

Logistics:

- Mid-term evaluation
- As we transition to in-person lectures and sections starting 1/31/2022, some OHs will be in-person and some will be on zoom.

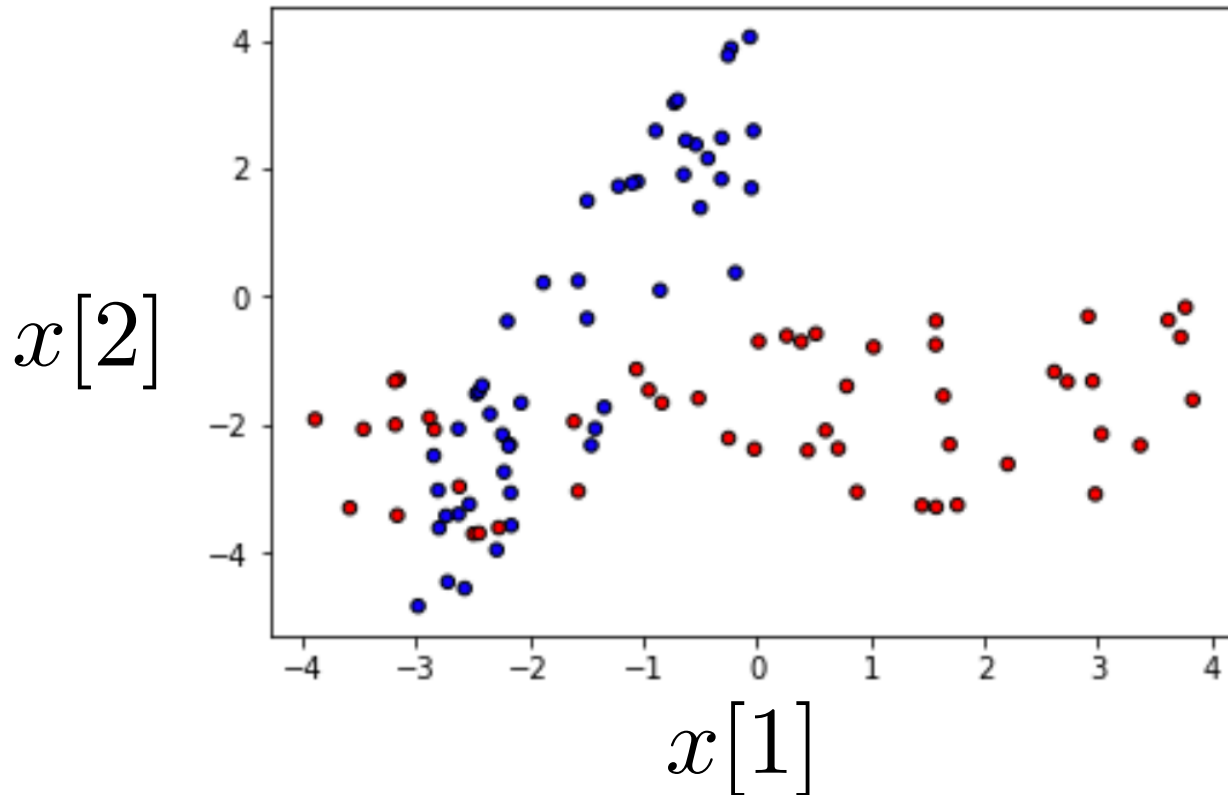
Lecture 11:

Classification with logistic regression

- Regression: label is continuous valued
- Classification: label is discrete valued, e.g., $\{0,1\}$
- Note that logistic regression is a classification algorithm not a regression algorithm



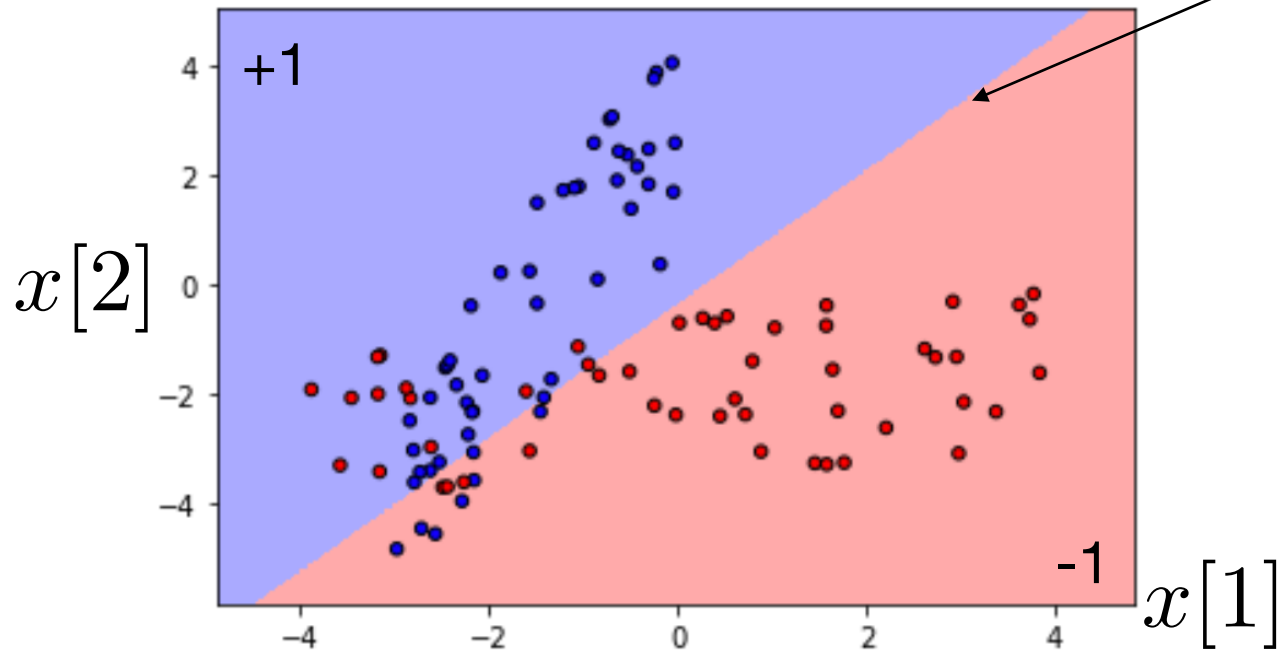
Training data for a binary classification problem



- in this example, each input is $x_i \in \mathbb{R}^2$
- Red points have label $y_i=-1$, blue points have label $y_i=1$
- We want a predictor that maps any $x \in \mathbb{R}^2$ to a prediction $\hat{y} \in \{-1, +1\}$

Example: linear classifier trained on 100 samples

simple decision boundary at $w^T x + b = 0$

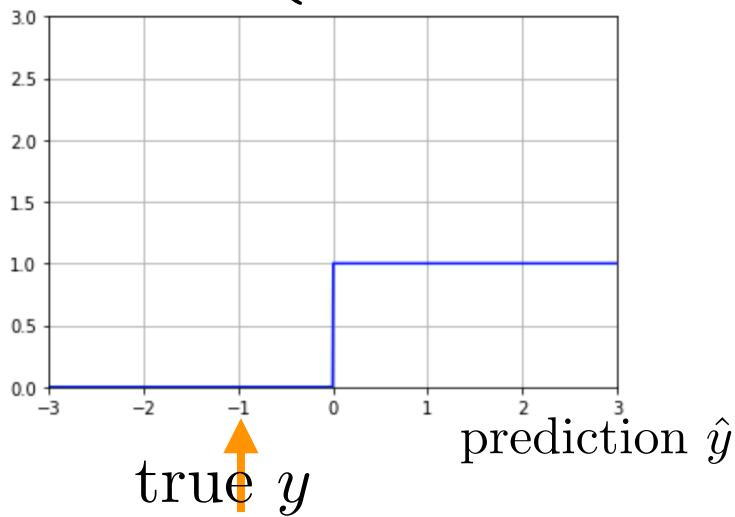


- We fit a linear model: $w_0 + w_1x[1] + w_2x[2] = 0.8 - 1.1x[1] + 0.9x[2]$
- predict using $\hat{y} = \text{sign}(0.8 - 1.1x[1] + 0.9x[2])$
- decision boundary is the line (or hyperplane in higher dimensions) defined by $0.8 - 1.1x[1] + 0.9x[2] = 0$
- note that a model $2w^T x + 2b$ has the same predictions as $w^T x + b$
- How do we find such a good linear classifier that fits the data?

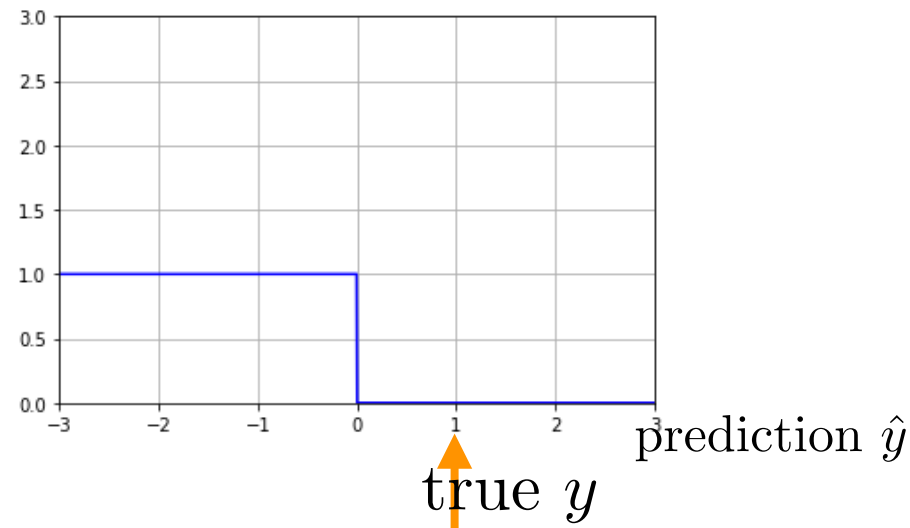
Binary Classification with 0-1 loss

- **Learn** a linear model: $f : x \mapsto \hat{y} = b + x^T w$
 - x – input/features, $y \in \{-1, +1\}$ – label in target classes
 - Prediction: $\text{sign}(\hat{y})$
- **Ideal loss function** $\ell(\hat{y}, y)$:
 - **0-1 loss**, because we care about how many were classified correctly
 - What are weaknesses? **Not differentiable and zero derivative**

$$\ell(\hat{y}, -1) = \begin{cases} 0 & \hat{y} < 0 \\ +1 & \hat{y} \geq 0 \end{cases}$$



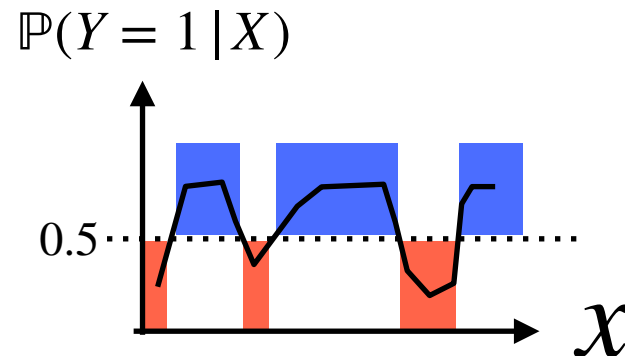
$$\ell(\hat{y}, +1) = \begin{cases} 0 & \hat{y} > 0 \\ +1 & \hat{y} \leq 0 \end{cases}$$



