Logistic Regression

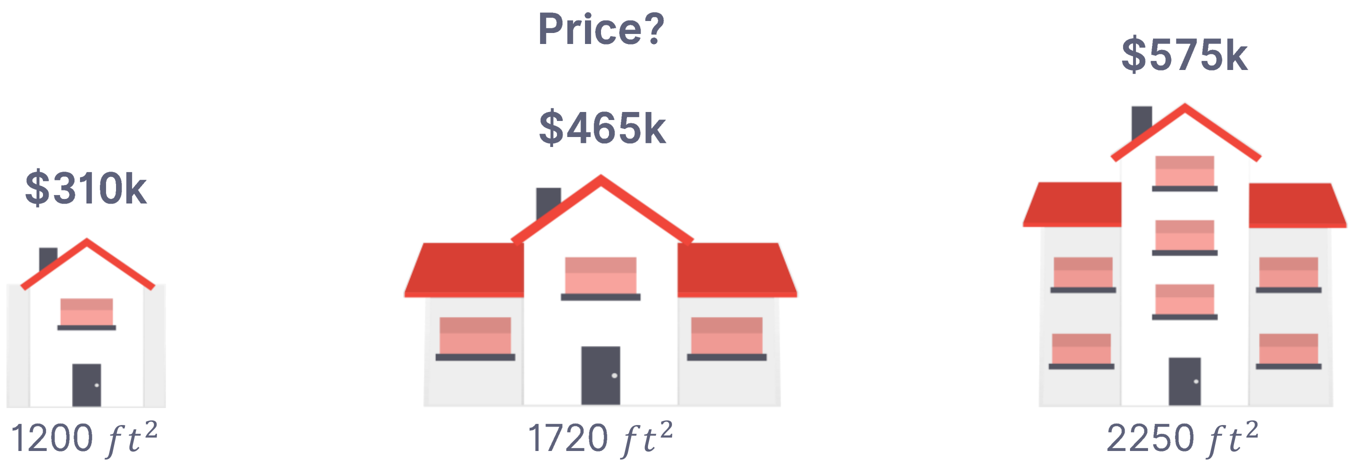
Predicting probabilities with an “S” curve

Faculty of Mathematics and Computer Science, University of Bucharest
and
Sparktech Software

Academic Year 2018/2019, 1st Semester

Classification vs. Regression

- Previously, we wanted to predict the price of a house, given its size.



Classification vs. Regression

- Previously, we wanted to predict the price of a house, given its size.
- Now, we want to predict if a house costs more than $450k or not.
    - Not only that, but we only have access to a set of samples with this information.

Price > $450k?

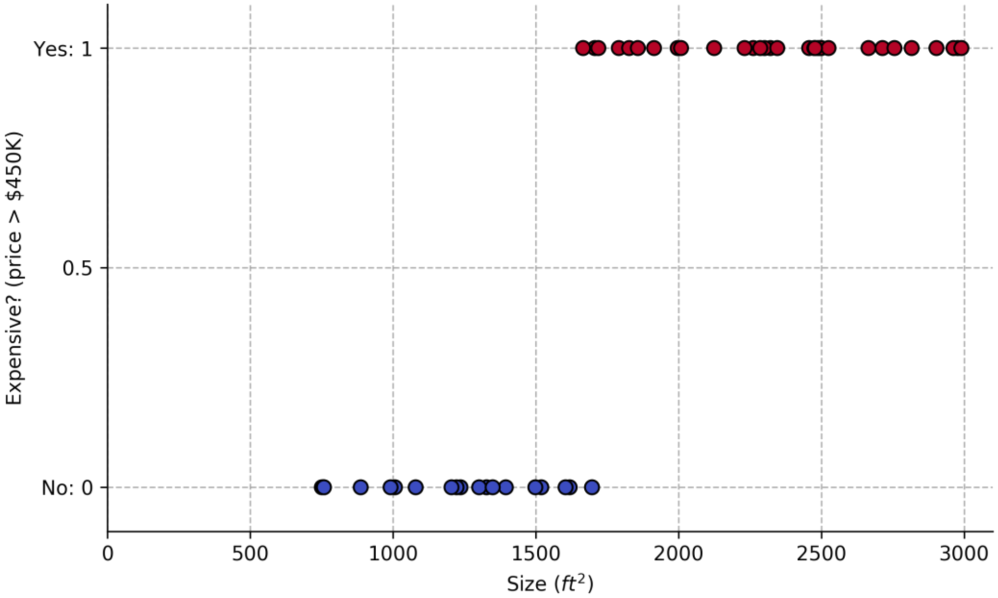
Classification vs. Regression

- Previously, we wanted to predict the price of a house, given its size.
- Now, we want to predict if a house costs more than $450k or not.
    - Not only that, but we only have access to a set of samples with this information.
    - Even better, we want a model which gives us the **likelihood** of a house costing more than $450k

Chances of Price > $450k?

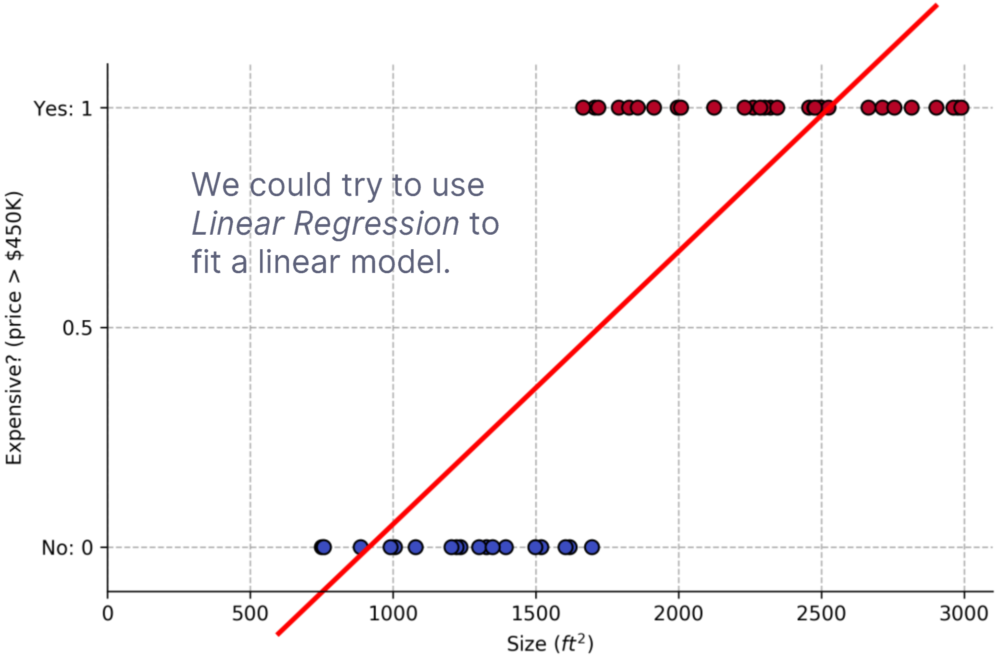
Classification vs. Regression

- Set of *expensive / non-expensive* house observations:



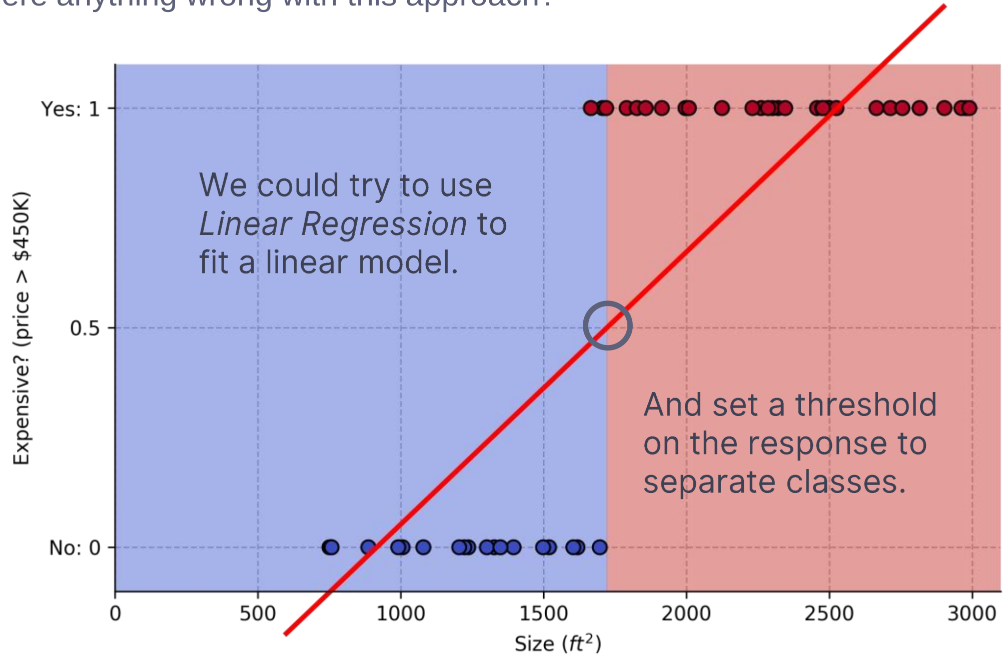
Classification vs. Regression

- Set of *expensive / non-expensive* house observations:



