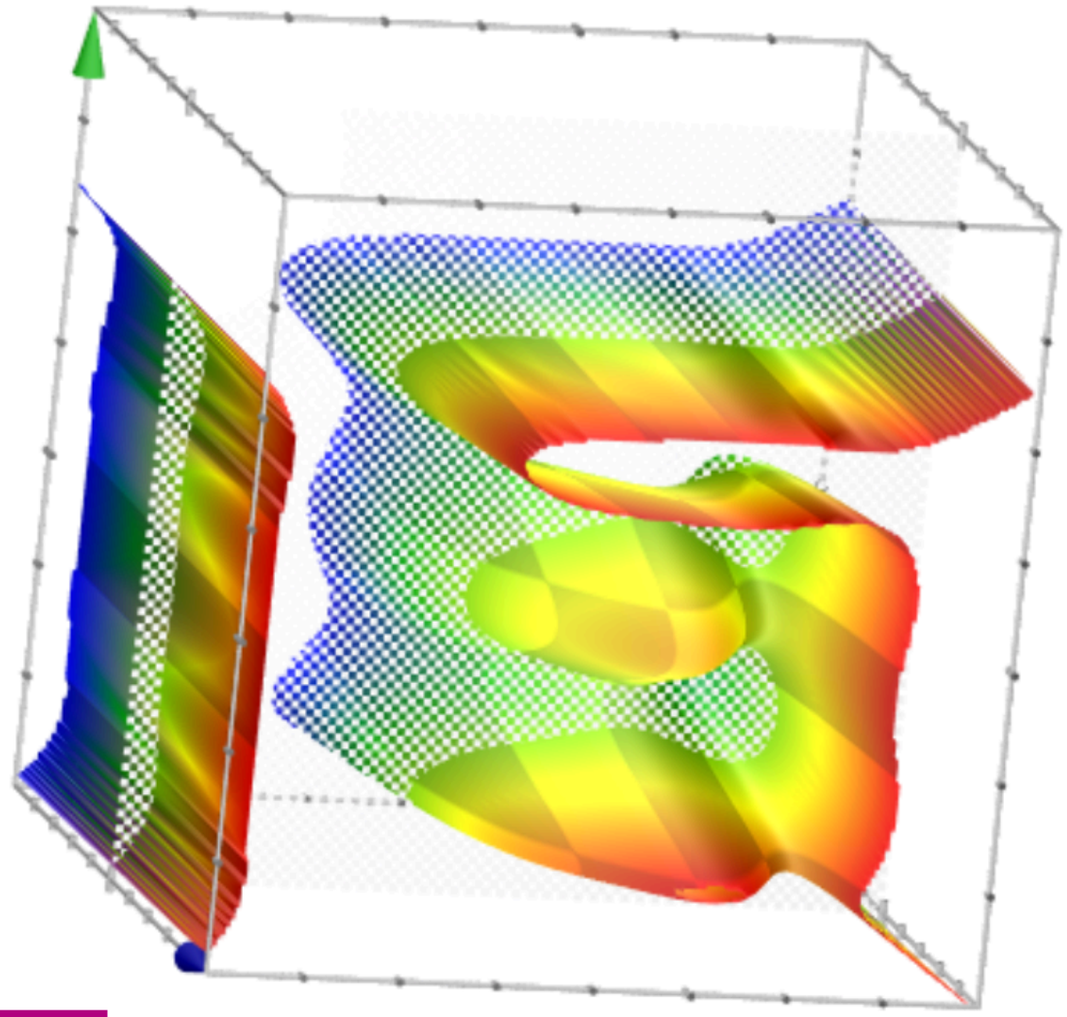




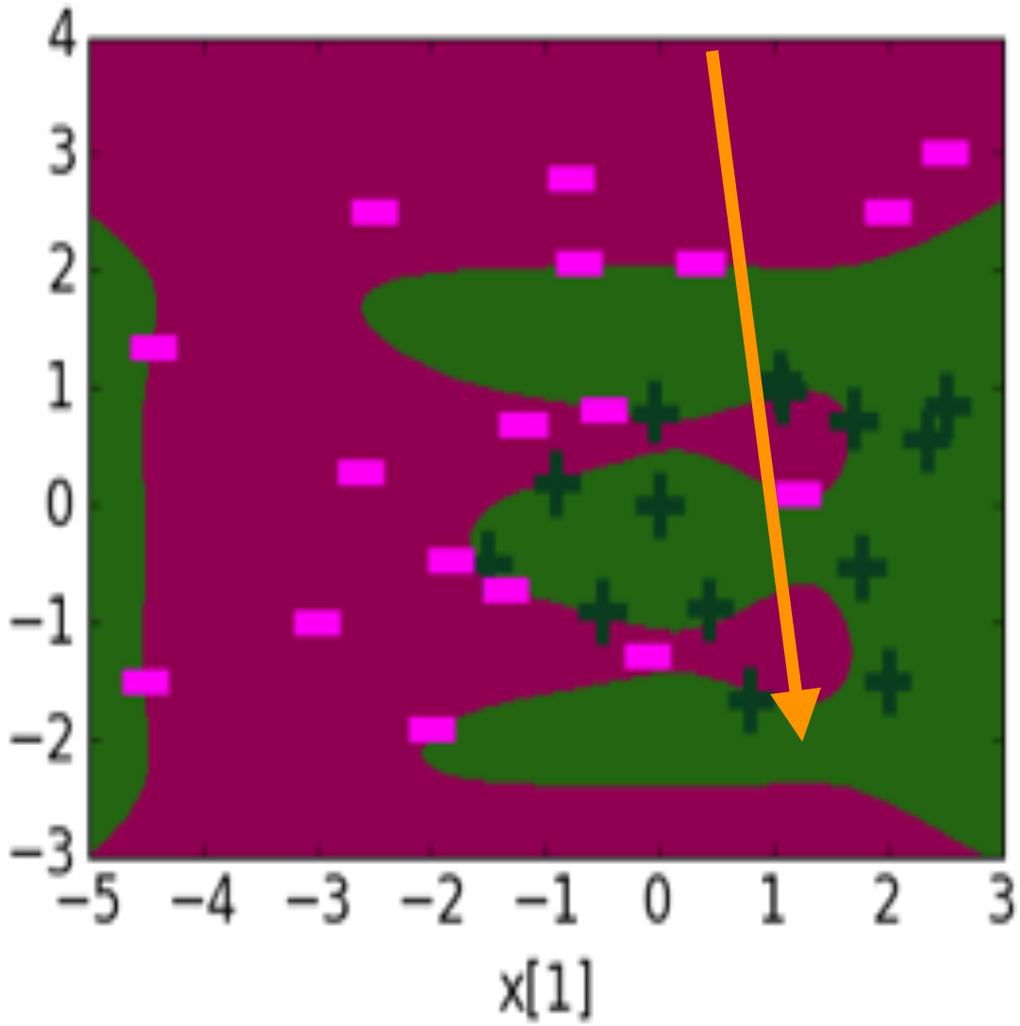
Feature	Value	Coefficient
$h_0(x)$	1	1.68
$h_1(x)$	x[1]	1.39
$h_2(x)$	x[2]	-0.59
$h_3(x)$	$(x[1])^2$	-0.17
$h_4(x)$	$(x[2])^2$	-0.96
$h_5(x)$	x[1]x[2]	Omitted