Binary Classification with 0-1 loss

- If we know the underlying distribution, $(x, y) \sim P_{X,Y}$ and if we do not restrict ourselves to **any function class**, then we could find the optimal predictor under **0-1 loss**, called **Bayes optimal classifier**

- $$f_{\text{Bayes}}(x) = \arg \max_{\hat{y} \in \{-1, 1\}} \mathbb{P}_{Y|X}(Y = \hat{y} | X = x)$$



- Claim: Bayes optimal classifier achieves the minimum possible achievable **true error for 0-1 loss**
- True error: $\mathbb{E}_{X,Y}[\ell(f(X), Y)] = \mathbb{P}(\text{sign}(f(X)) \neq Y)$
- Proof:

We can write the true error of a classifier $f(\cdot)$ using chain rule as

$$\mathbb{E}_{X,Y}[\mathbb{I}\{Y \neq f(X)\}] = \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathbb{I}\{Y \neq f(x)\}] | X = x] = \mathbb{E}_X[\mathbb{P}_{Y|X}(Y \neq f(x) | X = x)]$$

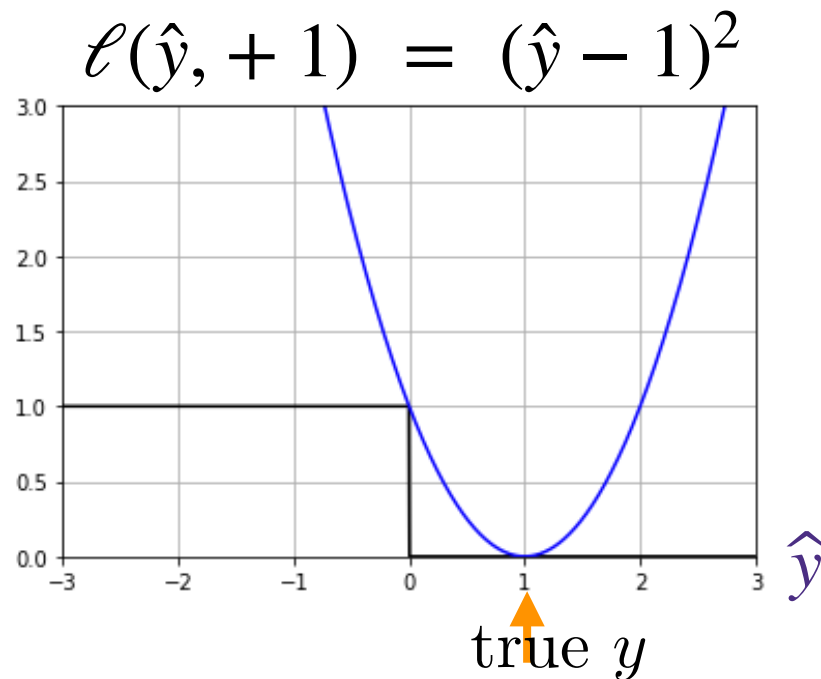
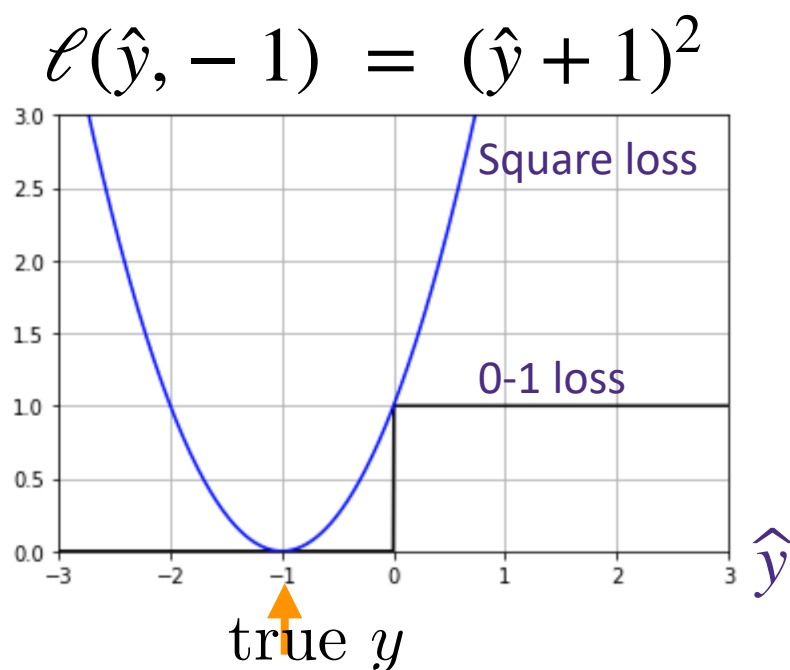
optimal classifier minimizes this true error, at every x

$$f_{\text{opt}}(x) = \arg \min_{\hat{y} \in \{-1, 1\}} \mathbb{P}_{Y|X}(Y \neq \hat{y} | x)$$

- But, we do not know $P_{X,Y}$ and 0-1 loss cannot be optimized with gradient descent

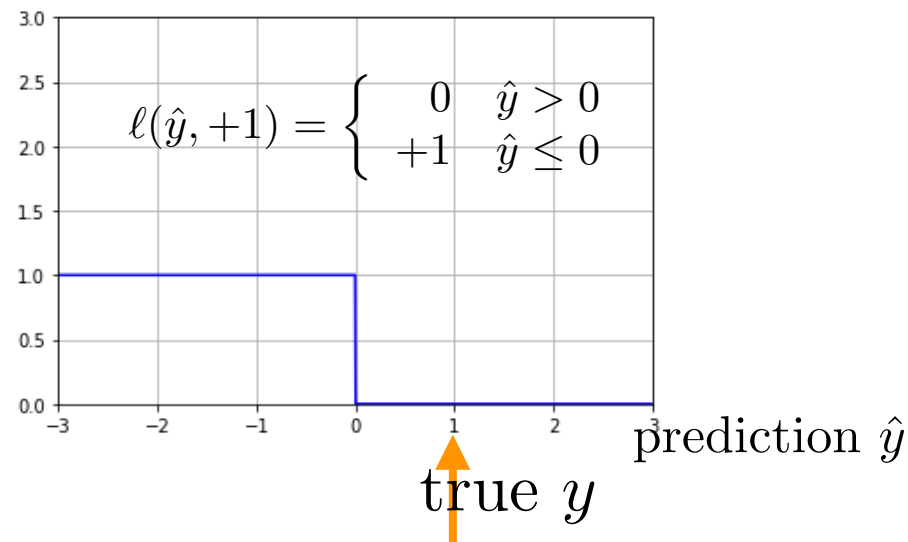
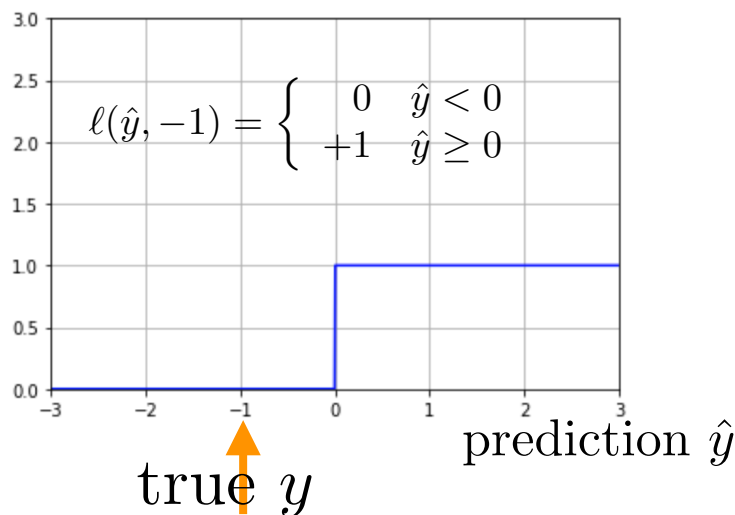
Binary Classification with square loss

- **Learn** a linear model: $f : x \mapsto \hat{y} = b + x^T w$
 - x input/features, $y \in \{-1, +1\}$ label in target classes
 - Prediction: $\text{sign}(\hat{y})$
 - **Square loss function** $\ell(b + x^T w, y) = (y - x^T w - b)^2$
 - This is the same as treating this as a linear regression problem
- $$(\hat{w}, \hat{b}) = \arg \min_{b, w} \sum_{i=1}^n (y_i - (b + x_i^T w))^2$$
- What is the strengths and weaknesses? Goes back up in the “correct” regime



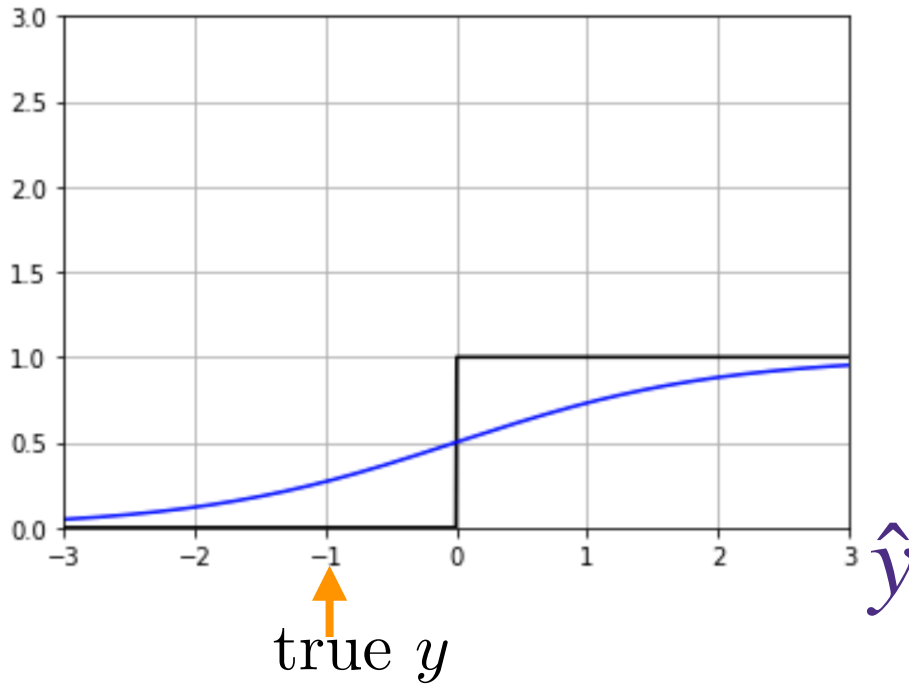
Looking for a better loss function

- we get better results using loss functions that
 - approximate, or captures the flavor of, the 0-1 loss
 - is more easily optimized (e.g. convex and/or non-zero derivatives)
- concretely, we want a **loss function**
 - with $\ell(\hat{y}, -1)$ small when $\hat{y} < 0$ and larger when $\hat{y} > 0$
 - with $\ell(\hat{y}, 1)$ small when $\hat{y} > 0$ and larger when $\hat{y} < 0$
 - Which has other nice characteristics, e.g., differentiable or convex