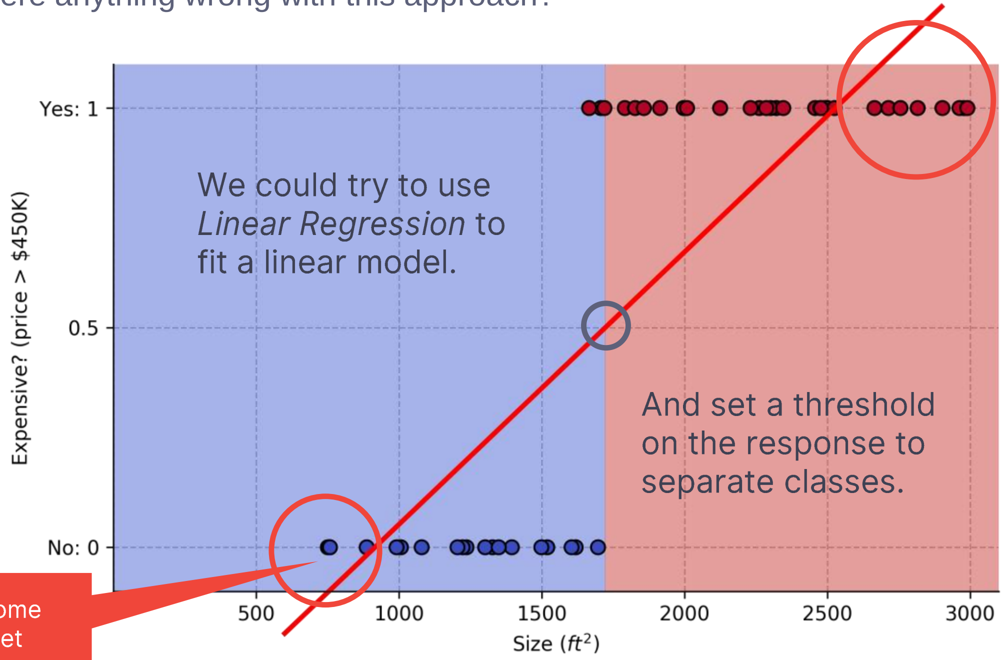
Classification vs. Regression

- Set of *expensive / non-expensive* house observations.
    - Is there anything wrong with this approach?



Classification vs. Regression

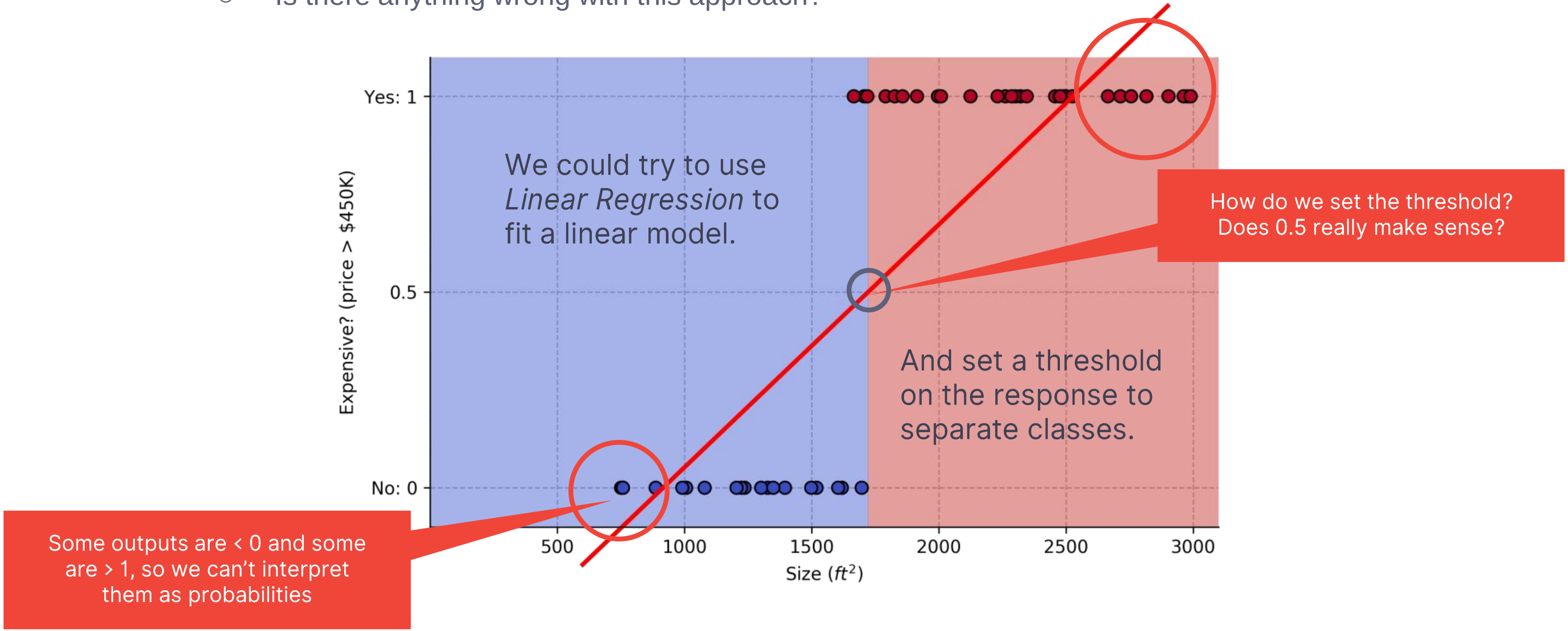
- Set of *expensive / non-expensive* house observations.
  - Is there anything wrong with this approach?



Some outputs are < 0 and some are > 1, so we can't interpret them as probabilities

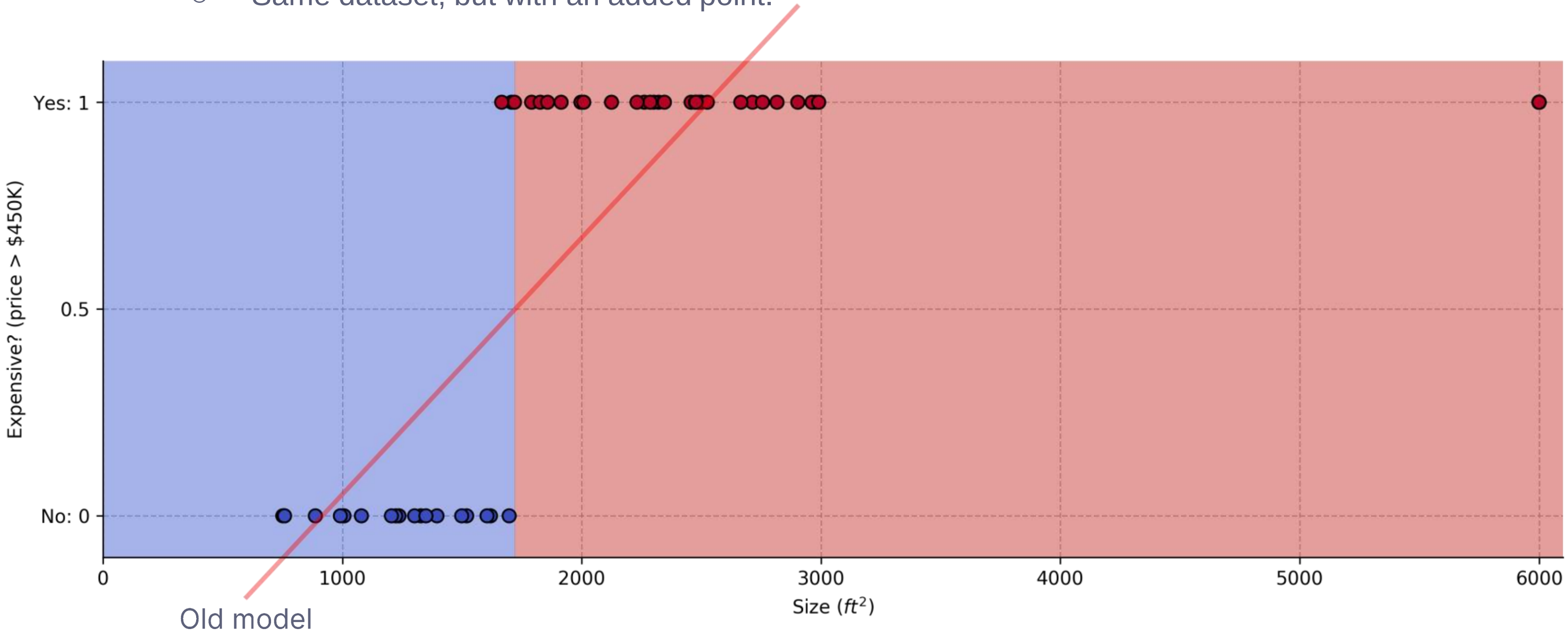
Classification vs. Regression

- Set of *expensive / non-expensive* house observations.
  - Is there anything wrong with this approach?