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

Adding higher degree polynomial features



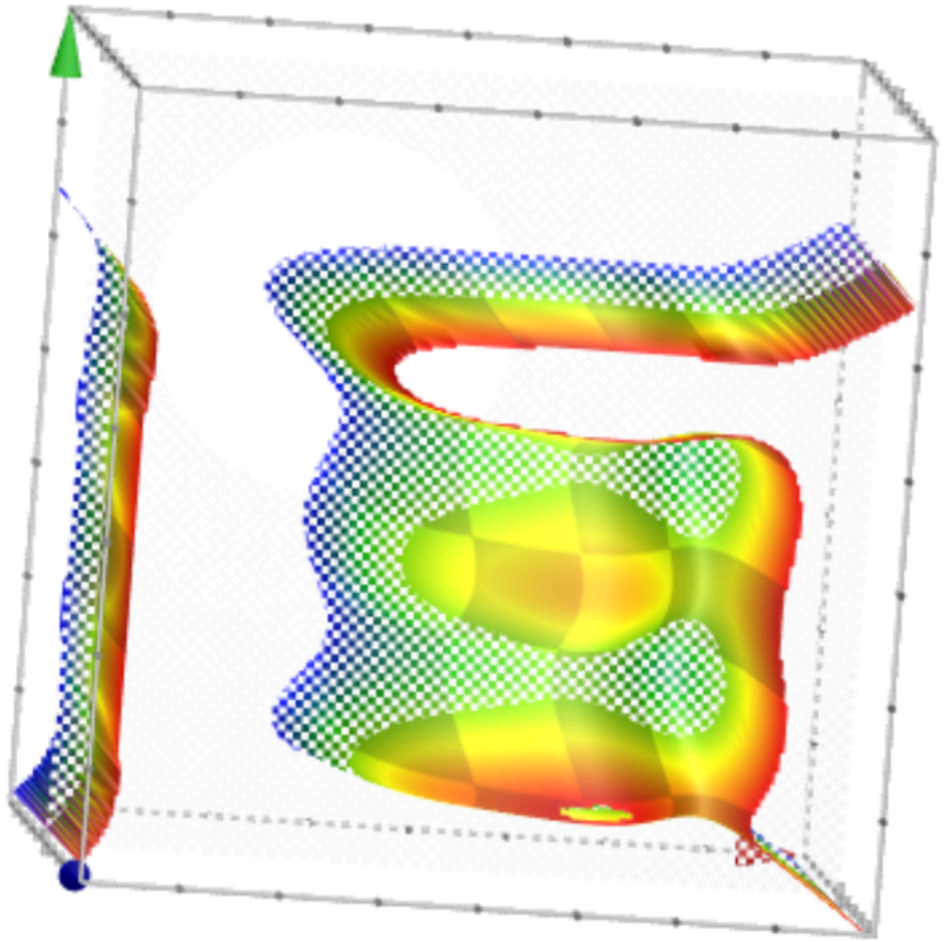
Overfitting leads to non-generalization



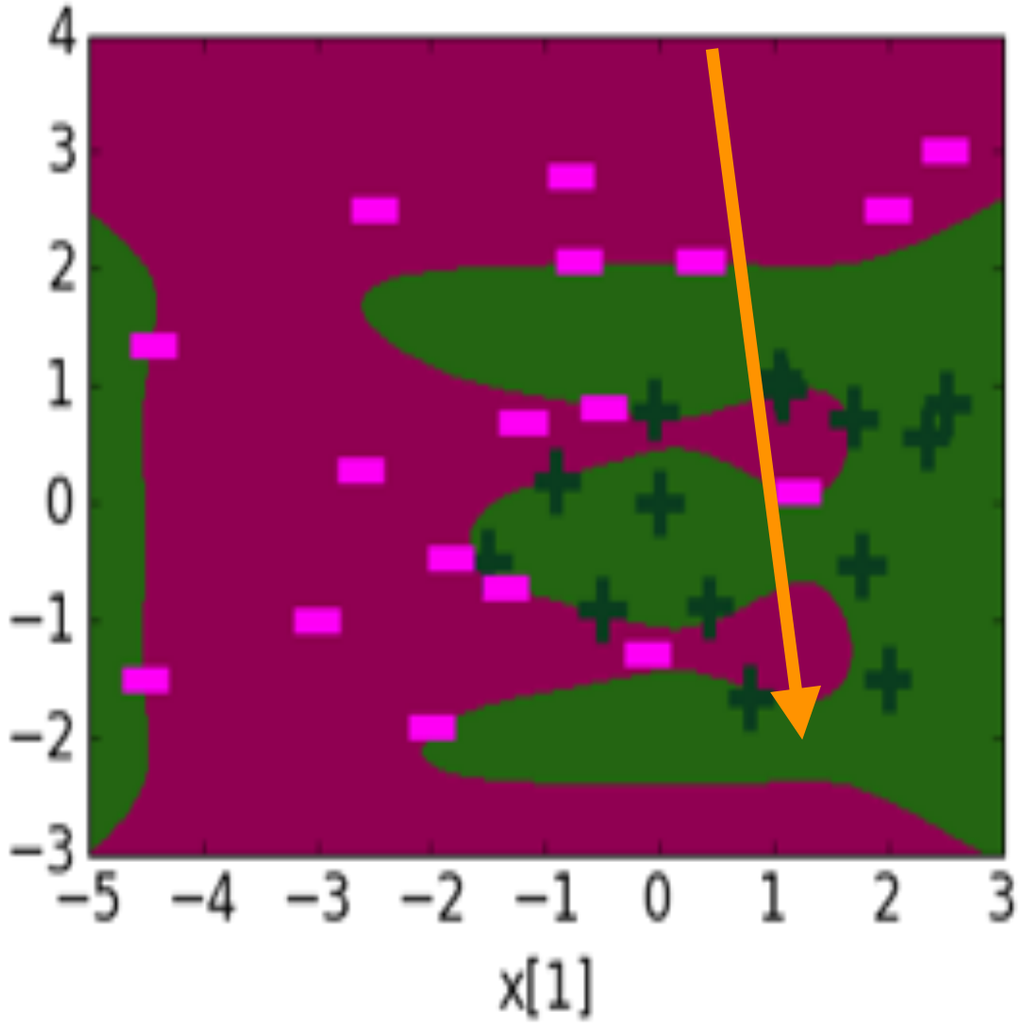
Feature	Value	Coefficient learned
$h_0(x)$	1	21.6
$h_1(x)$	x[1]	5.3
$h_2(x)$	x[2]	-42.7
$h_3(x)$	$(x[1])^2$	-15.9
$h_4(x)$	$(x[2])^2$	-48.6
$h_5(x)$	$(x[1])^3$	-11.0
$h_6(x)$	$(x[2])^3$	67.0
$h_7(x)$	$(x[1])^4$	1.5
$h_8(x)$	$(x[2])^4$	48.0
$h_9(x)$	$(x[1])^5$	4.4
$h_{10}(x)$	$(x[2])^5$	-14.2
$h_{11}(x)$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