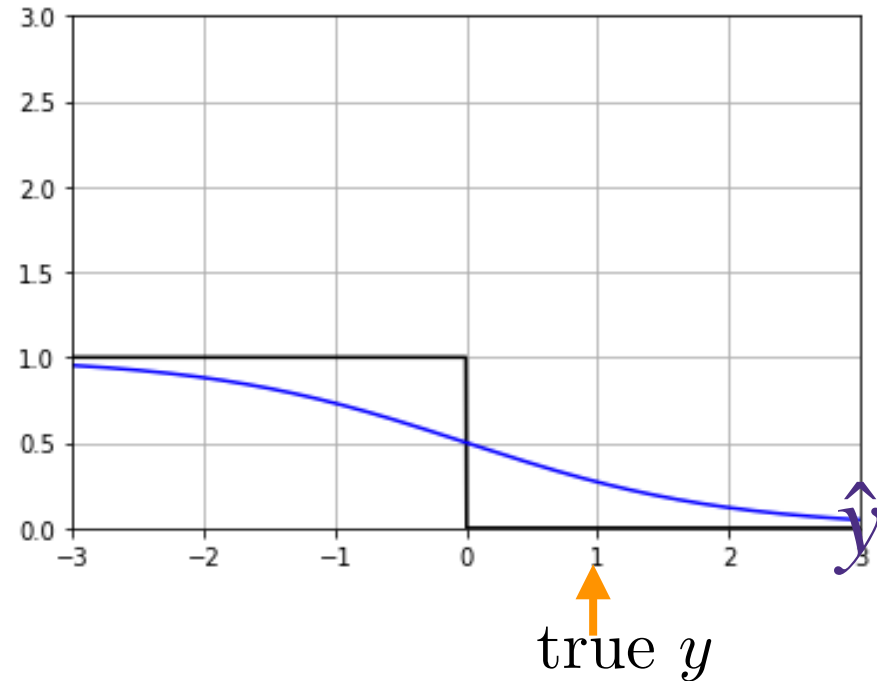


Sigmoid loss $\ell(\hat{y}, y) = \frac{1}{1 + e^{y\hat{y}}}$

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



$$\ell(\hat{y}, +1) = \frac{1}{1 + e^{\hat{y}}}$$



- differentiable approximation of 0-1 loss
- What is the weakness? **not convex in $\hat{y}$**
- the two losses sum to one

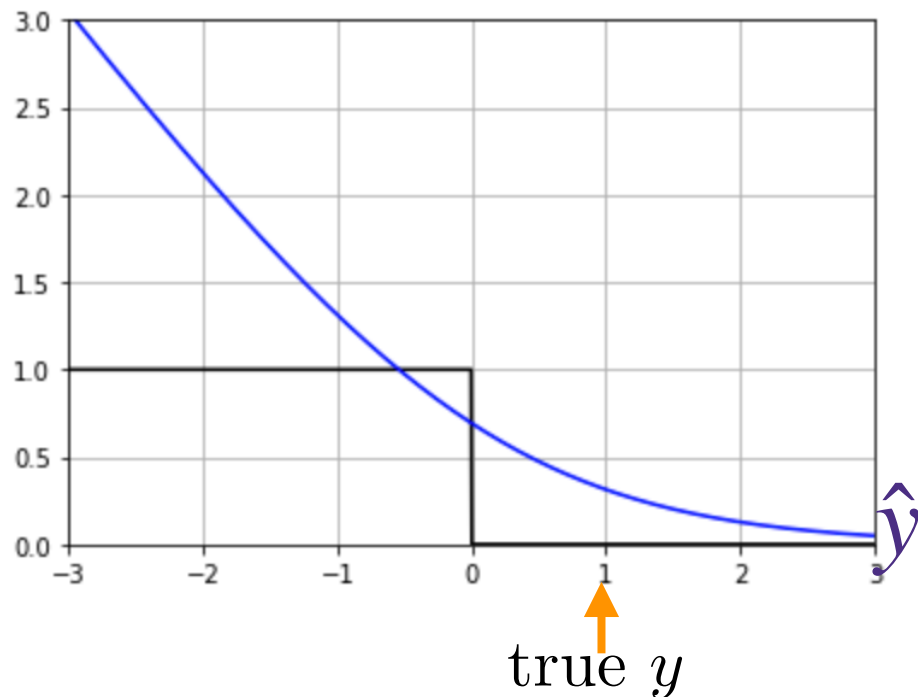
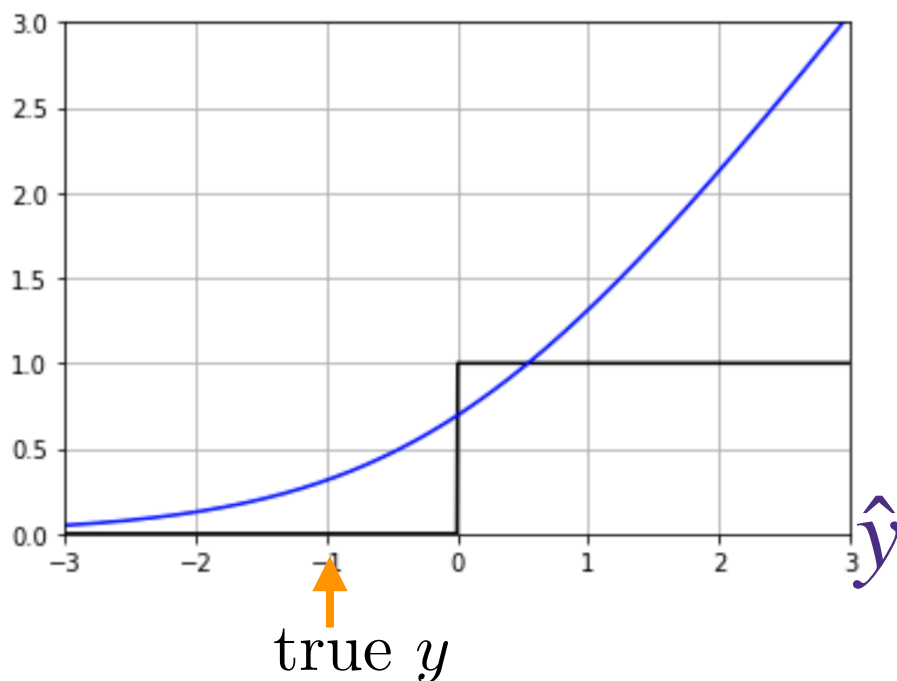
$$\frac{1}{1 + e^{-\hat{y}}} + \frac{1}{1 + e^{\hat{y}}} = \frac{e^{\hat{y}}}{e^{\hat{y}} + 1} + \frac{1}{1 + e^{\hat{y}}} = 1$$

- softer (or smoothed) version of the 0-1 loss

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

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

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



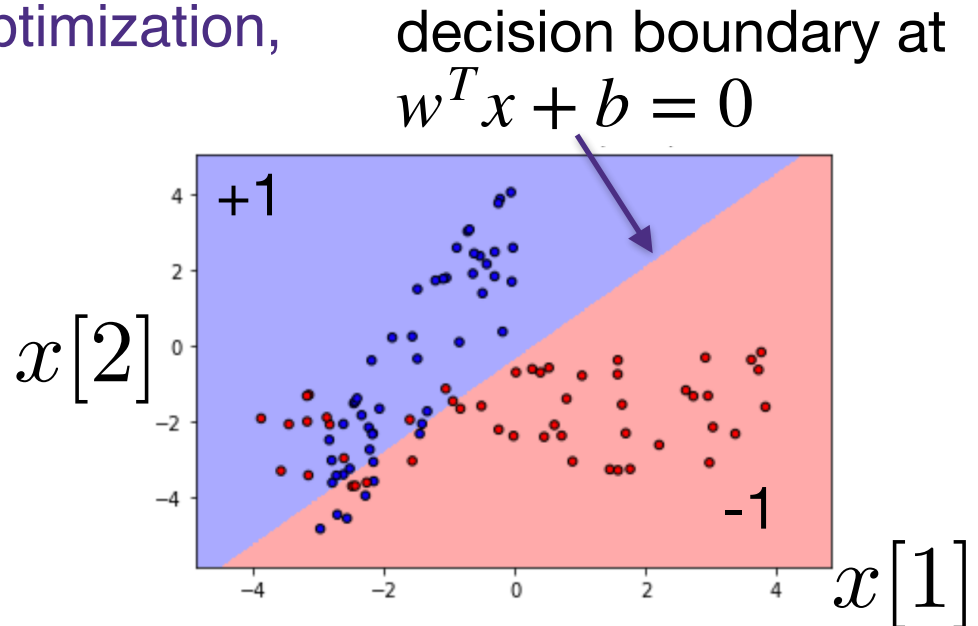
- differentiable and convex in $\hat{y}$
- how do we show $\ell(\cdot, y)$ is convex?
- approximation of 0-1
- Most popular choice of a loss function for classification problems

Logistic regression for binary classification

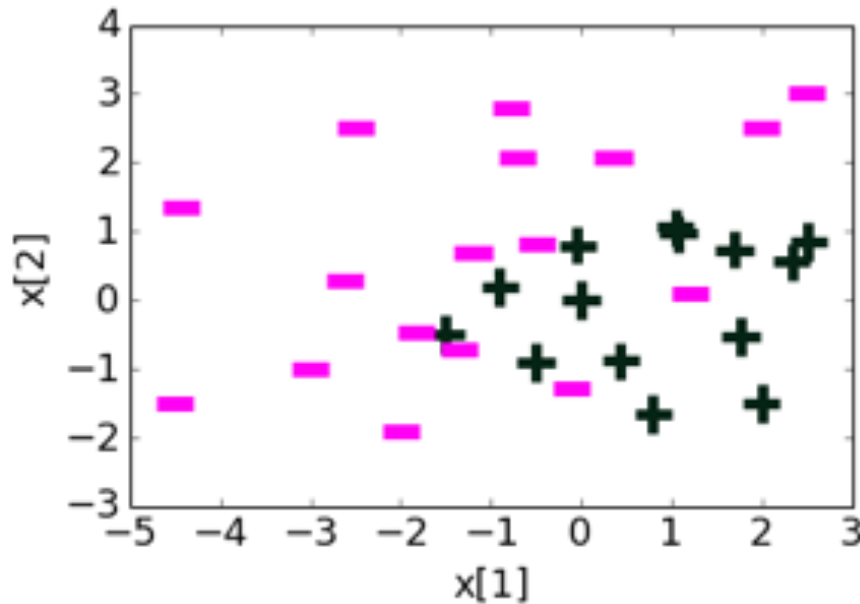
- Data $\mathcal{D} = \{(x_i \in \mathbb{R}^d, y_i \in \{-1, +1\})\}_{i=1}^n$
- Model: $\hat{y} = x^T w + b$
- Loss function: logistic loss $\ell(\hat{y}, y) = \log(1 + e^{-y\hat{y}})$
- Optimization: solve for

$$(\hat{b}, \hat{w}) = \arg \min_{b, w} \sum_{i=1}^n \log(1 + e^{-y_i(b + x_i^T w)})$$