Classification vs. Regression

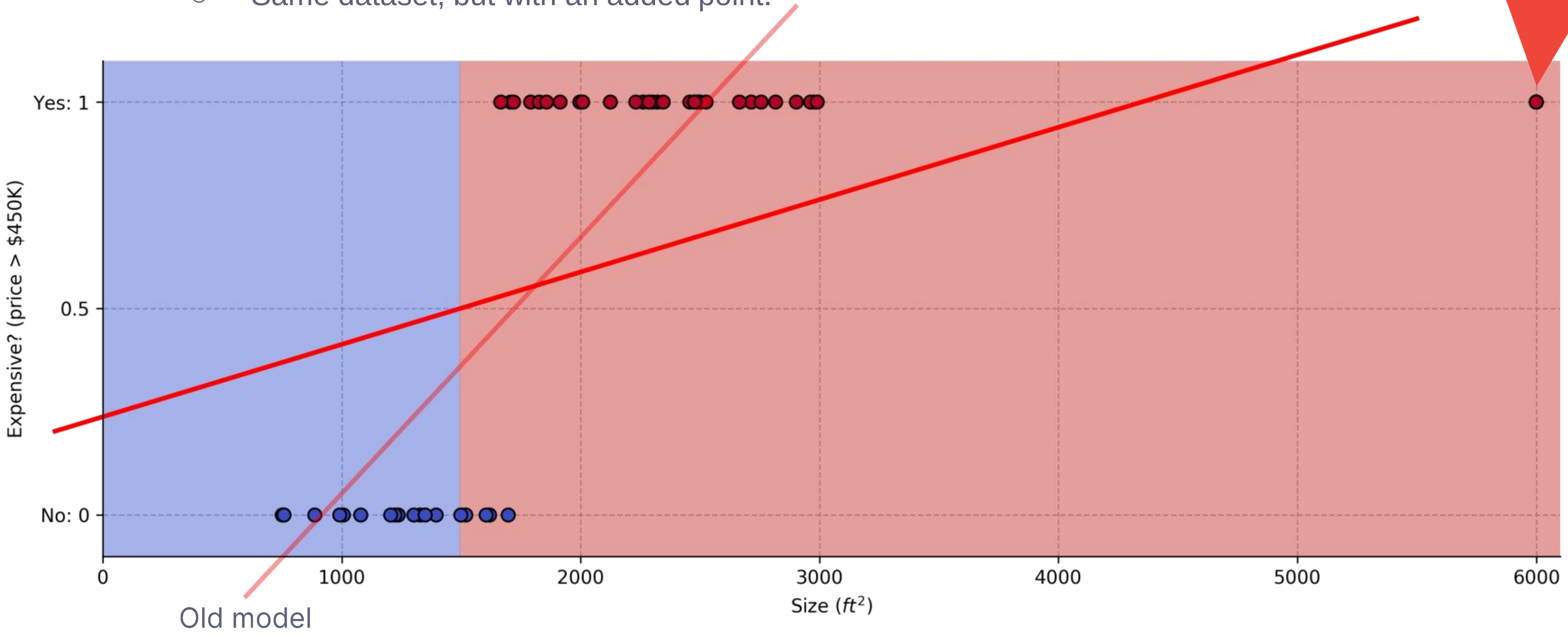
- Set of *expensive / non-expensive* house observations.
  - Same dataset, but with an added point.



Classification vs. Regression

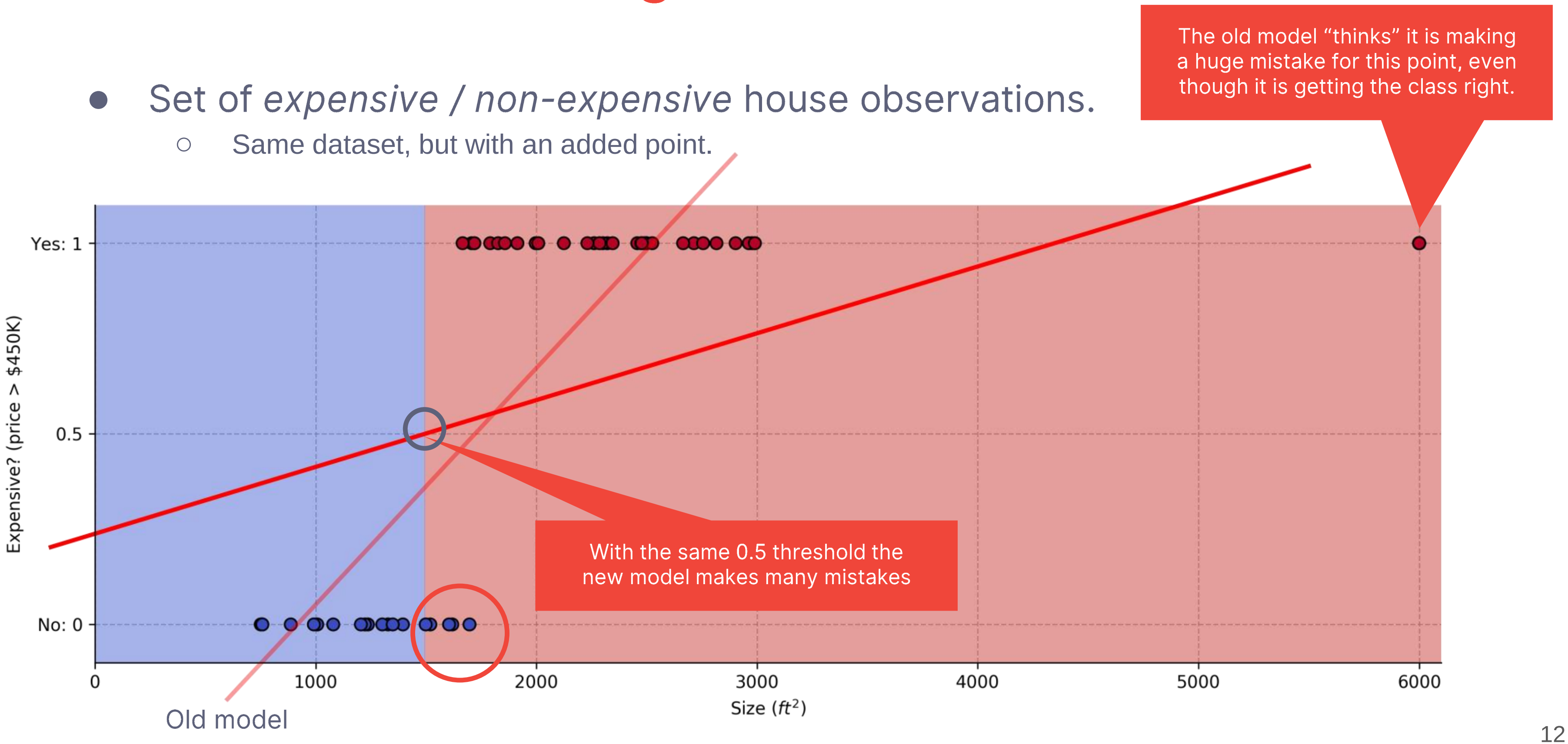
- Set of *expensive / non-expensive* house observations.
    - Same dataset, but with an added point.

The old model "thinks" it is making a huge mistake for this point, even though it is getting the class right.



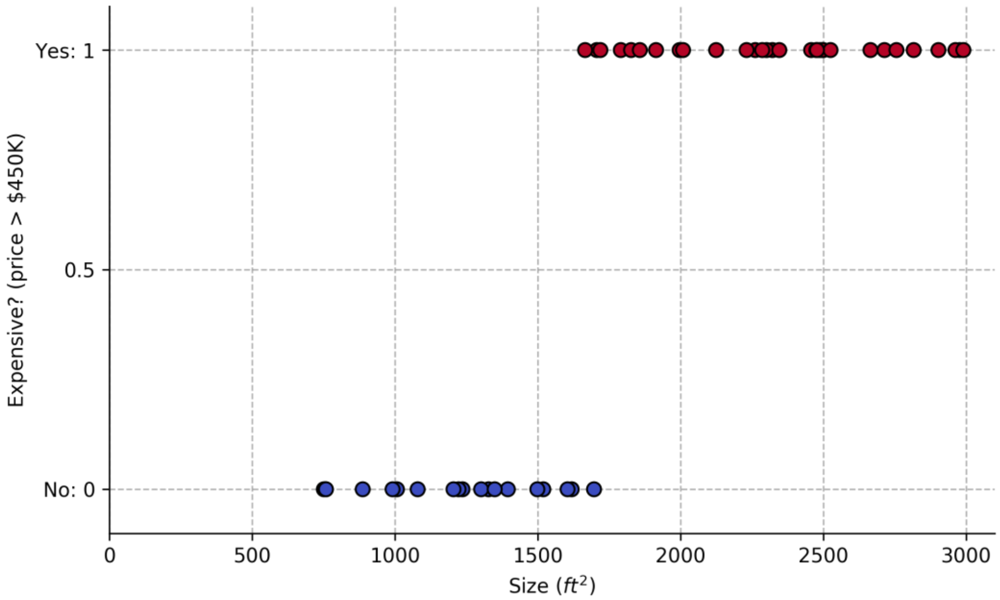
Classification vs. Regression

- Set of *expensive / non-expensive* house observations.
    - Same dataset, but with an added point.