Coefficient values getting large

Adding higher degree polynomial features



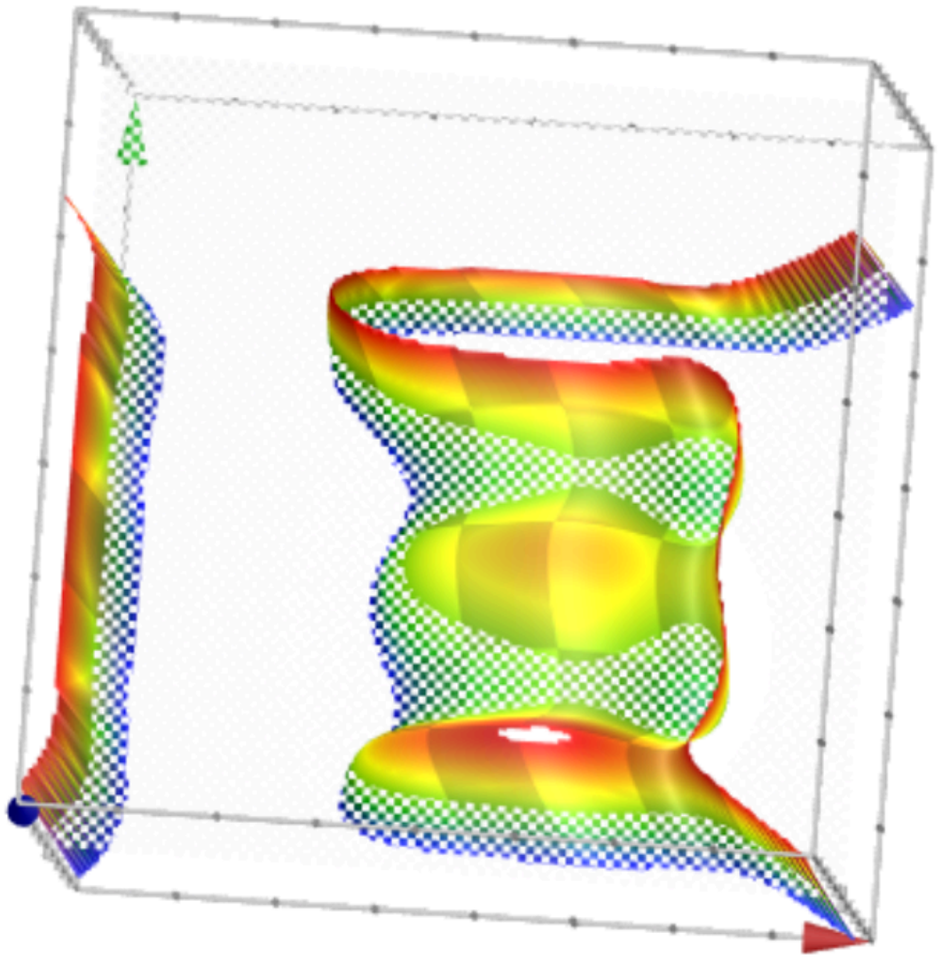
Overfitting leads to non-generalization



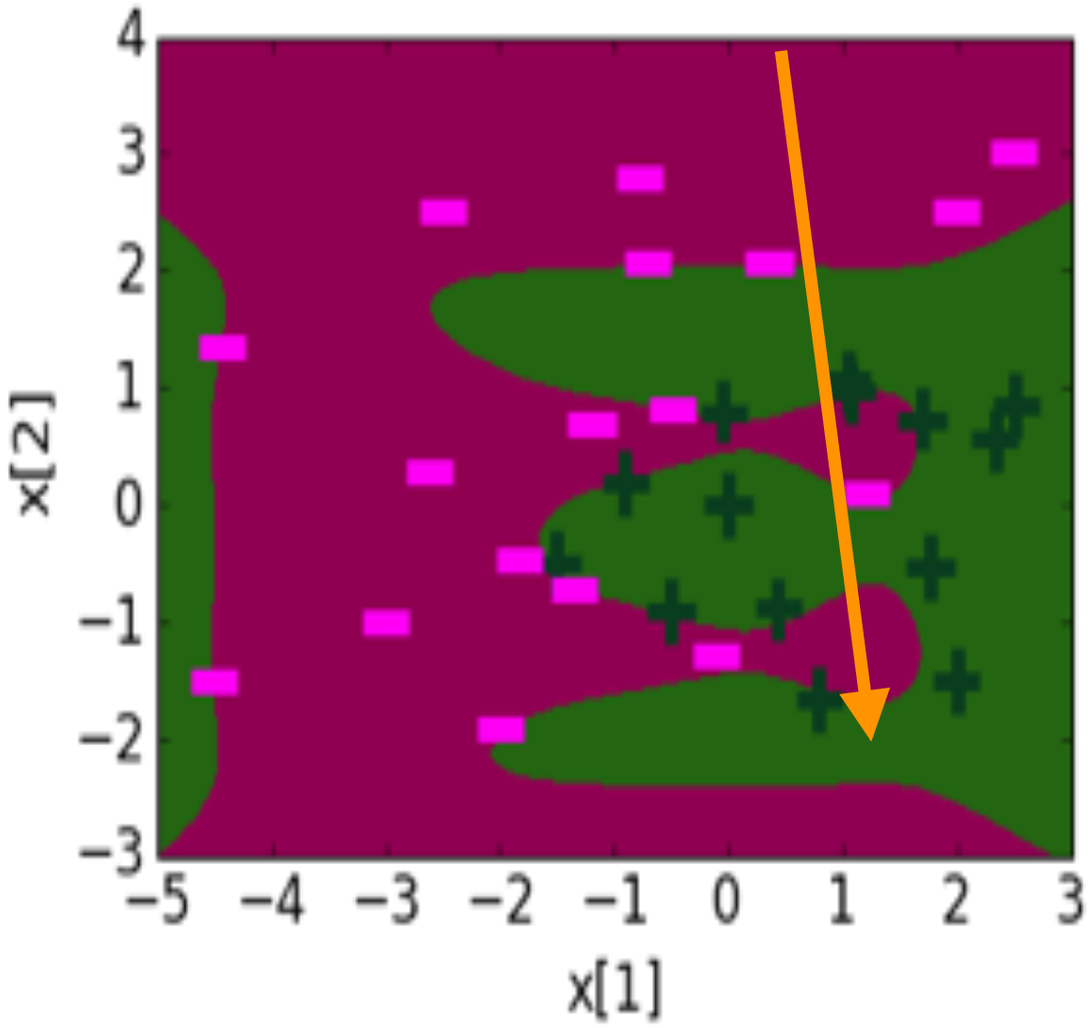
Feature	Value	Coefficient learned
$h_0(x)$	1	21.6
$h_1(x)$	x[1]	5.3
$h_2(x)$	x[2]	-42.7
$h_3(x)$	$(x[1])^2$	-15.9
$h_4(x)$	$(x[2])^2$	-48.6
$h_5(x)$	$(x[1])^3$	-11.0
$h_6(x)$	$(x[2])^3$	67.0
$h_7(x)$	$(x[1])^4$	1.5
$h_8(x)$	$(x[2])^4$	48.0
$h_9(x)$	$(x[1])^5$	4.4
$h_{10}(x)$	$(x[2])^5$	-14.2
$h_{11}(x)$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