- As this is a **smooth convex** optimization, it can be solved efficiently using gradient descent
- Prediction: $\text{sign}(b + x^T w)$



Example: adding more polynomial features



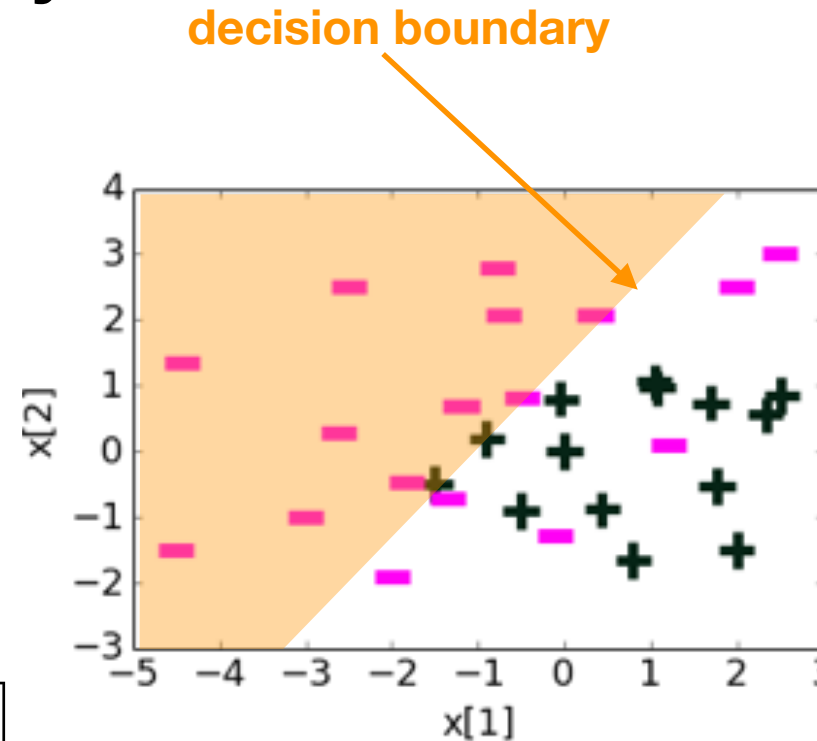
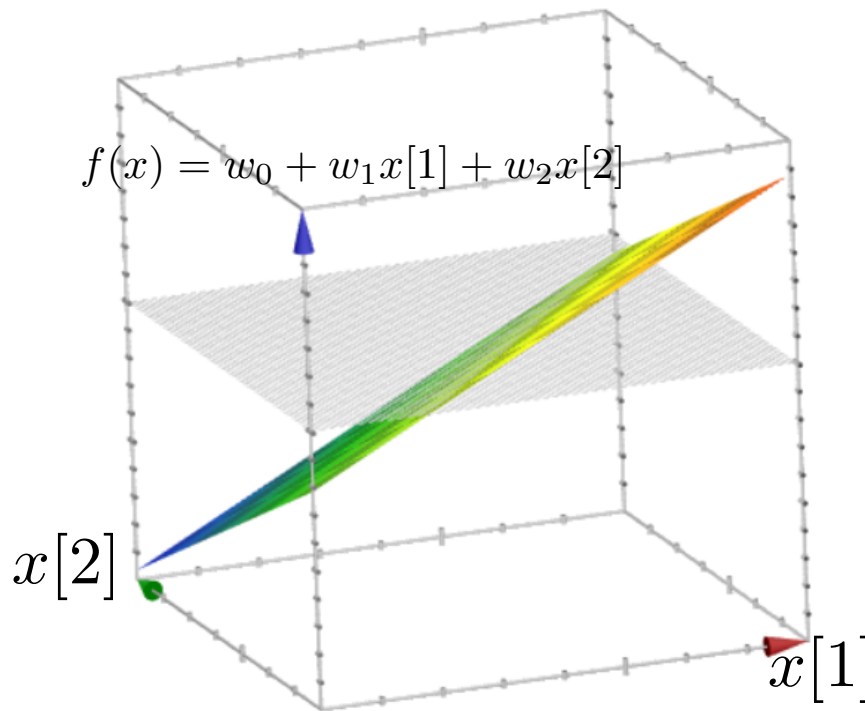
Polynomial
features

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

- data: $\mathbf{x}$ in 2-dimensions, $\mathbf{y}$ in $\{+1, -1\}$
- features: polynomials
- model: linear on polynomial features

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

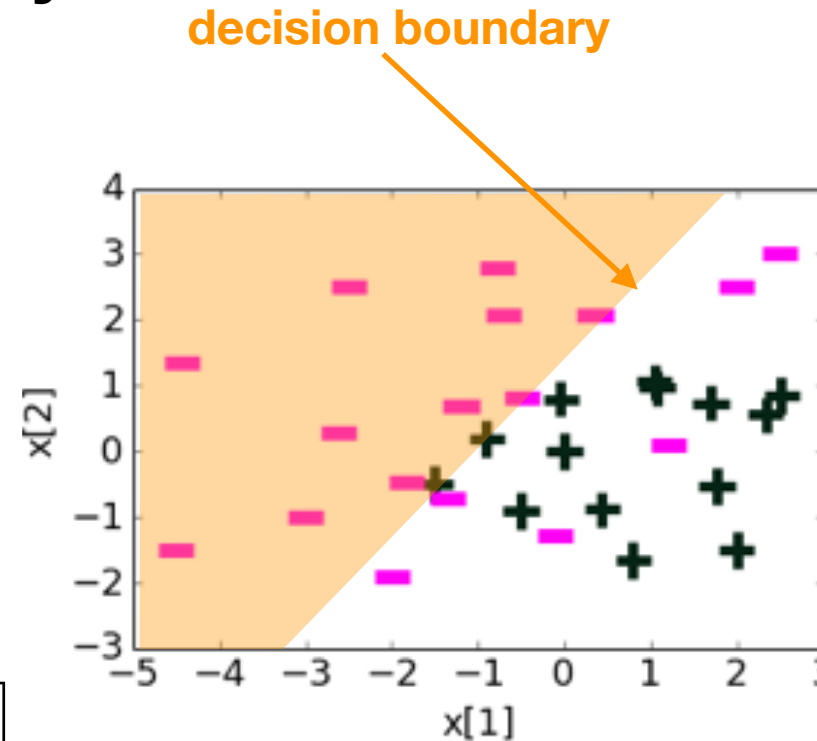
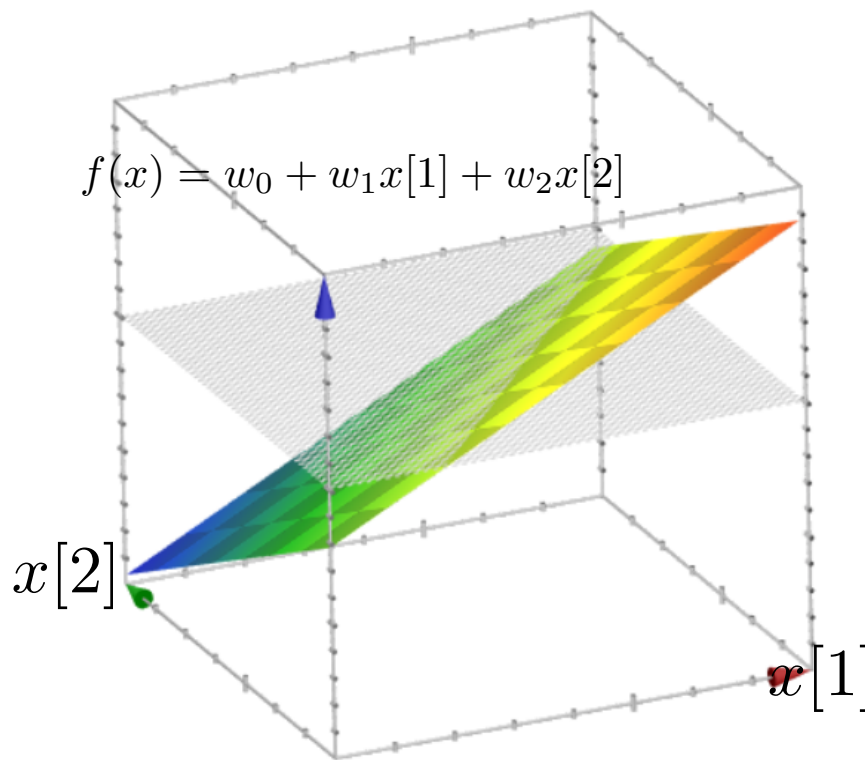
Learned decision boundary



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**

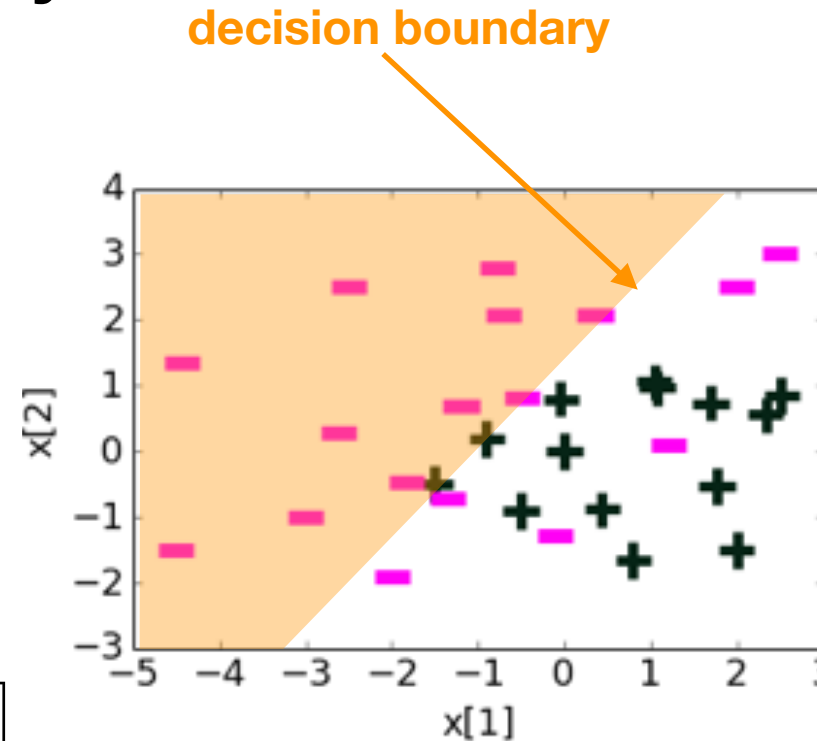
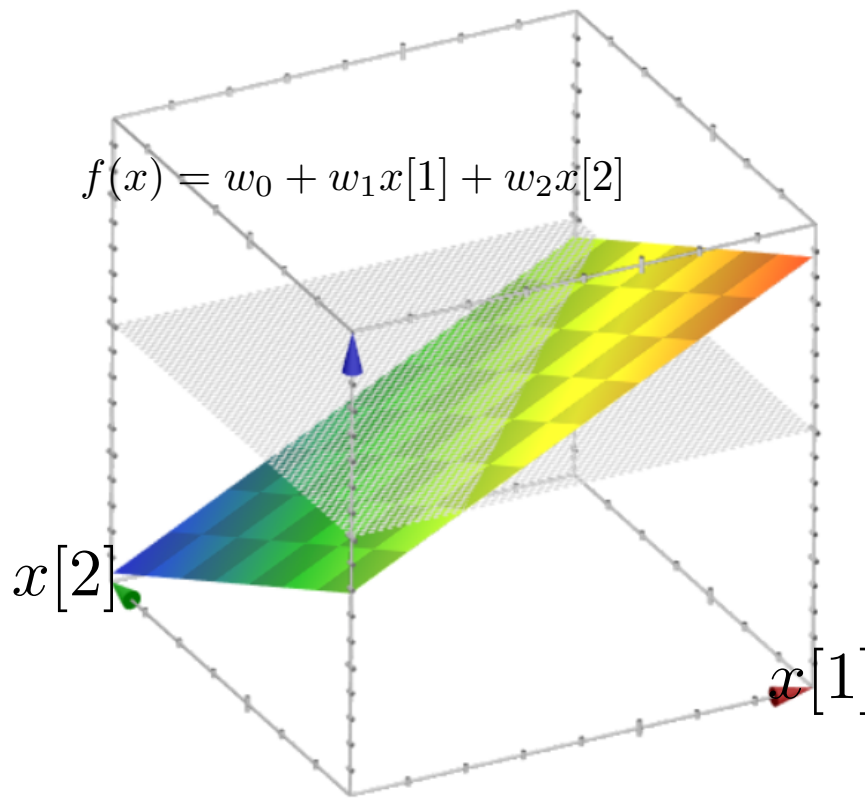
Learned decision boundary