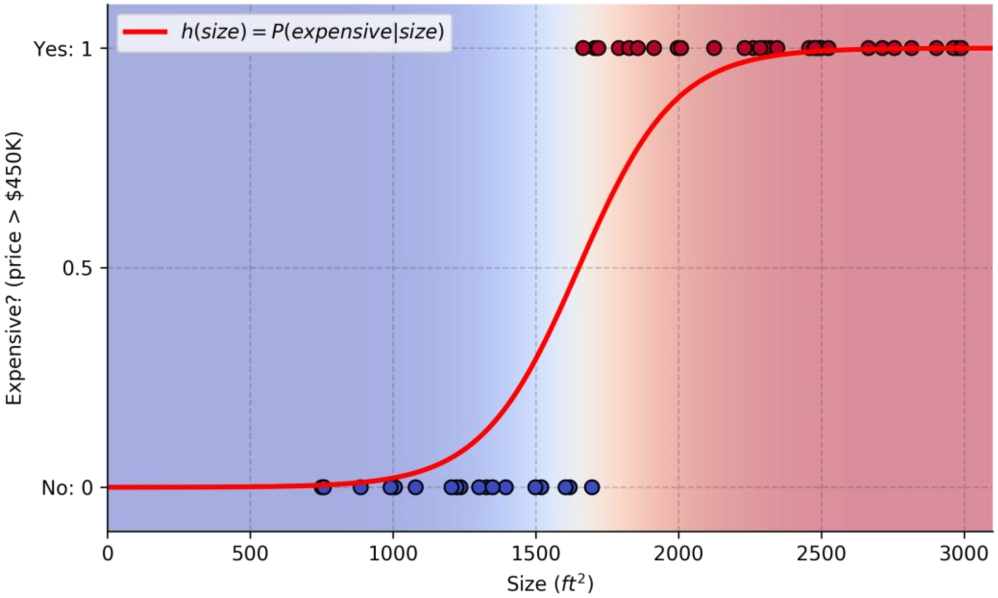
Predicting probabilities

- Set of *expensive / non-expensive* house observations.
  - We want to learn a model *h*, which gives us the **probability** of a house being expensive, given its size.



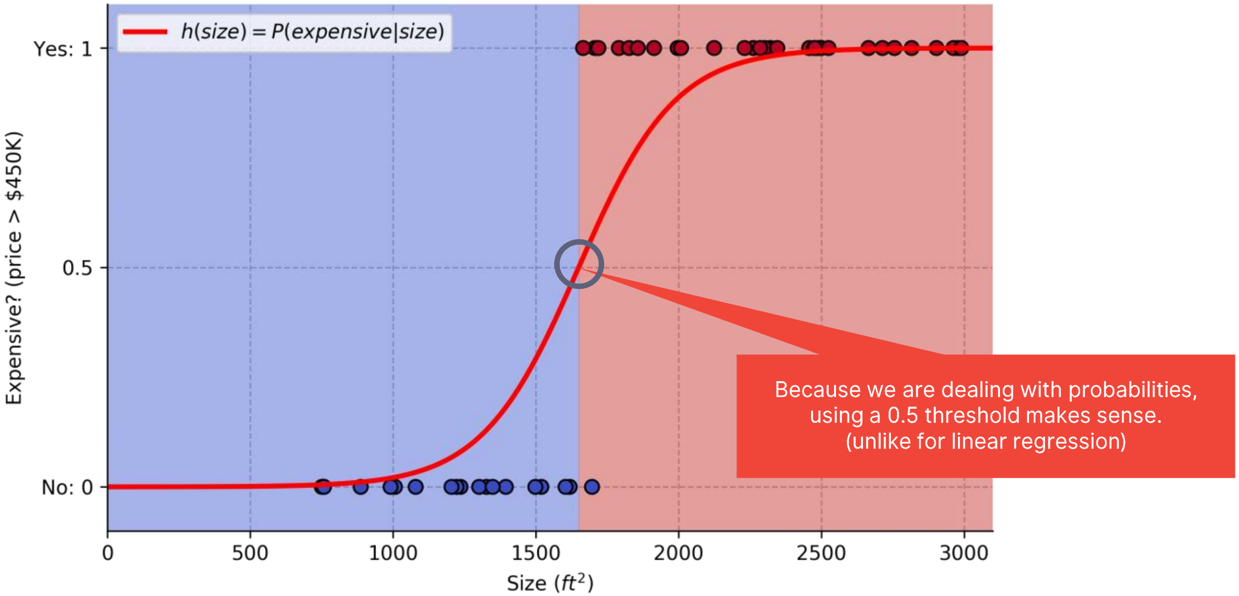
Predicting probabilities

- Set of *expensive / non-expensive* house observations.
    - We want to learn a model *h*, which gives us the **probability** of a house being expensive, given its size.



Predicting probabilities

- Set of *expensive / non-expensive* house observations.
  - We want to learn a model *h*, which gives us the **probability** of a house being expensive, given its size.



Logistic Function

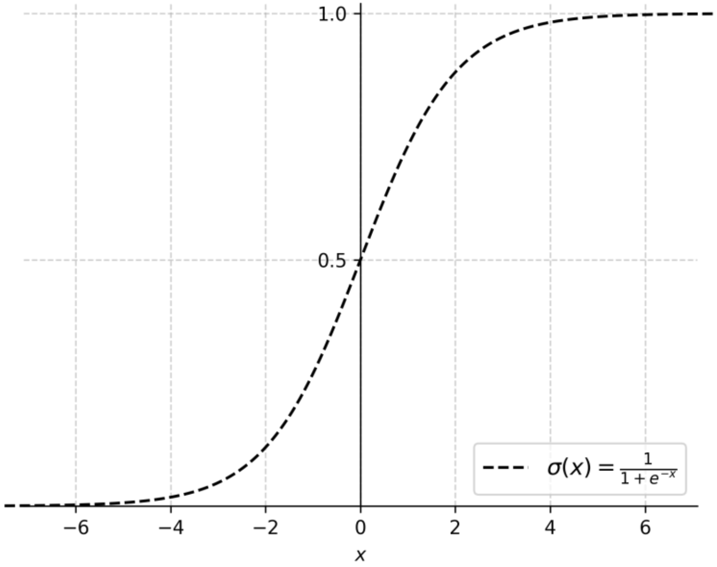
- Sigmoid function ("S-shaped" curve):
  - Family of functions which are bounded, differentiable, real and with a non-negative derivative at each point.
- Logistic function (special case of sigmoid):

$$\sigma(x) = \frac{L}{1 + e^{-k(x-x_0)}}$$

- **Standard logistic function** $(\mathrm{L} = 1, \mathrm{k} = 1, \mathrm{x}_0 = 0)$:

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

  - $\sigma(-x) = 1 - \sigma(x) = \frac{1}{1+e^{x}}$
  - $0 < \sigma(x) < 1 \Rightarrow$ Can be interpreted as **probability**



Logistic Model

- The relationship is modeled with a *logistic model*:
  - $\vec{x} \in \mathbb{R}^n$ represents the independent variables
  - $y \in \{0, 1\}$ is the dependent variable

$$\hat{y} = \sigma(w_0 + w_1 x_1 + \cdots + w_n x_n)$$

Logistic Model

- The relationship is modeled with a *logistic model*:
  - $\vec{x} \in \mathbb{R}^n$ represents the independent variables
  - $y \in \{0, 1\}$ is the dependent variable

$$\hat{y} = \sigma(w_0 + w_1 x_1 + \cdots + w_n x_n) = \frac{1}{1 + e^{-\langle \vec{w}, \vec{x} \rangle}}$$

Same trick as before: $x_0 = 1$

  - In other words, we apply a logistic function on top of a linear model.