Coefficient values getting large

Adding higher degree polynomial features



Overfitting leads to non-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

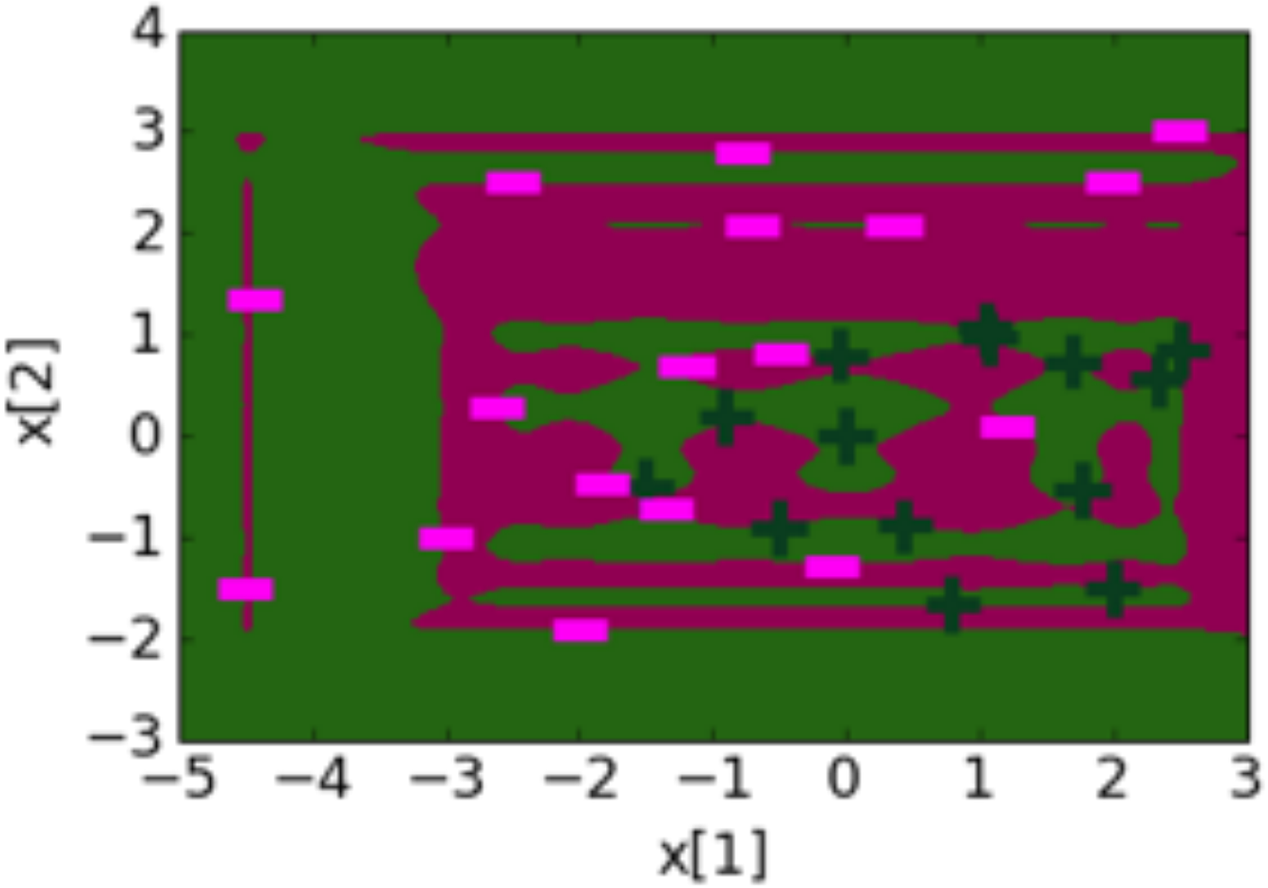
Coefficient values getting large

- Overfitting leads to very large values of

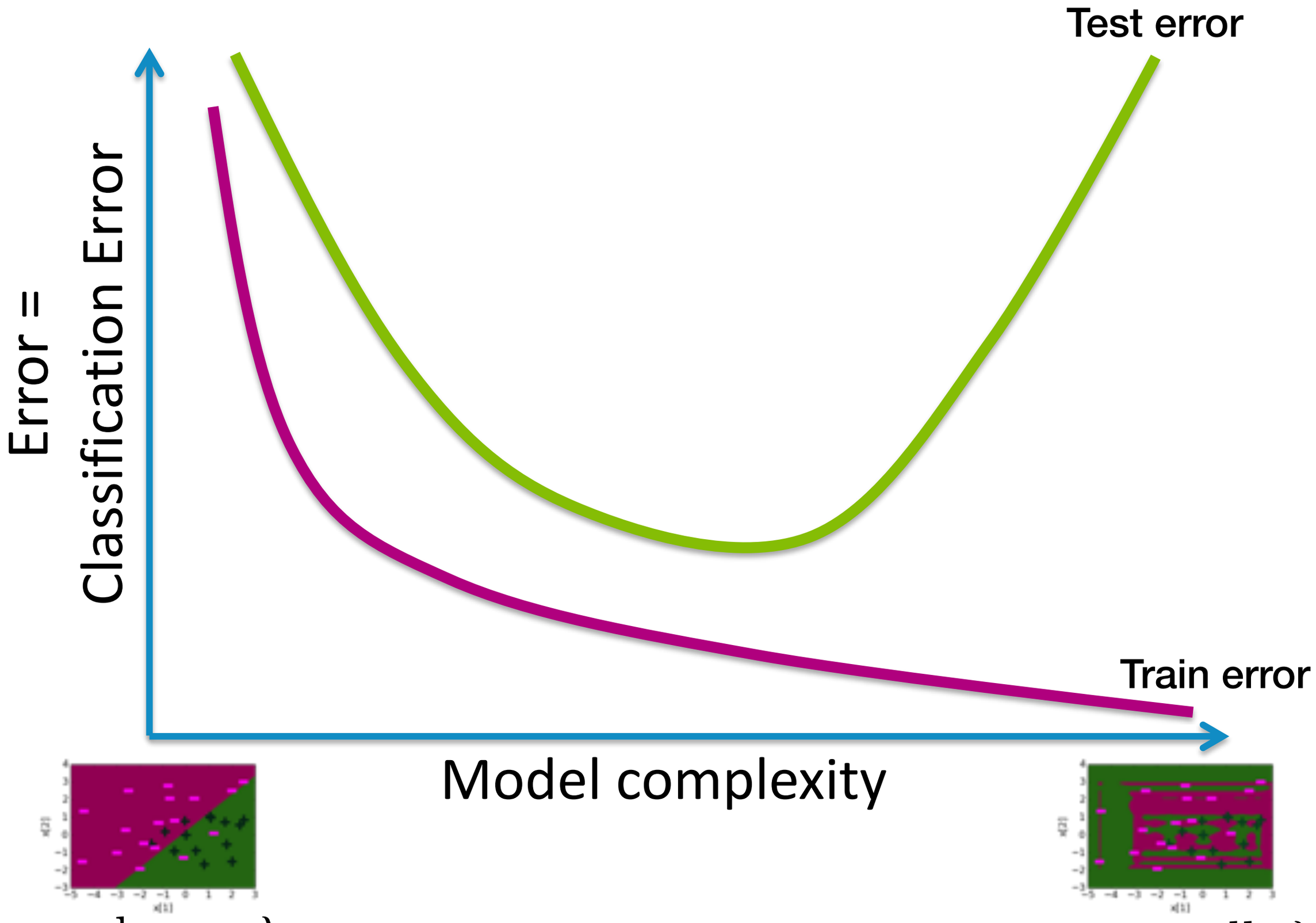
$$f(x) = w_0 h_0(x) + w_1 h_1(x) + w_2 h_2(x) + \cdots$$