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**

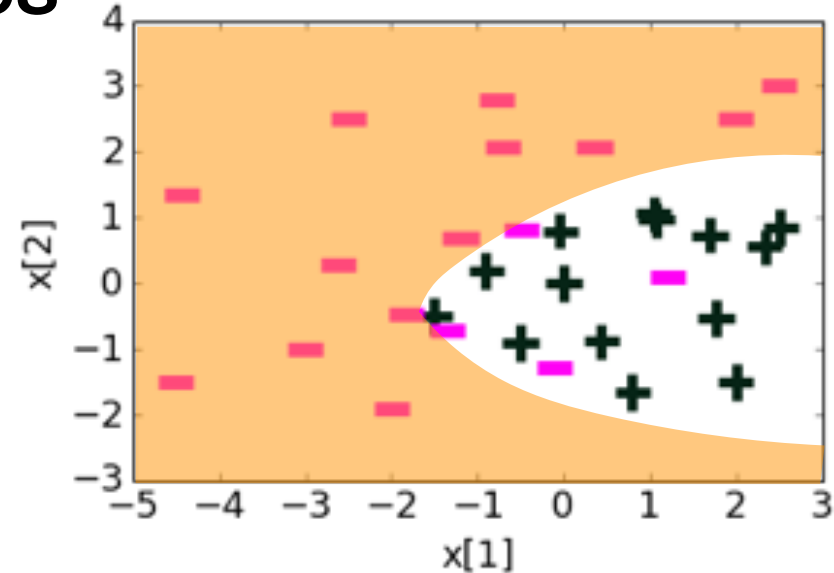
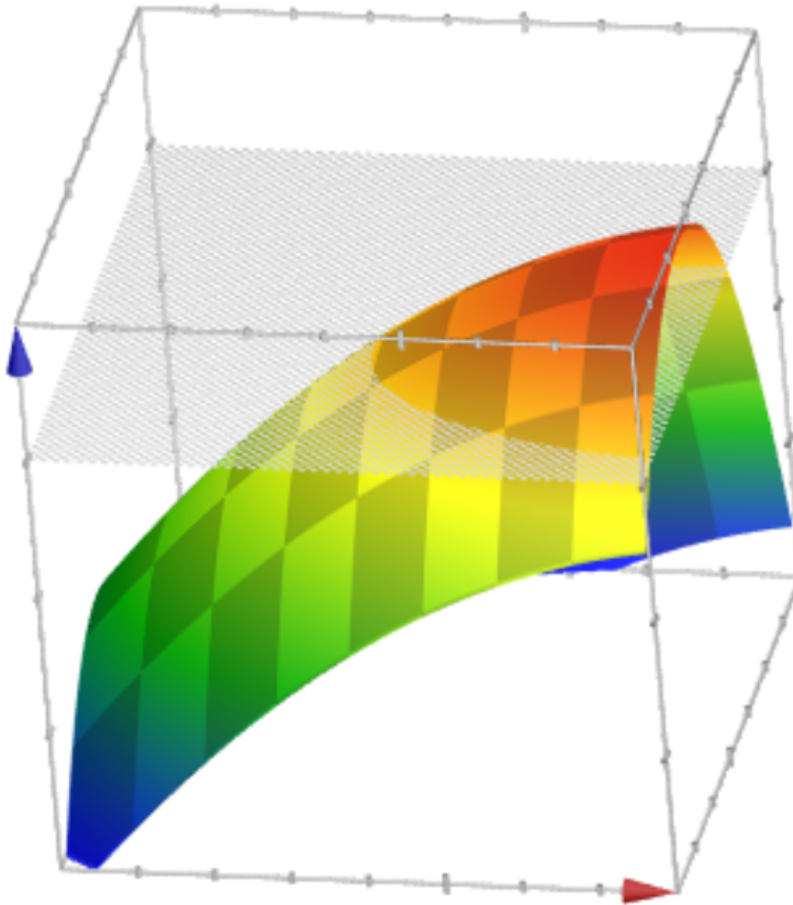
Learned decision boundary



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**

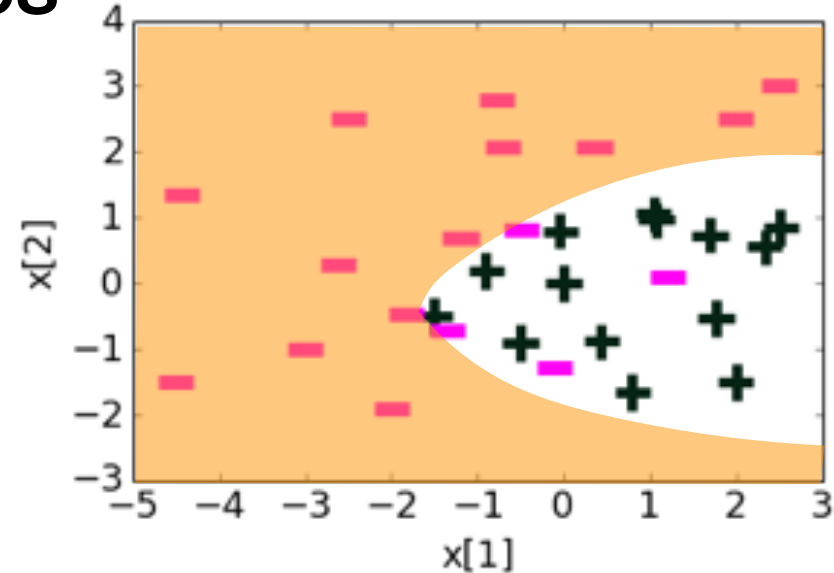
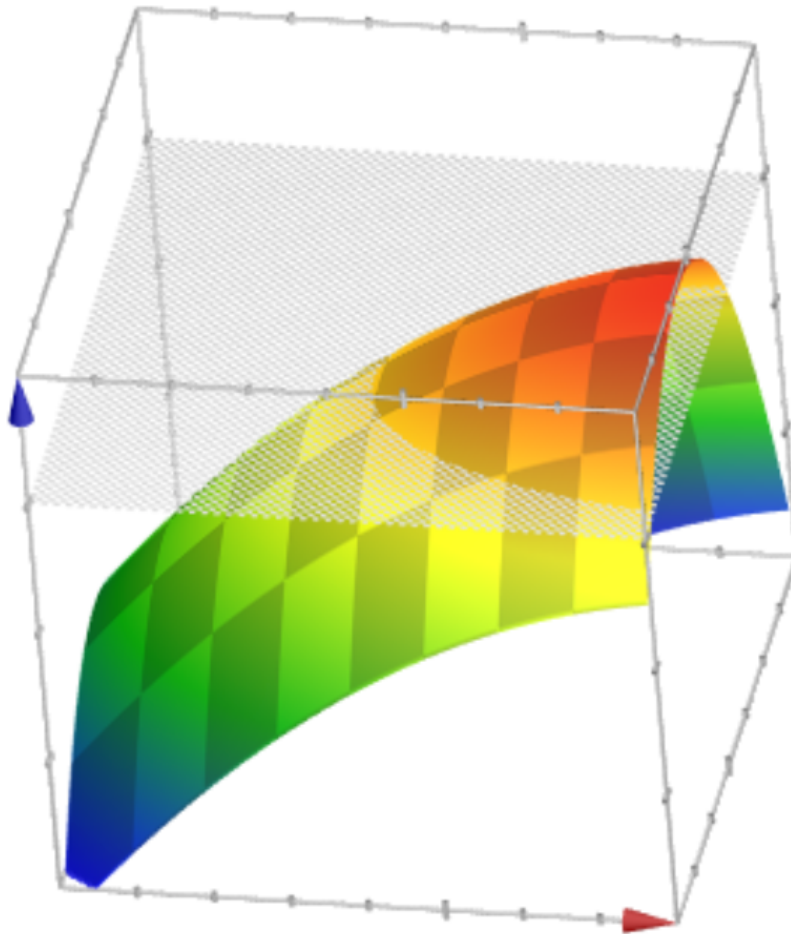
Adding quadratic features