Logistic Model

- The relationship is modeled with a *logistic model*:
  - $\vec{x} \in \mathbb{R}^n$ represents the independent variables
  - $y \in \{0, 1\}$ is the dependent variable

$$\hat{y} = \sigma(w_0 + w_1 x_1 + \cdots + w_n x_n) = \frac{1}{1 + e^{-\langle \vec{w}, \vec{x} \rangle}}$$

Same trick as before: $x_0 = 1$

  - In other words, we apply a logistic function on top of a linear model.

- Prediction can be interpreted as probability:

$$\hat{y} = P(y = 1 | \vec{x}, \vec{w})$$

Loss function

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\}, \vec{x}^{(i)} \in \mathbb{R}^n, y^{(i)} \in \{0,1\}$$

Loss function

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \ldots, (\vec{x}^{(m)}, y^{(m)})\}, \vec{x}^{(i)} \in \mathbb{R}^n, y^{(i)} \in \{0,1\}$$

- We could use *squared-error loss* like in linear regression.

$$\mathcal{L}_{\mathrm{E}} = \sum_i \left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \sum_i \left(y^{(i)} - \frac{1}{1 + e^{-\langle \vec{w}, \vec{x}^{(i)} \rangle}}\right)^2$$

Loss function

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \ldots, (\vec{x}^{(m)}, y^{(m)})\}, \vec{x}^{(i)} \in \mathbb{R}^n, y^{(i)} \in \{0,1\}$$

- We could use *squared-error loss* like in linear regression.

$$\mathcal{L}_{\mathrm{E}} = \sum_i \left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \sum_i \left(y^{(i)} - \frac{1}{1 + e^{-\langle \vec{w}, \vec{x}^{(i)} \rangle}}\right)^2$$

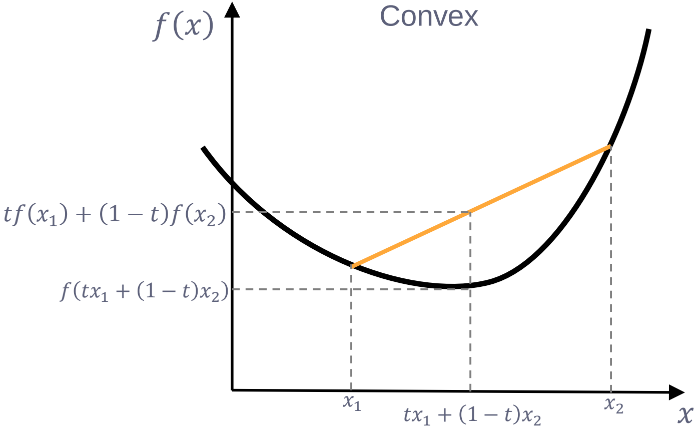
  - But this turns out to be a **non-convex function**

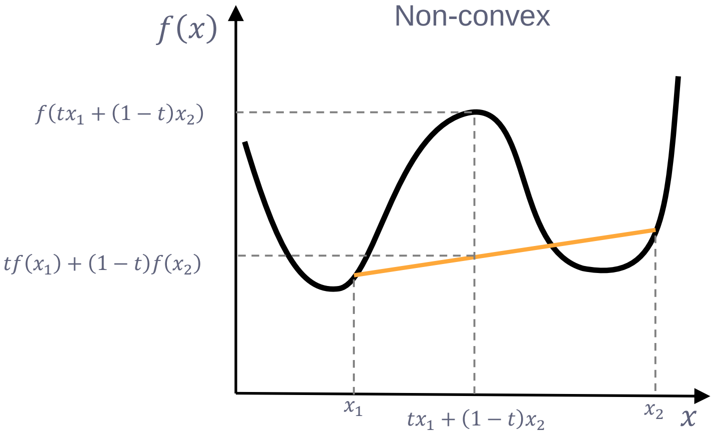
Convex vs. Non-convex

- A function is **convex** if a *segment between* any two points on its graph lies *above* the graph.

Convex vs. Non-convex

- A function is **convex** if a *segment between* any two points on its graph lies *above* the graph.

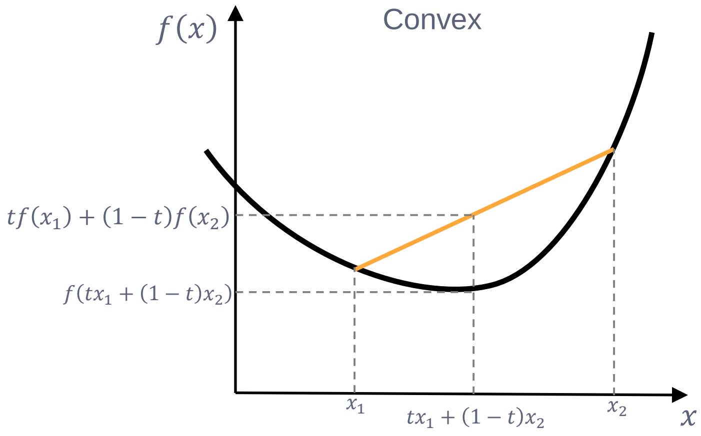


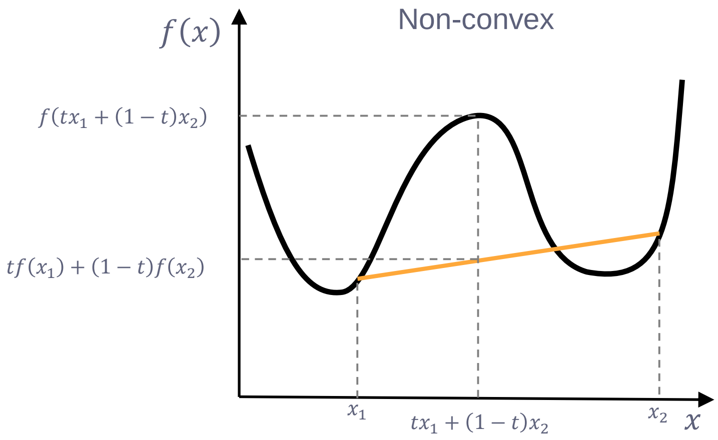


- $f: X \to Y$ is convex if $\boldsymbol{f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)} \quad \forall \boldsymbol{x_1, x_2} \in \boldsymbol{X}, \forall \boldsymbol{t} \in [\boldsymbol{0}, \boldsymbol{1}]$

Convex vs. Non-convex

- A function is **convex** if a *segment between* any two points on its graph lies *above* the graph.





- $f: X \to Y$ is convex if $\boldsymbol{f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)} \quad \forall \boldsymbol{x_1, x_2 \in X}, \forall \boldsymbol{t \in [0,1]}$
- A convex function has *only one local minimum* → We want error functions to be convex.

Loss function

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\}, \vec{x}^{(i)} \in \mathbb{R}^n, y^{(i)} \in \{0,1\}$$

- We could use *squared-error loss* like in linear regression:

$$\mathcal{L}_{\mathrm{E}} = \sum_i \left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \sum_i \left(y^{(i)} - \frac{1}{1 + e^{-\langle \vec{w}, \vec{x}^{(i)} \rangle}}\right)^2$$

  - But this turns out to be a **non-convex function** $\Longrightarrow$ multiple local minima

Loss function

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\}, \vec{x}^{(i)} \in \mathbb{R}^n, y^{(i)} \in \{0,1\}$$

- We could use *squared-error loss* like in linear regression:

$$\mathcal{L}_{\mathrm{E}} = \sum_i \left(y^{(i)} - \hat{y}^{(i)}\right)^2 = \sum_i \left(y^{(i)} - \frac{1}{1 + e^{-\langle \vec{w}, \vec{x}^{(i)} \rangle}}\right)^2$$

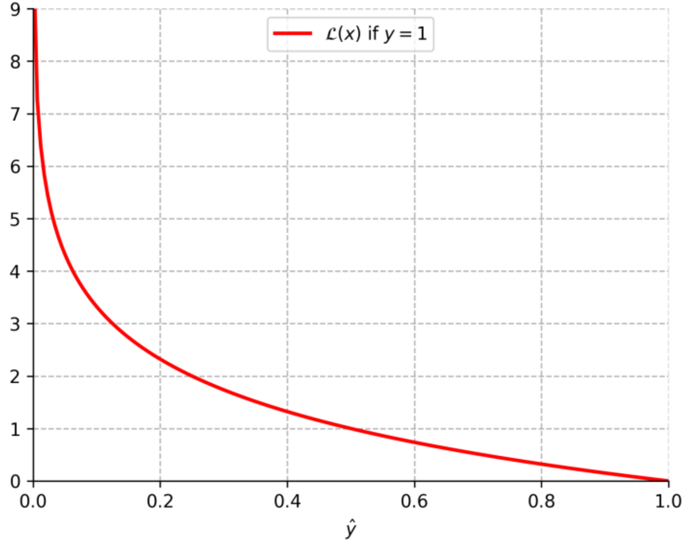
  - But this turns out to be a **non-convex function** $\Longrightarrow$ multiple local minima

- Using the so-called **Cross-entropy** or **Log-loss** works better for logistic regression.