Even higher degree polynomial features

Feature	Value	Coefficient
$h_0(x)$	1	8.7
$h_1(x)$	x[1]	5.1
$h_2(x)$	x[2]	78.7
…	…	…
$h_{11}(x)$	$(x[1])^6$	-7.5
$h_{12}(x)$	$(x[2])^6$	3803
$h_{13}(x)$	$(x[1])^7$	21.1
$h_{14}(x)$	$(x[2])^7$	-2406
…	…	…
$h_{37}(x)$	$(x[1])^{19}$	$-2*10^{-6}$
$h_{38}(x)$	$(x[2])^{19}$	-0.15
$h_{39}(x)$	$(x[1])^{20}$	$-2*10^{-8}$
$h_{40}(x)$	$(x[2])^{20}$	0.03



Overfitting in classification



Over-confident predictions

Logistic regression

- Recall the probabilistic interpretation of the trained linear model $w^T x$ under logistic regression

$$\underbrace{\left(\frac{1}{1+e^{-w^T x}}\right.}_{\mathbb{P}(y_i=+1|x_i)},\ \underbrace{\left.\frac{1}{1+e^{w^T x}}\right)}_{\mathbb{P}(y_i=-1|x_i)}$$

- Overfitting leads to large ***w***'s
- Resulting $w^T x$ changes rapidly having very positive/negative values
- class probability (for one class) approaches 1 and class probability of the other class approaches 0
- Such model is called **over-confident** in its predictions