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

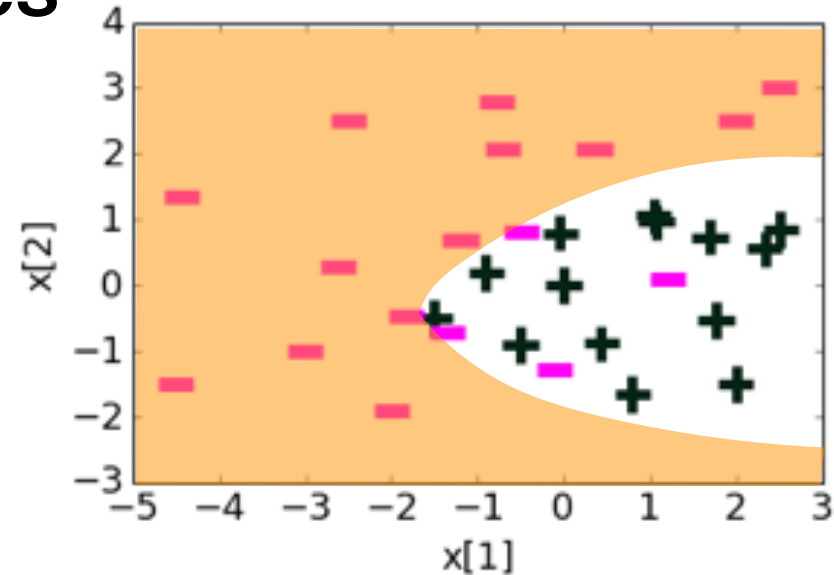
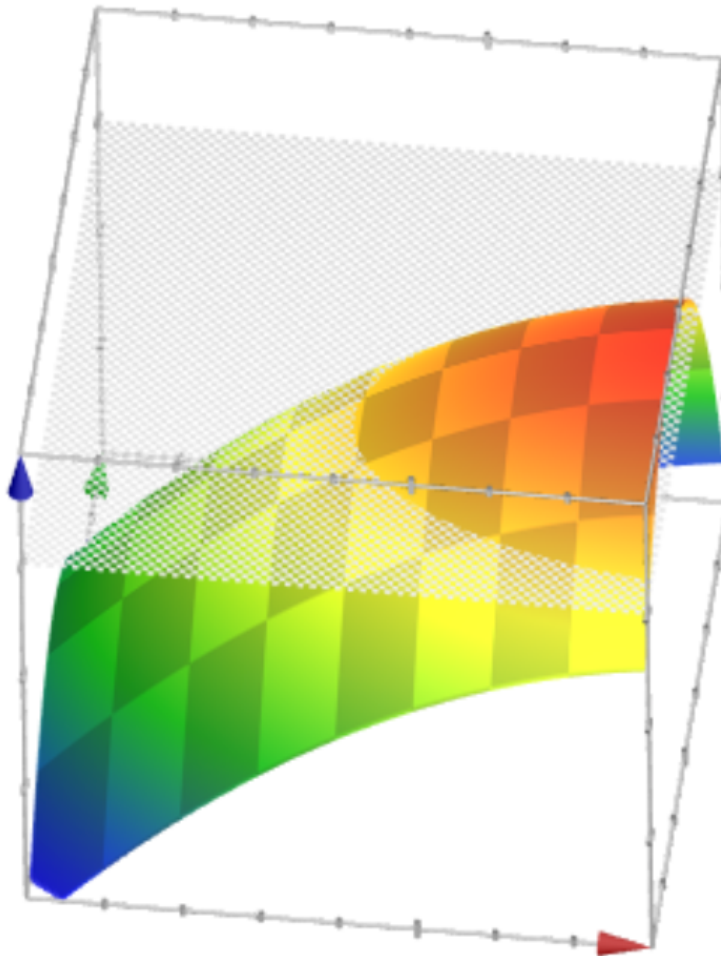
Adding quadratic features



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

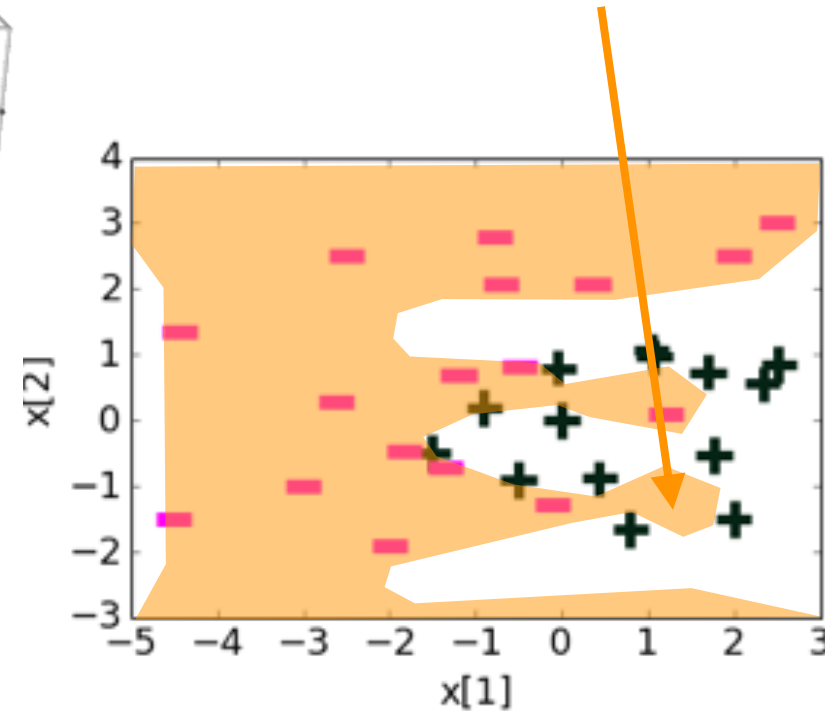
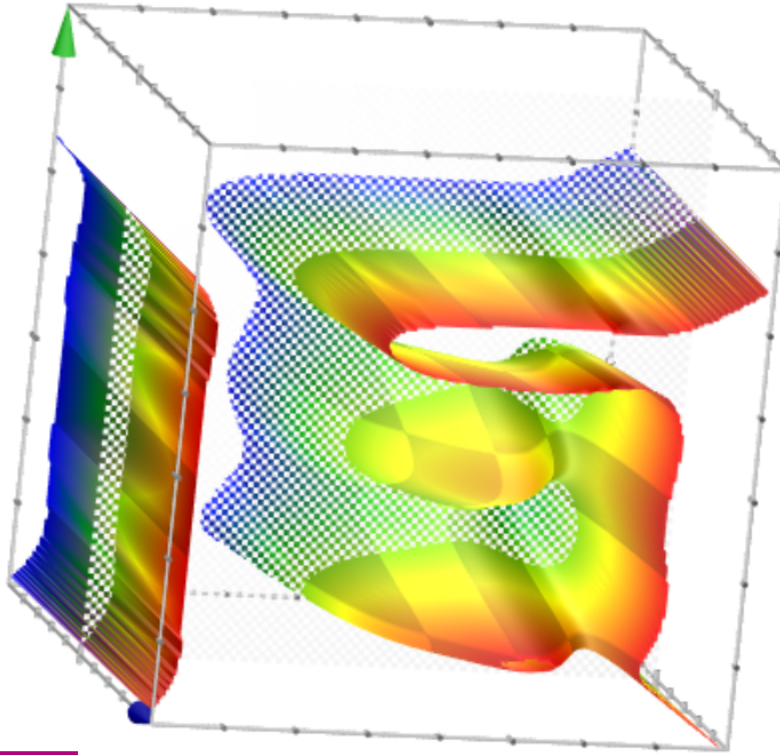


Feature	Value	Coefficient
$h_0(x)$	1	1.68
$h_1(x)$	$x[1]$	1.39
$h_2(x)$	$x[2]$	-0.59
$h_3(x)$	$(x[1])^2$	-0.17
$h_4(x)$	$(x[2])^2$	-0.96
$h_5(x)$	$x[1]x[2]$	Omitted

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

Adding higher degree polynomial features

Overfitting leads to non-generalization

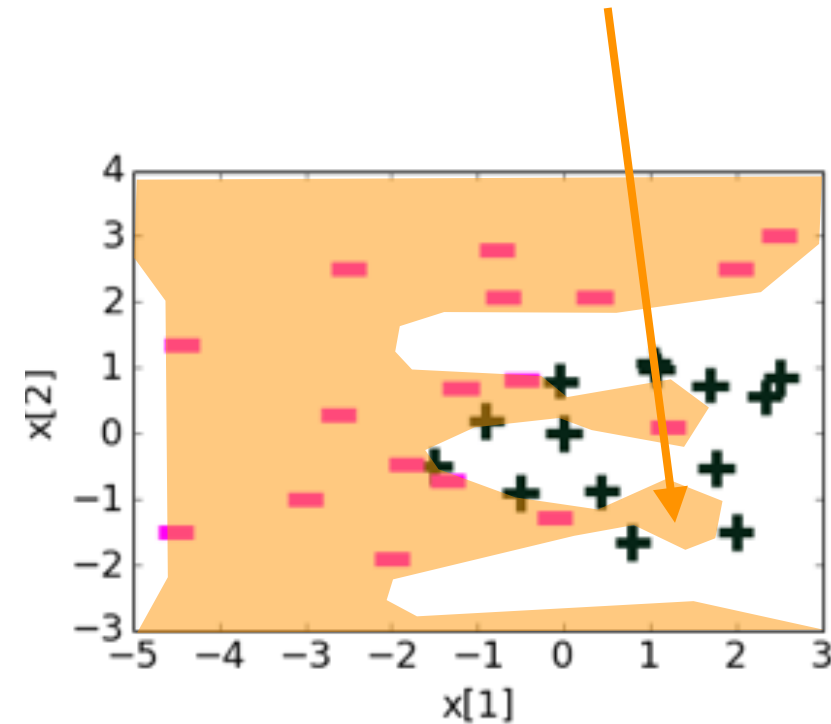
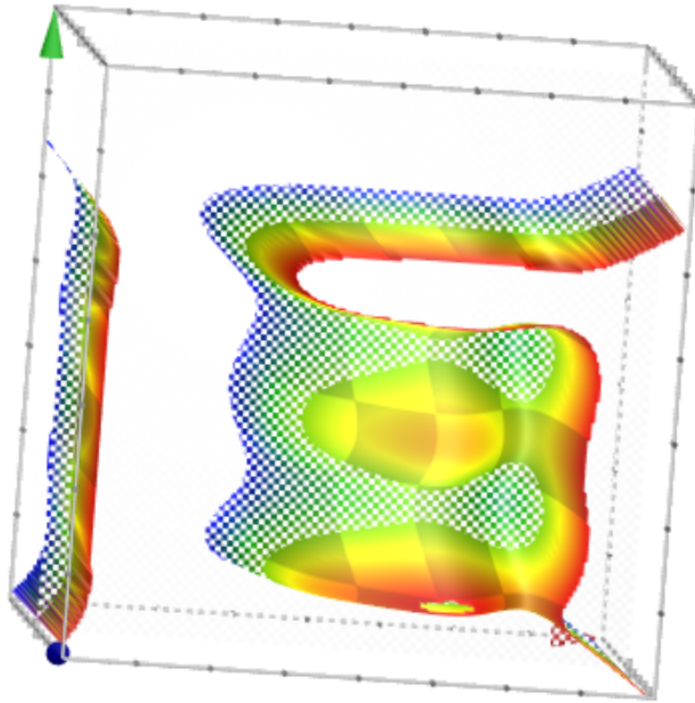


Feature	Value	Coefficient learned
$h_0(x)$	1	21.6
$h_1(x)$	$x[1]$	5.3
$h_2(x)$	$x[2]$	-42.7
$h_3(x)$	$(x[1])^2$	-15.9
$h_4(x)$	$(x[2])^2$	-48.6
$h_5(x)$	$(x[1])^3$	-11.0
$h_6(x)$	$(x[2])^3$	67.0
$h_7(x)$	$(x[1])^4$	1.5
$h_8(x)$	$(x[2])^4$	48.0
$h_9(x)$	$(x[1])^5$	4.4
$h_{10}(x)$	$(x[2])^5$	-14.2
$h_{11}(x)$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