Loss function

- Cross-entropy for a single example:

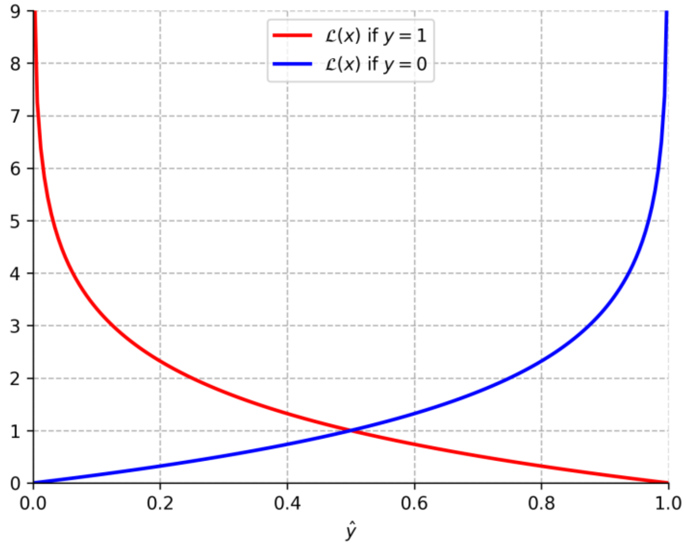
$$\mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right) = \begin{cases} -\log \hat{y}^{(i)} & \text{if } y^{(i)} = 1 \\ & \end{cases}$$



Loss function

- Cross-entropy for a single example:

$$\mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right) = \begin{cases} -\log \hat{y}^{(i)} & \text{if } y^{(i)} = 1 \\ -\log\left(1 - \hat{y}^{(i)}\right) & \text{if } y^{(i)} = 0 \end{cases}$$



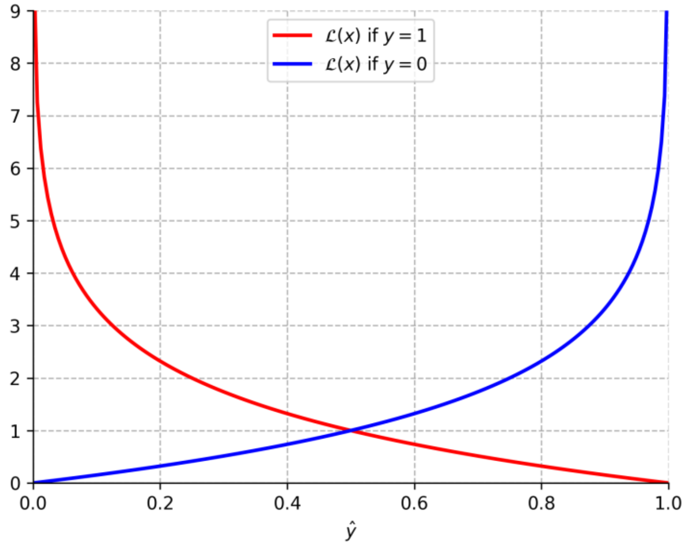
Loss function

- Cross-entropy for a single example:

$$\mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right) = \begin{cases} -\log \hat{y}^{(i)} & \text{if } y^{(i)} = 1 \\ -\log\left(1 - \hat{y}^{(i)}\right) & \text{if } y^{(i)} = 0 \end{cases}$$

  - $\mathcal{L} \to 0$ as $\hat{y}^{(i)} \to y^{(i)}$
  - $\mathcal{L} \to \infty$ as $\hat{y}^{(i)} \to 1 - y^{(i)}$

Loss grows exponentially if model is very confident in the wrong prediction.



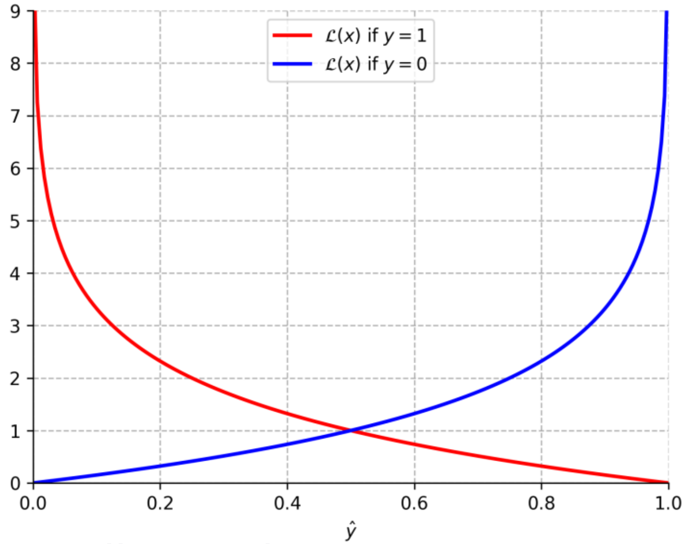
Loss function

- Cross-entropy for a single example:

$$\mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right) = \begin{cases} -\log \hat{y}^{(i)} & \text{if } y^{(i)} = 1 \\ -\log\left(1 - \hat{y}^{(i)}\right) & \text{if } y^{(i)} = 0 \end{cases}$$

  - $\mathcal{L} \to 0$ as $\hat{y}^{(i)} \to y^{(i)}$
  - $\mathcal{L} \to \infty$ as $\hat{y}^{(i)} \to 1 - y^{(i)}$

Loss grows exponentially if model is very confident in the wrong prediction.



- Combining both branches and summing over all samples:

$$\mathcal{L}_{\mathrm{E}} = -\sum_{i} \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right]$$

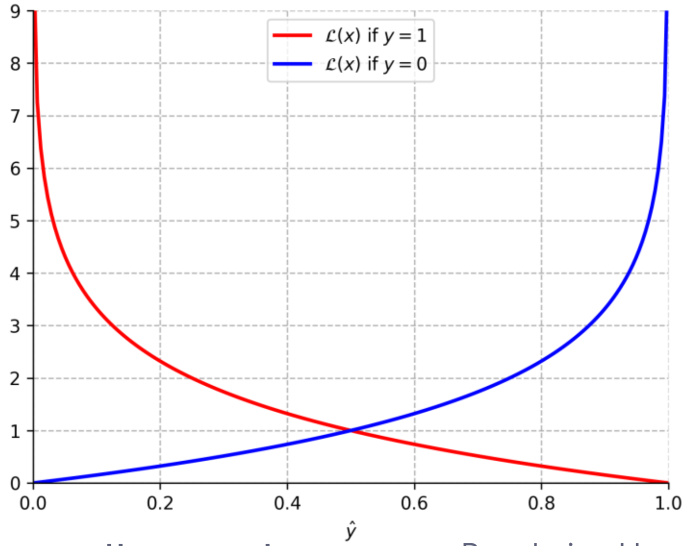
Loss function

- Cross-entropy for a single example:

$$\mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right) = \begin{cases} -\log \hat{y}^{(i)} & \text{if } y^{(i)} = 1 \\ -\log\left(1 - \hat{y}^{(i)}\right) & \text{if } y^{(i)} = 0 \end{cases}$$

  - $\mathcal{L} \to 0$ as $\hat{y}^{(i)} \to y^{(i)}$
  - $\mathcal{L} \to \infty$ as $\hat{y}^{(i)} \to 1 - y^{(i)}$

Loss grows exponentially if model is very confident in the wrong prediction.



- Combining both branches and summing over all samples:

Regularized Logistic Regression

$$\mathcal{L}_{\mathrm{E}} = -\sum_{i}\left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right)\right] + \lambda \|\vec{w}\|_2^2$$

Optimization

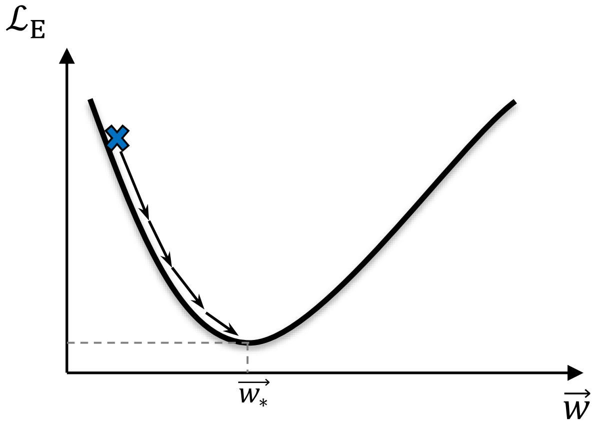
- We need to find $\overrightarrow{w_*} = \underset{\vec{w}}{\operatorname{argmin}}\, \mathcal{L}_{\mathrm{E}}(\vec{w})$