Effect of size of the weight on confidence

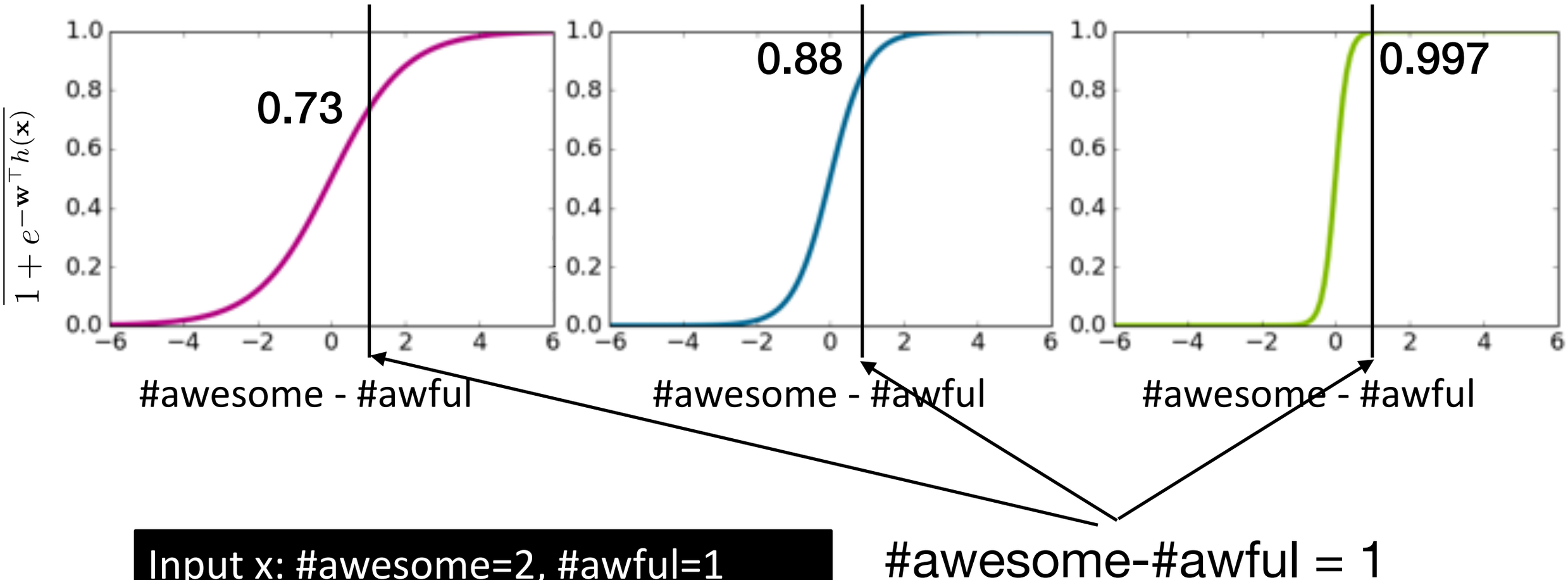
Small weight

w_0	0
$w_{\#awesome}$	+1
$w_{\#awful}$	-1

w_0	0
$w_{\#awesome}$	+2
$w_{\#awful}$	-2

Large weight

w_0	0
$w_{\#awesome}$	+6
$w_{\#awful}$	-6



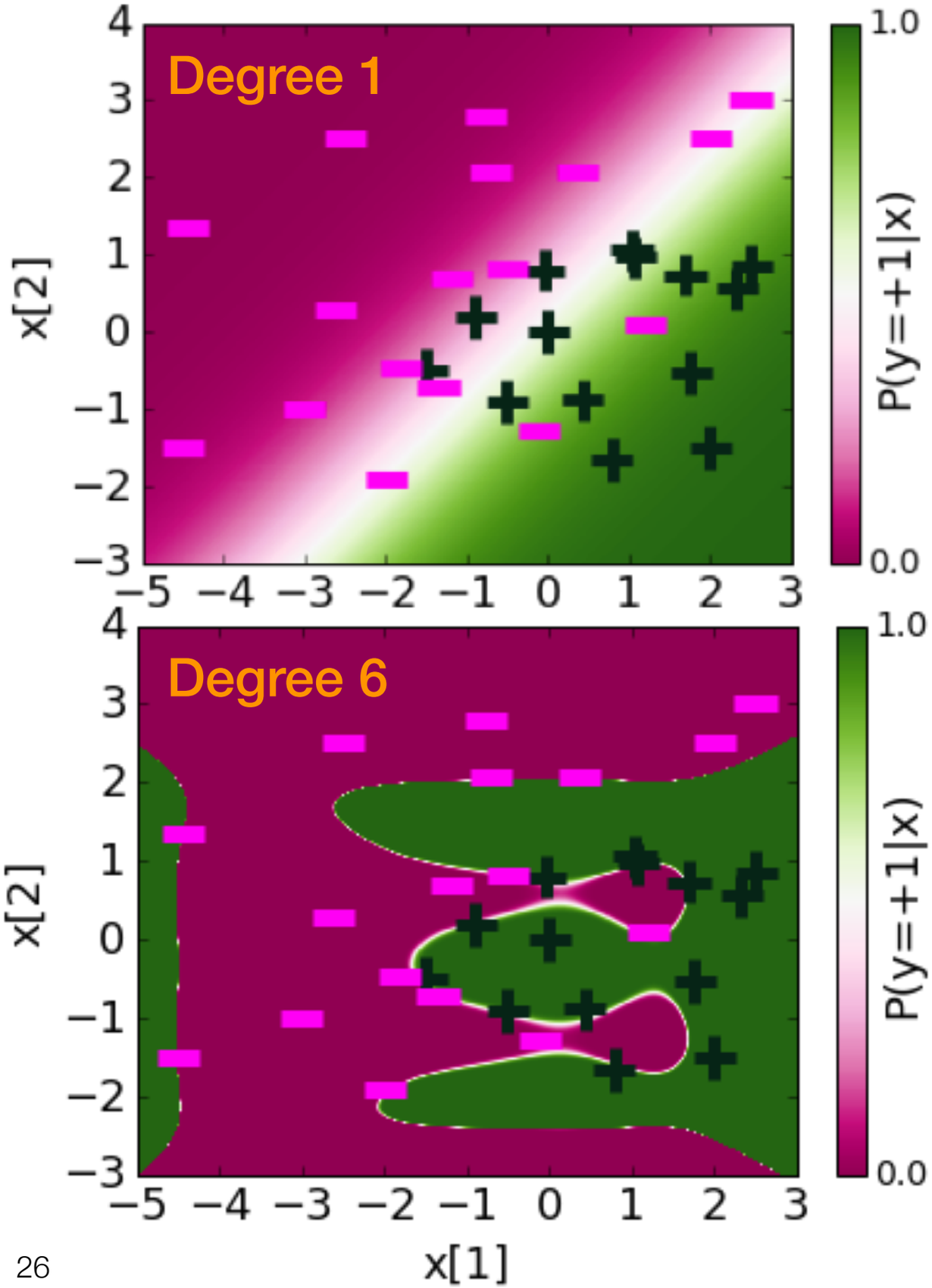
Input x: #awesome=2, #awful=1

#awesome-#awful = 1

- Over-fitted model gives higher confidence for same data point

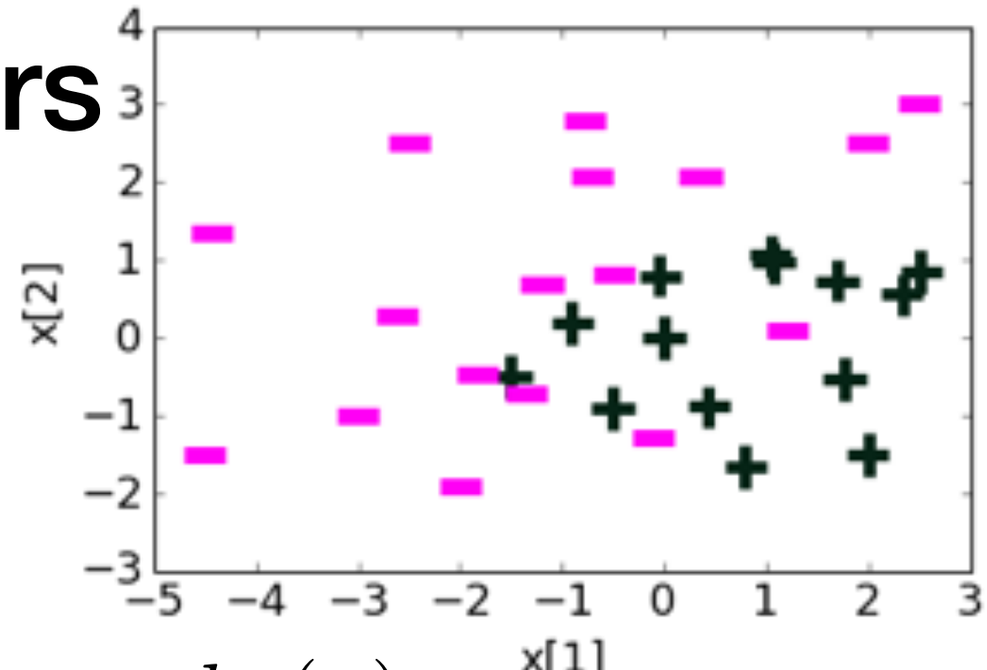
Plotting the prediction

$$P(y = +1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^\top h(\mathbf{x})}}$$



Regularization

Optimizing with regularizers



- data: ***x*** in 2-dimensions, ***y*** in {+1,-1}
- features: polynomials
- model: linear

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

- quality metric: logistic regression

$$\mathcal{L}(w) = \frac{1}{N} \sum_{i=1}^{N} \ell(\underbrace{f(x_i)}_{\hat{y}_i}, y_i)$$

$$\ell(\hat{y}_i, y_i) = \log(1 + e^{-y_i \hat{y}_i})$$

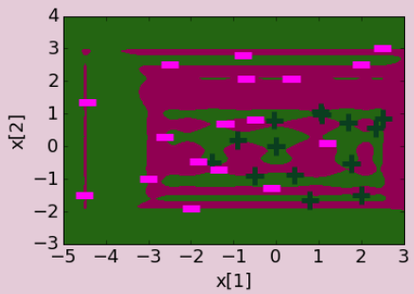
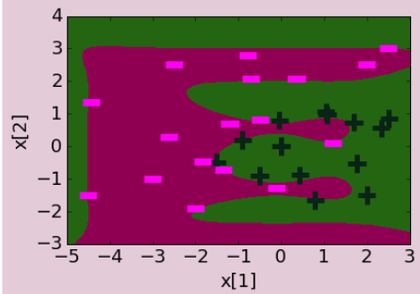
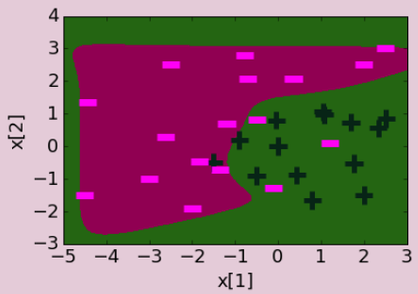
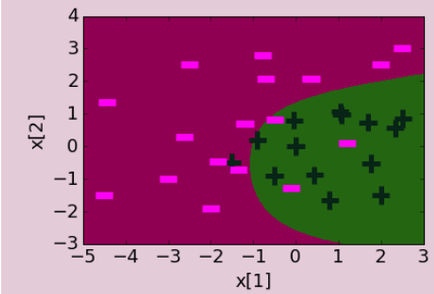
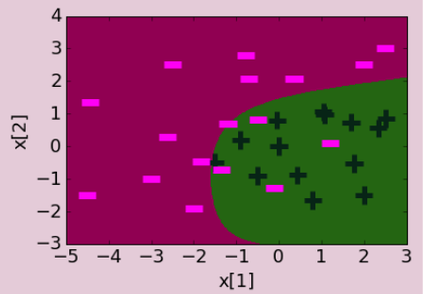
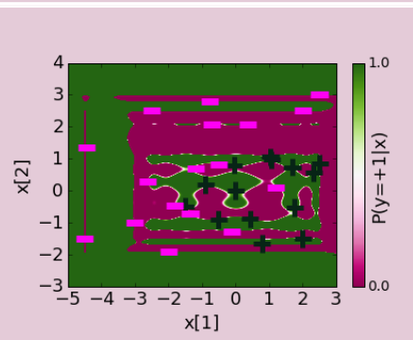
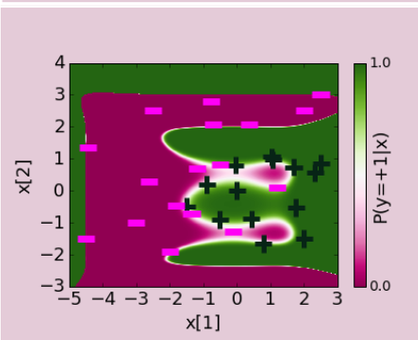
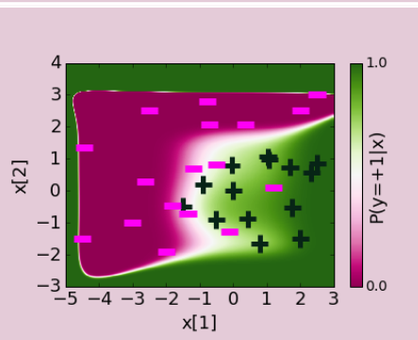
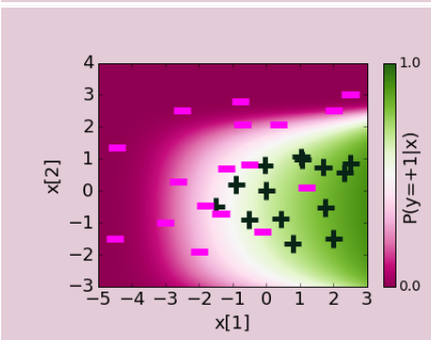
- regularizer: regularizer