Coefficient values getting large

Adding higher degree polynomial features

Overfitting leads to non-generalization

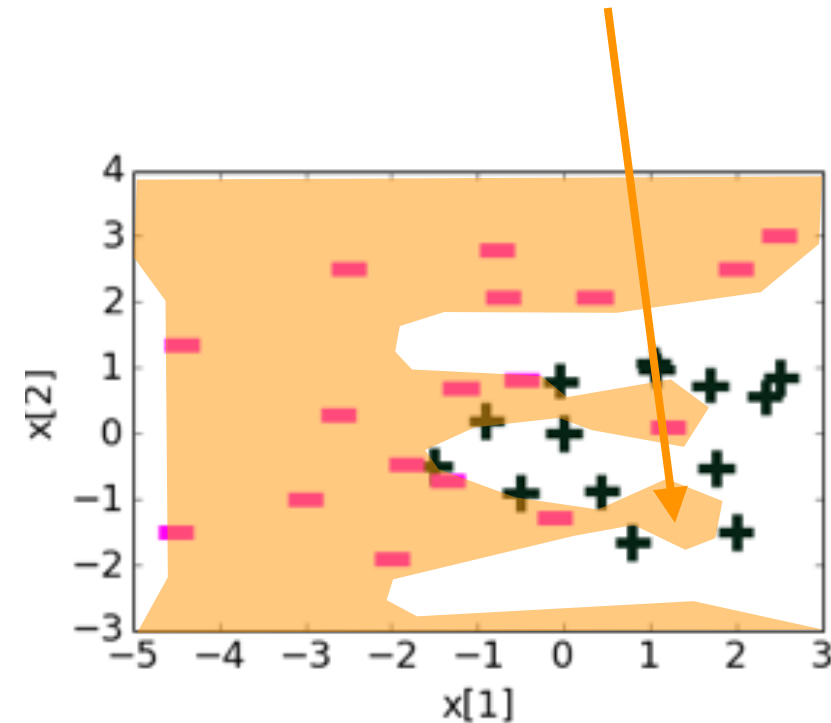
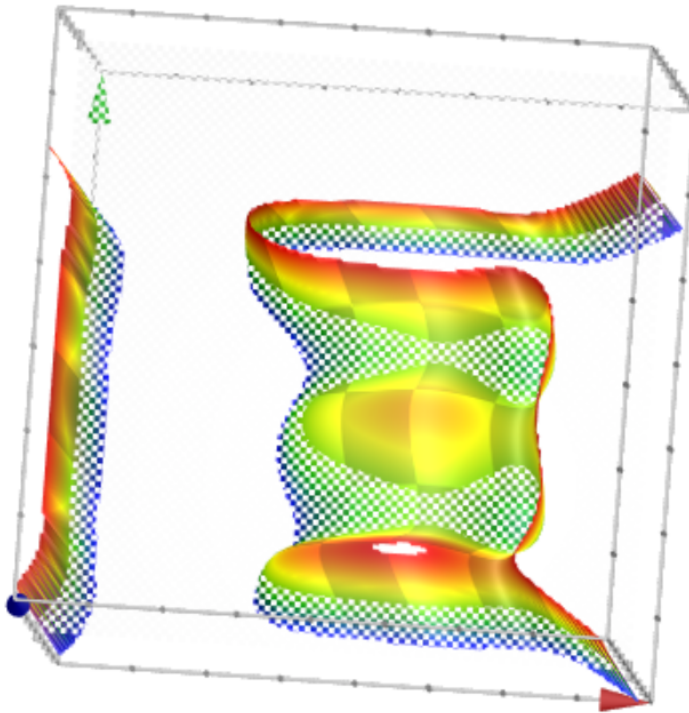


Feature	Value	Coefficient learned
$h_0(x)$	1	21.6
$h_1(x)$	$x[1]$	5.3
$h_2(x)$	$x[2]$	-42.7
$h_3(x)$	$(x[1])^2$	-15.9
$h_4(x)$	$(x[2])^2$	-48.6
$h_5(x)$	$(x[1])^3$	-11.0
$h_6(x)$	$(x[2])^3$	67.0
$h_7(x)$	$(x[1])^4$	1.5
$h_8(x)$	$(x[2])^4$	48.0
$h_9(x)$	$(x[1])^5$	4.4
$h_{10}(x)$	$(x[2])^5$	-14.2
$h_{11}(x)$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

Coefficient values getting large

Adding higher degree polynomial features

Overfitting leads to non-generalization



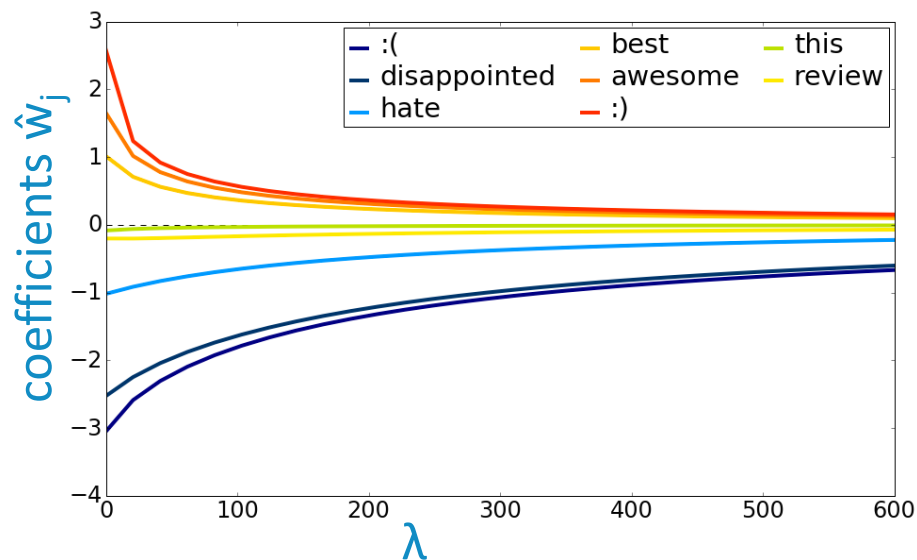
Feature	Value	Coefficient learned
$h_0(x)$	1	21.6
$h_1(x)$	$x[1]$	5.3
$h_2(x)$	$x[2]$	-42.7
$h_3(x)$	$(x[1])^2$	-15.9
$h_4(x)$	$(x[2])^2$	-48.6
$h_5(x)$	$(x[1])^3$	-11.0
$h_6(x)$	$(x[2])^3$	67.0
$h_7(x)$	$(x[1])^4$	1.5
$h_8(x)$	$(x[2])^4$	48.0
$h_9(x)$	$(x[1])^5$	4.4
$h_{10}(x)$	$(x[2])^5$	-14.2
$h_{11}(x)$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

Coefficient values getting large

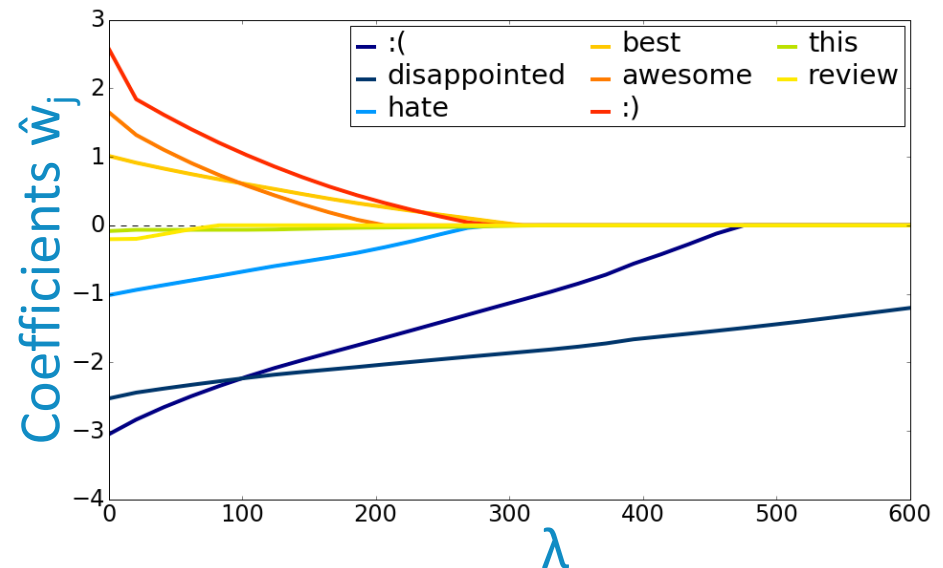
- Overfitting leads to very large values of $f(x) = w_0 h_0(x) + w_1 h_1(x) + w_2 h_2(x) + \dots$

Regularization path

ℓ_2 regularizer: $\|W\|_2^2 = |w_1|^2 + \dots + |w_d|^2$



ℓ_1 regularizer: $\|w\|_1 = |w_1| + \dots + |w_d|$



- Absolute regularizer (a.k.a ℓ_1 regularizer) gives sparse parameters, which is desired for interpretability, feature selection, and efficiency

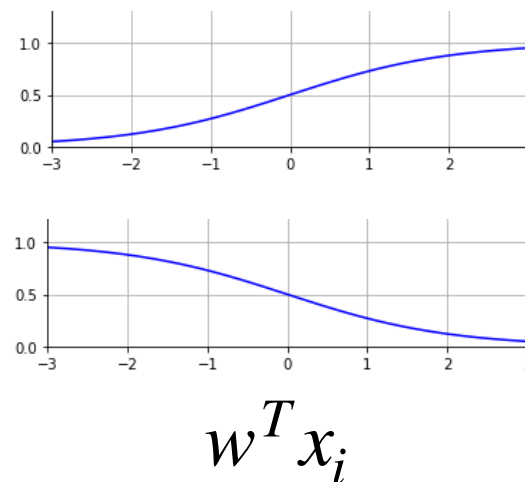
Probabilistic interpretation of **logistic regression**

- just as Maximum Likelihood Estimator (MLE) under linear model and additive Gaussian noise model recovers **linear least squares**,
- we study a particular noise model that recovers **logistic regression** as MLE
- a probabilistic noise model for Binary labels:

$$\mathbb{P}(y_i = +1 | x_i) = \frac{1}{1 + e^{-w^T x_i}}$$

$$\mathbb{P}(y_i = -1 | x_i) = \frac{1}{1 + e^{w^T x_i}}$$

with a ground truth model parameter $w \in \mathbb{R}^d$



- this function $\sigma(z) = \frac{1}{1 + e^{-z}}$ is called a **logistic function** (not to be confused with logistic loss, which is different) or a **sigmoid function**
- if we know that the data came from such a model, but do not know the ground truth parameter $w \in \mathbb{R}^d$, we can apply MLE to find the best w
- this MLE recovers the logistic regression algorithm, exactly

Maximum Likelihood Estimator (MLE)

- if the data came from a probabilistic model model: $\left(\underbrace{\frac{1}{1 + e^{-w^T x}}}_{\mathbb{P}(y_i = +1 | x_i)}, \underbrace{\frac{1}{1 + e^{w^T x}}}_{\mathbb{P}(y_i = -1 | x_i)} \right)$
- log-likelihood of observing a data point (x_i, y_i) is

$$\text{log-likelihood} = \log \left(\mathbb{P}(y_i | x_i) \right) = \begin{cases} \log \left(\frac{1}{1 + e^{-w^T x_i}} \right) & \text{if } y_i = +1 \\ \log \left(\frac{1}{1 + e^{w^T x_i}} \right) & \text{if } y_i = -1 \end{cases}$$