Optimization

- We need to find $\vec{w}_* = \underset{\vec{w}}{\operatorname{argmin}} \mathcal{L}_{\mathrm{E}}(\vec{w})$
- There is no *closed-form solution* for finding the minimum of the log-loss.
  - i.e. We cannot compute it directly through mathematical operations, like for linear regression
  - We need to use an *optimization method.*

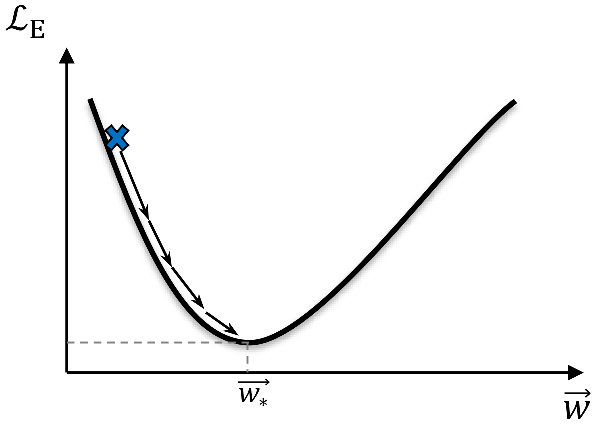
Optimization

- We need to find $\vec{w}_* = \underset{\vec{w}}{\operatorname{argmin}} \mathcal{L}_{\mathrm{E}}(\vec{w})$
- There is no *closed-form solution* for finding the minimum of the log-loss.
  - i.e. We cannot compute it directly through mathematical operations, like for linear regression
  - We need to use an *optimization method.*
- Gradient Descent
  - Gradually go down the slope of the error function



Optimization

- We need to find $\vec{w}_* = \underset{\vec{w}}{\operatorname{argmin}} \mathcal{L}_{\mathrm{E}}(\vec{w})$
- There is no *closed-form solution* for finding the minimum of the log-loss.
  - i.e. We cannot compute it directly through mathematical operations, like for linear regression
  - We need to use an *optimization method.*
- Gradient Descent
  - Gradually go down the slope of the error function
- More complex methods:
  - Conjugate Gradient
  - BFGS
  - L-BFGS



Optimization – C vs. λ

$$\underset{\vec{w}}{\operatorname{argmin}}\, \mathcal{L}_{\mathrm{E}}(\vec{w}) =$$

$$\underset{\vec{w}}{\operatorname{argmin}} \left\{ -\sum_i \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right] + \lambda \|\vec{w}\|_2^2 \right\}$$

Optimization – C vs. λ

$$\underset{\vec{w}}{\operatorname{argmin}}\, \mathcal{L}_{\mathrm{E}}(\vec{w}) =$$

$$\underset{\vec{w}}{\operatorname{argmin}} \left\{ -\sum_i \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right] + \lambda \|\vec{w}\|_2^2 \right\} =$$

$$\underset{\vec{w}}{\operatorname{argmin}} \left\{ -\frac{1}{\lambda} \sum_i \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right] + \|\vec{w}\|_2^2 \right\}$$

Dividing an expression by a constant does not change the *argmin*.

Optimization – C vs. λ

$$\underset{\vec{w}}{\operatorname{argmin}}\, \mathcal{L}_{\mathrm{E}}(\vec{w}) =$$

$$\underset{\vec{w}}{\operatorname{argmin}} \left\{ -\sum_i \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right] + \lambda \|\vec{w}\|_2^2 \right\} =$$

Dividing an expression by a constant does not change the *argmin*.

$$\underset{\vec{w}}{\operatorname{argmin}} \left\{ -\frac{1}{\lambda} \sum_i \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right] + \|\vec{w}\|_2^2 \right\} =$$

$${}^{1}/_{\lambda} \overset{\text{not}}{=} C$$

$$\underset{\vec{w}}{\operatorname{argmin}} \left\{ -C \sum_i \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right] + \|\vec{w}\|_2^2 \right\}$$

Optimization – C vs. λ

$$\underset{\vec{w}}{\operatorname{argmin}}\, \mathcal{L}_{\mathrm{E}}(\vec{w}) =$$

$$\underset{\vec{w}}{\operatorname{argmin}} \left\{ -\sum_i \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right] + \lambda \|\vec{w}\|_2^2 \right\} =$$

Dividing an expression by a constant does not change the *argmin*.

$$\underset{\vec{w}}{\operatorname{argmin}} \left\{ -\frac{1}{\lambda} \sum_i \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right] + \|\vec{w}\|_2^2 \right\} =$$

$$1/\lambda \overset{\text{not}}{=} C$$

$$\underset{\vec{w}}{\operatorname{argmin}} \left\{ -C \sum_i \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right] + \|\vec{w}\|_2^2 \right\}$$

- More common formulation of regularization for classification problems, because C is easier to interpret (*"cost of making a training mistake"*).

Different Formulation for Loss Function

Different Formulation

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(\mathrm{m})}, y^{(m)})\}, \vec{x}^{(\mathrm{i})} \in \mathbb{R}^n, \mathrm{y}^{(i)} \in \{-1,1\}$$

We can denote the two classes of binary classification with ± 1

Different Formulation

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(\mathrm{m})}, y^{(m)})\}, \vec{x}^{(\mathrm{i})} \in \mathbb{R}^n, \mathrm{y}^{(i)} \in \{-1,1\}$$

$$P(y = 1|\vec{x}, \vec{w}) = \sigma(\langle\vec{w}, \vec{x}\rangle) = \frac{1}{1 + e^{-\langle\vec{w},\vec{x}\rangle}}$$

We can denote the two classes of binary classification with ± 1

Different Formulation

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(\mathrm{m})}, y^{(m)})\}, \vec{x}^{(\mathrm{i})} \in \mathbb{R}^n, \mathrm{y}^{(i)} \in \{-1,1\}$$

$$P(y = 1|\vec{x}, \vec{w}) = \sigma(\langle \vec{w}, \vec{x} \rangle) = \frac{1}{1 + e^{-\langle \vec{w}, \vec{x} \rangle}}$$

$$P(y = -1|\vec{x}, \vec{w}) = 1 - \sigma(\langle \vec{w}, \vec{x} \rangle) = \frac{1}{1 + e^{\langle \vec{w}, \vec{x} \rangle}}$$

We can denote the two classes of binary classification with ± 1

Different Formulation

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(\mathrm{m})}, y^{(m)})\}, \vec{x}^{(\mathrm{i})} \in \mathbb{R}^n, \mathrm{y}^{(i)} \in \{-1,1\}$$

$$\left.\begin{aligned} P(y = 1|\vec{x}, \vec{w}) &= \frac{1}{1 + e^{-\langle \vec{w}, \vec{x} \rangle}} \\ P(y = -1|\vec{x}, \vec{w}) &= \frac{1}{1 + e^{\langle \vec{w}, \vec{x} \rangle}} \end{aligned}\right\} \quad P\left(y^{(i)} \middle| \vec{x}^{(\mathrm{i})}, \vec{w}\right) = \frac{1}{1 + e^{-\mathrm{y}^{(i)} \langle \vec{w}, \vec{x}^{(\mathrm{i})} \rangle}}$$

We can denote the two classes of binary classification with ± 1

Different Formulation

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \ldots, (\vec{x}^{(\mathrm{m})}, y^{(m)})\}, \vec{x}^{(\mathrm{i})} \in \mathbb{R}^n, \mathrm{y}^{(i)} \in \{-1,1\}$$

We can denote the two classes of binary classification with ± 1

$$\left.\begin{aligned} P(y = 1|\vec{x}, \vec{w}) &= \frac{1}{1 + e^{-\langle \vec{w}, \vec{x} \rangle}} \\ P(y = -1|\vec{x}, \vec{w}) &= \frac{1}{1 + e^{\langle \vec{w}, \vec{x} \rangle}} \end{aligned}\right\} P\left(y^{(i)} \middle| \vec{x}^{(\mathrm{i})}, \vec{w}\right) = \frac{1}{1 + e^{-\mathrm{y}^{(i)}\langle \vec{w}, \vec{x}^{(\mathrm{i})} \rangle}}$$

- Likelihood of data: $$\prod_{i=1}^{m} P\left(y^{(i)}\right) = \prod_{i=1}^{m} \frac{1}{1 + e^{-\mathrm{y}^{(i)}\langle \vec{w}, \vec{x}^{(\mathrm{i})} \rangle}}$$

Different Formulation

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(\mathrm{m})}, y^{(m)})\}, \vec{x}^{(\mathrm{i})} \in \mathbb{R}^n, \mathrm{y}^{(i)} \in \{-1,1\}$$

We can denote the two classes of binary classification with ± 1

$$\left.\begin{aligned} P(y = 1|\vec{x}, \vec{w}) &= \frac{1}{1 + e^{-\langle \vec{w}, \vec{x} \rangle}} \\ P(y = -1|\vec{x}, \vec{w}) &= \frac{1}{1 + e^{\langle \vec{w}, \vec{x} \rangle}} \end{aligned}\right\} P\left(y^{(i)} \middle| \vec{x}^{(\mathrm{i})}, \vec{w}\right) = \frac{1}{1 + e^{-\mathrm{y}^{(i)} \langle \vec{w}, \vec{x}^{(\mathrm{i})} \rangle}}$$