$$\text{minimize}_w \; \mathcal{L}(w) + \lambda \underbrace{r(w)}_{\|w\|^2 \text{ or } \|w\|_1}$$

square regularizer: $\|w\|^2 = w_1^2 + \cdots + w_d^2$

absolute regularizer: $\|w\|_1 = |w_1| + \cdots + |w_d|$

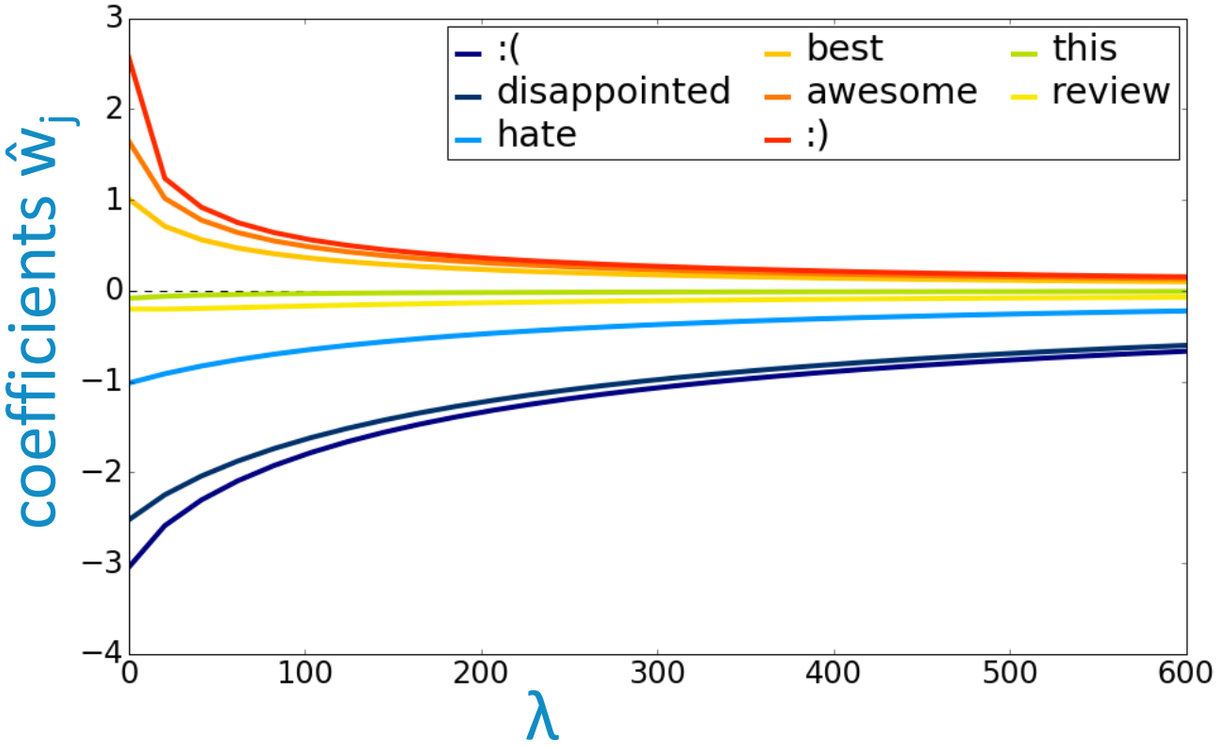
Example: degree-20 polynomial features

Regularization	λ = 0	λ = 0.00001	λ = 0.001	λ = 1	λ = 10
Range of coefficients	-3170 to 3803	-8.04 to 12.14	-0.70 to 1.25	-0.13 to 0.57	-0.05 to 0.22
Decision boundary					
Learned probabilities					

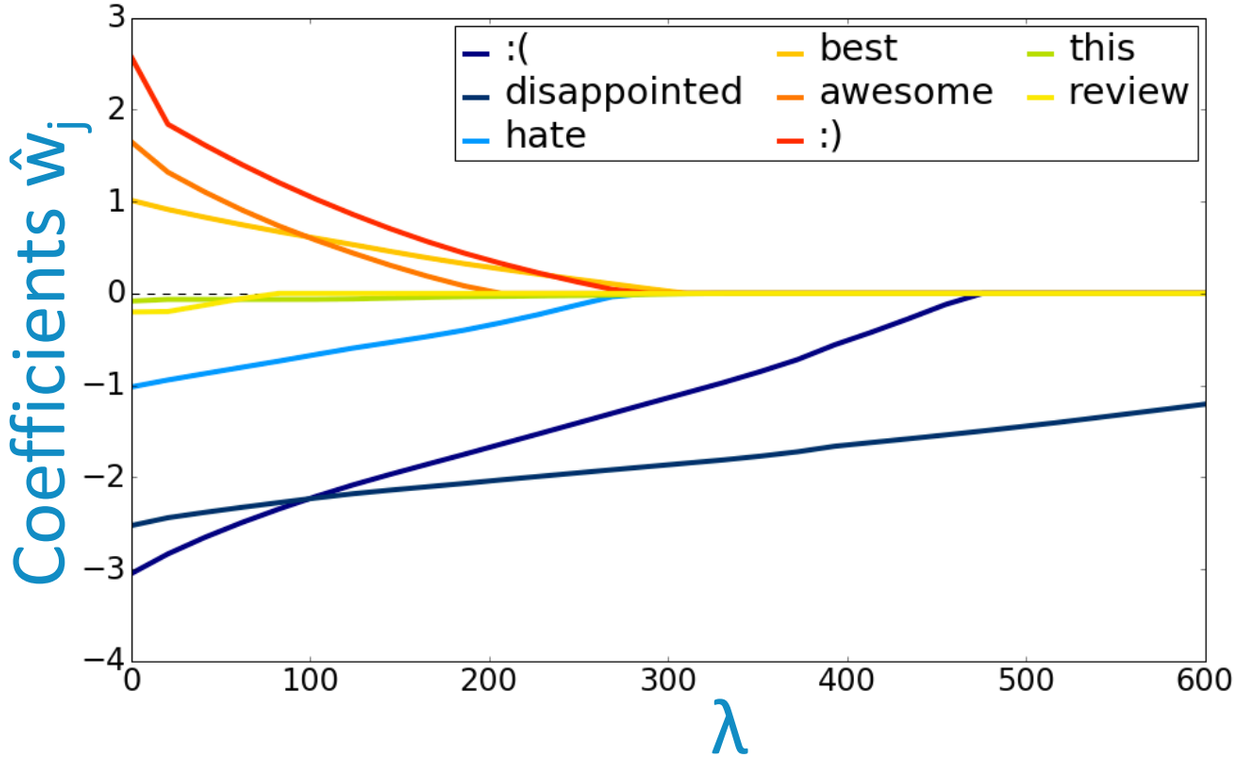
- Trade-off between regularization and model complexity
  - Adding regularizer with appropriate lambda avoids over-fitting