- Maximum Likelihood Estimator is the one that maximizes the sum of all log-likelihoods on training data points

$$\hat{w}_{\text{MLE}} = \arg \max_w \mathbb{P}(\{y_1, \dots, y_n\} | \{x_1, \dots, x_n\})$$

$$= \arg \max_w \prod_{i=1}^n \mathbb{P}(y_i | x_i) \quad \text{(independence)}$$

$$= \arg \max_w \sum_{i: y_i = -1} \log \left(\frac{1}{1 + e^{w^T x_i}} \right) + \sum_{i: y_i = 1} \log \left(\frac{1}{1 + e^{-w^T x_i}} \right) \quad \text{(substitution)}$$

- notice that this is exactly the **logistic regression**:

$$\hat{w}_{\text{logistic}} = \arg \min_w \frac{1}{n} \left(\sum_{i:y_i=-1} \log(1 + e^{w^T x_i}) + \sum_{i:y_i=1} \log(1 + e^{-w^T x_i}) \right)$$

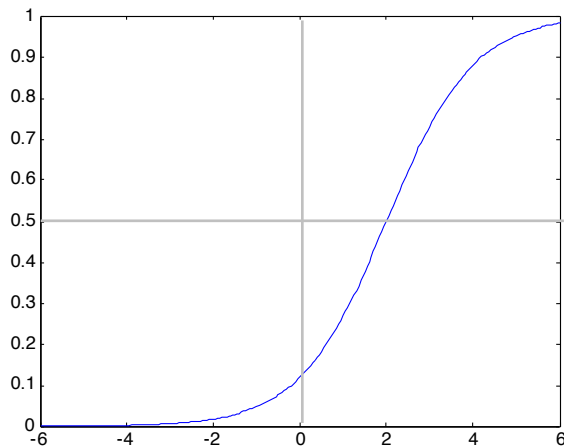
- once we have trained a model $\hat{w}_{\text{logistic}}$, we can make a hard prediction $\hat{v}$ of the label at an input example x

$$\begin{aligned} \hat{v} &= \begin{cases} +1 & \text{if } \mathbb{P}(+1|x) \geq \mathbb{P}(-1|x) \\ -1 & \text{otherwise} \end{cases} \\ &= \begin{cases} +1 & \text{if } \frac{1}{1+e^{-w^T x}} \geq \frac{1}{1+e^{w^T x}} \\ -1 & \text{otherwise} \end{cases} \\ &= \begin{cases} +1 & \text{if } 1 \leq e^{2w^T x} \\ -1 & \text{otherwise} \end{cases} \\ &= \text{sign}(w^T x) \end{aligned}$$

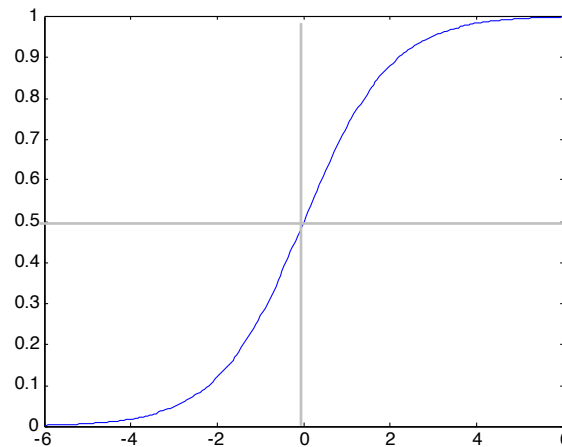
Understanding the sigmoid

$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{w_0 + \sum_i w_i x_i}}$$

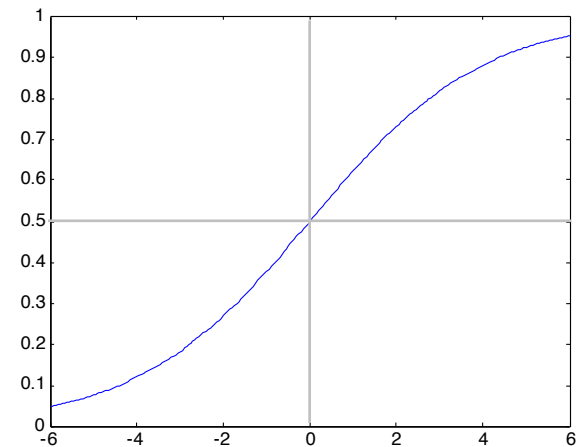
$w_0 = -2, w_1 = -1$



$w_0 = 0, w_1 = -1$



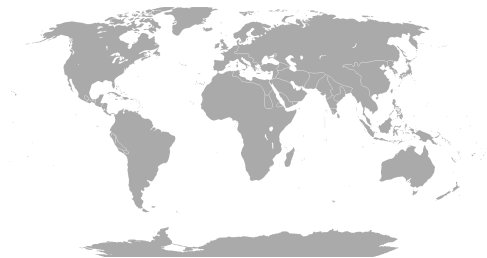
$w_0 = 0, w_1 = -0.5$



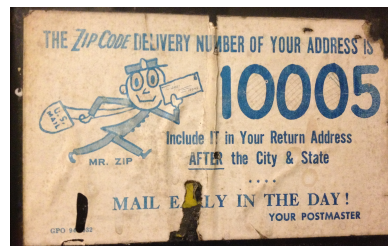
Multi-class regression

How do we encode categorical data y ?

- so far, we considered Binary case where there are two categories
- encoding y is simple: $\{+1, -1\}$
- multi-class classification predicts categorical y
- taking values in $C = \{c_1, \dots, c_k\}$
- c_j 's are called **classes** or **labels**
- examples:



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



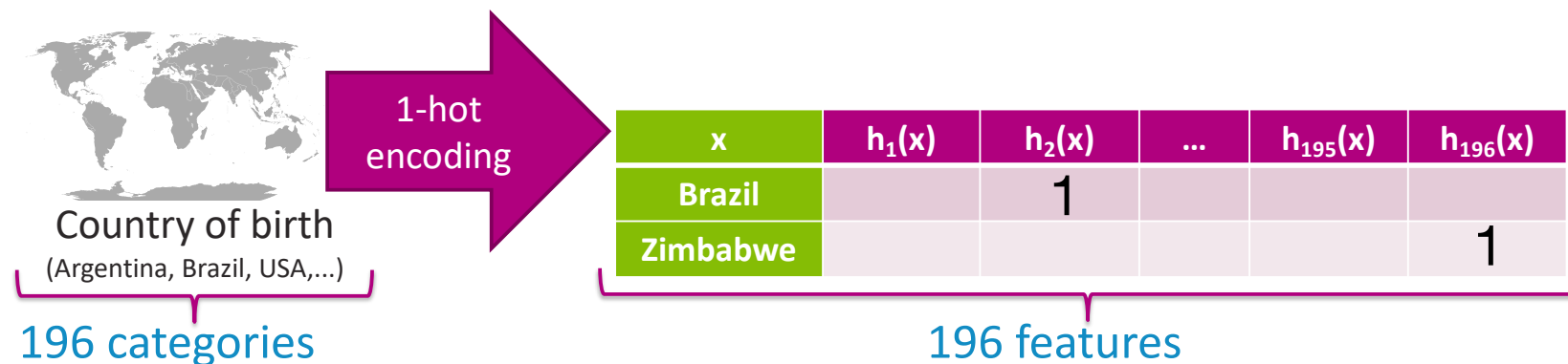
Zipcode
(10005, 98195,...)

All English words

- a **k-class classifier** predicts y given x

Embedding c_j 's in real values

- for optimization we need to **embed** raw categorical c_j 's into real valued vectors
- there are many ways to embed categorical data
 - True->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)
 - Ordered sequence:
(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 $\{c_1, \dots, c_k\}$ with k categories

we use one-hot encoding, s.t. $y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ implies that $y = c_1$

- model: linear vector-function makes a linear prediction $\hat{y} \in \mathbb{R}^k$

$$\hat{y}_i = f(x_i) = w^T x_i \in \mathbb{R}^k$$

with model parameter matrix $w \in \mathbb{R}^{d \times k}$ and sample $x_i \in \mathbb{R}^d$

$$f(x_i) = \begin{bmatrix} f_1(x_i) \\ f_2(x_i) \\ \vdots \\ f_k(x_i) \end{bmatrix} = \underbrace{\begin{bmatrix} w_{1,0} & w_{1,1} & w_{1,2} & \cdots \\ w_{2,0} & w_{2,1} & w_{2,2} & \cdots \\ \vdots & & & \\ w_{k,0} & w_{k,1} & w_{k,2} & \cdots \end{bmatrix}}_{w^T} \underbrace{\begin{bmatrix} 1 \\ x_i[1] \\ \vdots \\ x_i[d] \end{bmatrix}}_{x_i} = \begin{bmatrix} w_{1,0} + w_{1,1}x_i[1] + w_{1,2}x_i[2] + \cdots \\ w_{2,0} + w_{2,1}x_i[1] + w_{2,2}x_i[2] + \cdots \\ \vdots \\ w_{k,0} + w_{k,1}x_i[1] + w_{k,2}x_i[2] + \cdots \end{bmatrix}$$

$$w = \begin{bmatrix} w[:,1] & w[:,2] & \cdots & w[:,k] \end{bmatrix}$$

- Logistic regression

2 classes

$$\mathbb{P}(y_i = -1 | x_i) = \frac{1}{1 + e^{w^T x_i}}$$

$$\mathbb{P}(y_i = +1 | x_i) = \frac{1}{1 + e^{-w^T x_i}} = \frac{e^{w^T x_i}}{1 + e^{w^T x_i}}$$

k classes

$$\mathbb{P}(y_i = c_1 | x_i) = \frac{e^{w[:,1]^T x_i}}{e^{w[:,1]^T x_i} + \dots + e^{w[:,k]^T x_i}}$$

$\vdots$

$$\mathbb{P}(y_i = c_k | x_i) = \frac{e^{w[:,k]^T x_i}}{e^{w[:,1]^T x_i} + \dots + e^{w[:,k]^T x_i}}$$

Without loss of generality setting $w[:,1]=0$ when $k = 2$ recovers the original binary class case

Maximum Likelihood Estimator

$$\text{maximize}_w \frac{1}{n} \sum_{i=1}^n \log(\mathbb{P}(y_i | x_i))$$

$$\text{maximize}_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \log\left(\frac{1}{1 + e^{-y_i w^T x_i}}\right)$$

$$\text{maximize}_{w \in \mathbb{R}^{d \times k}} \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^k \mathbf{I}\{y_i = c_j\} \log\left(\frac{e^{w[:,j]^T x_i}}{\sum_{j'=1}^k e^{w[:,j']^T x_i}}\right)$$

$\mathbf{I}\{y_i = j\}$ is an indicator that is one only if $y_i = j$

Questions?