Regularization path

square regularizer: $\|w\|^2 = w_1^2 + \cdots + w_d^2$



absolute regularizer: $\|w\|_1 = |w_1| + \cdots + |w_d|$



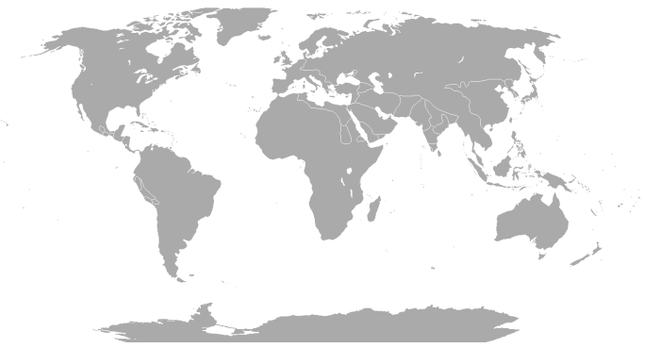
- Absolute regularizer (a.k.a L1 regularizer) gives sparse parameters, which is desired for interpretability, feature selection, and efficiency

Multi-class classification

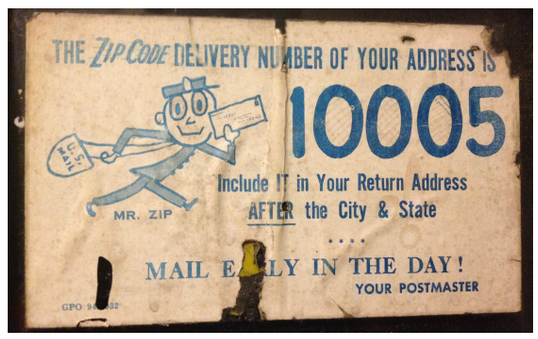
How do we encode categorical data *y*?

- So far, we considered Boolean case where there are two categories
- Encoding ***y*** is simple: {+1,-1}, as there is not much difference

- Multi-class classification predicts categorial ***y***
- Taking values in $\boldsymbol{V=\{v_1,v_2,\ldots,v_K\}}$
- $\boldsymbol{V_i}$'s are called **classes** or **labels**
- examples:



Country of birth
(Argentina, Brazil, USA,...)



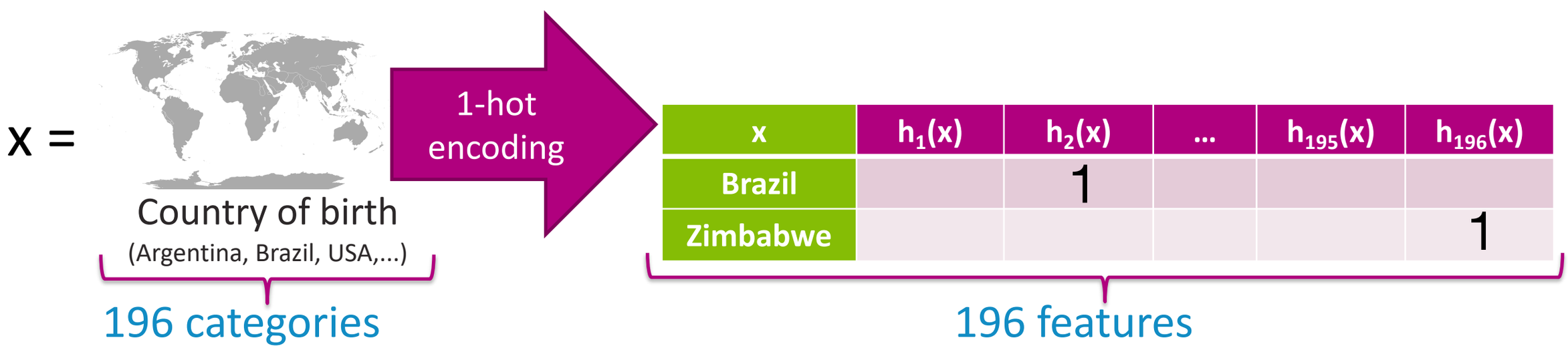
Zipcode
(10005, 98195,...)

All English words

- A **K-class classifier** predicts ***y*** given ***x***

Embedding v_i in real values

- For optimization we need to **embed** raw categorial v_i's into real valued vectors
- There are many ways to embed categorial data
  - Trua->1, False->-1
  - Yes->1, Maybe->0, No->-1
  - Yes->(1,0), Maybe->(0,0), No->(0,1)
  - Apple->(1,0,0), Orange->(0,1,0), Banana->(0,0,1)
  - Ordering: (Horse 3, Horse 1, Horse 2) -> (3,1,2)
- We use **one-hot embedding** (a.k.a. **one-hot encoding**)
  - Each class is a standard basis vector in K dimension



Multi-class logistic regression

- data: categorical ***y*** in ***{v₁,v₂,…,v_K}*** with ***K*** categories
- model: linear

$$f(x_i) = \begin{bmatrix} f_1(x_i) \\ f_2(x_i) \\ \vdots \\ f_{K-1}(x_i) \end{bmatrix} = \begin{bmatrix} w_{1,0} + w_{1,1}h_1(x) + w_{1,2}h_2(x) + \cdots \\ w_{2,0} + w_{2,1}h_1(x) + w_{2,2}h_2(x) + \cdots \\ \vdots \\ w_{K-1,0} + w_{K-1,1}h_1(x) + w_{K-1,2}h_2(x) + \cdots \end{bmatrix} = W^T\, h(x)$$

- quality metric: logistic regression

2 classes

$$\mathbb{P}(\hat{y}_i = -1) = \frac{e^{f(x_i)}}{1 + e^{f(x_i)}}$$

$$\mathbb{P}(\hat{y}_i = +1) = \frac{1}{1 + e^{f(x_i)}}$$

K classes

$$\mathbb{P}(\hat{y}_i = v_1) = \frac{e^{f_1(x_i)}}{1 + e^{f_1(x_i)} + \cdots + e^{f_{K-1}(x_i)}}$$

$$\mathbb{P}(\hat{y}_i = v_2) = \frac{e^{f_2(x_i)}}{1 + e^{f_1(x_i)} + \cdots + e^{f_{K-1}(x_i)}}$$

$$\vdots$$

$$\mathbb{P}(\hat{y}_i = v_{K-1}) = \frac{e^{f_{K-1}(x_i)}}{1 + e^{f_1(x_i)} + \cdots + e^{f_{K-1}(x_i)}}$$

$$\mathbb{P}(\hat{y}_i = v_K) = \frac{1}{1 + e^{f_1(x_i)} + \cdots + e^{f_{K-1}(x_i)}}$$

Maximum Likelihood Estimator

$$\text{maximize}_w \quad \frac{1}{N} \sum_{i=1}^{N} \log \Big(\mathbb{P}(\hat{y}_i = y_i) \Big)$$

$$\text{minimize}_w \quad \frac{1}{N} \sum_{i=1}^{N} \underbrace{\log \Big(1 + e^{-y_i f(x_i)} \Big)}_{\text{logistic loss}}$$

$$\text{minimize}_w \quad \frac{1}{N} \sum_{i=1}^{N} \underbrace{\log \Big((1 + e^{f_1(x_i)} + \cdots + e^{f_{K-1}(x_i)}) e^{-f_{y_i}(x_i)} \Big)}_{\text{logistic loss}}$$