- Log-Likelihood of data: $\log\left(\prod_{i=1}^{m} \frac{1}{1 + e^{-\mathrm{y}^{(i)} \langle \vec{w}, \vec{x}^{(\mathrm{i})} \rangle}}\right) = -\sum_{i=1}^{m} \log\left(1 + e^{-\mathrm{y}^{(i)} \langle \vec{w}, \vec{x}^{(\mathrm{i})} \rangle}\right)$

Different Formulation

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(\mathrm{m})}, y^{(m)})\}, \vec{x}^{(\mathrm{i})} \in \mathbb{R}^n, \mathrm{y}^{(i)} \in \{-1,1\}$$

We can denote the two classes of binary classification with ± 1

$$\left.\begin{aligned} P(y = 1|\vec{x}, \vec{w}) &= \frac{1}{1 + e^{-\langle \vec{w}, \vec{x} \rangle}} \\ P(y = -1|\vec{x}, \vec{w}) &= \frac{1}{1 + e^{\langle \vec{w}, \vec{x} \rangle}} \end{aligned}\right\} \quad P\left(y^{(i)} \middle| \vec{x}^{(\mathrm{i})}, \vec{w}\right) = \frac{1}{1 + e^{-\mathrm{y}^{(i)} \langle \vec{w}, \vec{x}^{(\mathrm{i})} \rangle}}$$

- Log-Likelihood of data: $\log\left(\prod_{i=1}^{m} \frac{1}{1 + e^{-\mathrm{y}^{(i)} \langle \vec{w}, \vec{x}^{(\mathrm{i})} \rangle}}\right) = -\sum_{i=1}^{m} \log\left(1 + e^{-\mathrm{y}^{(i)} \langle \vec{w}, \vec{x}^{(\mathrm{i})} \rangle}\right)$
- Maximize the log-likelihood of data $\Longrightarrow$

$$\textbf{minimize} \sum_{i=1}^{m} \log\left(1 + e^{-y^{(i)} \langle \vec{w}, \vec{x}^{(i)} \rangle}\right)$$

Different Formulation

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\}, \vec{x}^{(i)} \in \mathbb{R}^n, y^{(i)} \in \{-1,1\}$$

We can denote the two classes of binary classification with ± 1

$$\left.\begin{aligned} P(y=1|\vec{x},\vec{w}) &= \frac{1}{1+e^{-\langle \vec{w},\vec{x}\rangle}} \\ P(y=-1|\vec{x},\vec{w}) &= \frac{1}{1+e^{\langle \vec{w},\vec{x}\rangle}} \end{aligned}\right\} \quad P\left(y^{(i)} \middle| \vec{x}^{(i)}, \vec{w}\right) = \frac{1}{1+e^{-y^{(i)}\langle \vec{w},\vec{x}^{(i)}\rangle}}$$

- Log-Likelihood of data: $\log\left(\prod_{i=1}^{m} \frac{1}{1+e^{-y^{(i)}\langle \vec{w},\vec{x}^{(i)}\rangle}}\right) = -\sum_{i=1}^{m} \log\left(1+e^{-y^{(i)}\langle \vec{w},\vec{x}^{(i)}\rangle}\right)$
- Maximize the log-likelihood of data $\Longrightarrow$

$$\textbf{minimize } C\sum_{i=1}^{m} \log\left(1+e^{-y^{(i)}\langle \vec{w},\vec{x}^{(i)}\rangle}\right) + \|\vec{w}\|_2^2$$

With regularization

Different Formulation

$$E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(\mathrm{m})}, y^{(m)})\}, \vec{x}^{(\mathrm{i})} \in \mathbb{R}^n, \mathrm{y}^{(i)} \in \{-1,1\}$$

We can denote the two classes of binary classification with ± 1

$$\left.\begin{aligned} P(y=1|\vec{x},\vec{w}) &= \frac{1}{1+e^{-\langle\vec{w},\vec{x}\rangle}} \\ P(y=-1|\vec{x},\vec{w}) &= \frac{1}{1+e^{\langle\vec{w},\vec{x}\rangle}} \end{aligned}\right\} \quad P\left(y^{(i)}\middle|\vec{x}^{(\mathrm{i})},\vec{w}\right) = \frac{1}{1+e^{-\mathrm{y}^{(i)}\langle\vec{w},\vec{x}^{(\mathrm{i})}\rangle}}$$

- Log-Likelihood of data: $\log\left(\prod_{i=1}^{m}\frac{1}{1+e^{-\mathrm{y}^{(i)}\langle\vec{w},\vec{x}^{(\mathrm{i})}\rangle}}\right) = -\sum_{i=1}^{m}\log\left(1+e^{-\mathrm{y}^{(i)}\langle\vec{w},\vec{x}^{(\mathrm{i})}\rangle}\right)$
- Maximize the log-likelihood of data $\Longrightarrow$

$$\textbf{minimize}\ \ C\sum_{i=1}^{m}\log\left(1+e^{-y^{(i)}\langle\vec{w},\vec{x}^{(i)}\rangle}\right) + \|\vec{w}\|_2^2$$

Formulation used by *Scikit-learn.*

With regularization

Recap

- **Logistic Regression** uses a *logistic function* on top of a *linear model* to establish a relationship between a ***binary*** *dependent variable* and a number of *independent variables*.

$$\hat{y} = \sigma(\langle \vec{w}, \vec{x} \rangle) = \frac{1}{1 + e^{-\langle \vec{w}, \vec{x} \rangle}}$$

- Parameters are obtained by minimizing the **Cross-entropy loss:**

$$-C \sum_{i} \left[y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \right] + \|\vec{w}\|_2^2$$

or by minimizing the negative **log-likelihood** of the data:

$$C \sum_{i=1}^{m} \log\left(1 + e^{-y^{(i)} \langle \vec{w}, \vec{x}^{(i)} \rangle}\right) + \|\vec{w}\|_2^2$$

Considering labels to be ± 1

Multinomial Logistic Regression

What if we have multiple classes?

- We want to know *how likely it is* that a certain dog is of one of four breeds.

Weight = *25 kg*

Height = *46 cm*

Fur length = *0*

Ear length = *3 cm*

Color = *Cream*

German Sheppard

Labrador Retriever

Shar Pei

Collie

What if we have multiple classes?

- We want to know *how likely it is* that a certain dog is of one of four breeds.

Weight = *25 kg*

Height = *46 cm*

Fur length = *0*

Ear length = *3 cm*

Color = *Cream*

Multinomial Logistic Regression

10%

German Sheppard

15%

Labrador Retriever

70%

Shar Pei

5%

Collie

What if we have multiple classes?

- We want to know *how likely it is* that a certain dog is of one of four breeds.

Weight = *25 kg*

Height = *46 cm*

Fur length = *0*

Ear length = *3 cm*

Color = *Cream*

Multinomial Logistic Regression

10%

German Sheppard

15%

Labrador Retriever

70%

Shar Pei

5%

Collie

Predictions should sum up to 100%.

The predicted class is the one with the highest probability.

What if we have multiple classes?

- If we have K classes, we predict a probability for each class:
  - The prediction $\hat{y}^{(i)}$ becomes array of probabilities.
  - ${\hat{y}_k}^{(i)}$ is the probability of $\vec{x}^{(i)}$ being class k (e.g. $\hat{y}^{(i)} = [0.1 \quad 0.15 \quad 0.7 \quad 0.05]$)

What if we have multiple classes?

- If we have K classes, we predict a probability for each class:
  - The prediction $\hat{y}^{(i)}$ becomes array of probabilities.
  - $\hat{y}_k{}^{(i)}$ is the probability of $\vec{x}^{(i)}$ being class k (e.g. $\hat{y}^{(i)} = [0.1 \quad 0.15 \quad 0.7 \quad 0.05]$)

The predicted class is the one with the highest probability.

What if we have multiple classes?

- If we have K classes, we predict a probability for each class:
  - The prediction $\hat{y}^{(i)}$ becomes array of probabilities.
  - ${\hat{y}_k}^{(i)}$ is the probability of $\vec{x}^{(i)}$ being class k (e.g. $\hat{y}^{(i)} = [0.1 \quad 0.15 \quad 0.7 \quad 0.05]$)
  - Training samples: $y_k^{(i)} = \begin{cases} 1 & \text{if } \vec{x}^{(i)} \text{ is class } k \\ 0 & \text{otherwise} \end{cases}$ (e.g. $y^{(i)} = [0 \quad 0 \quad 1 \quad 0]$)

The predicted class is the one with the highest probability.

What if we have multiple classes?

- If we have K classes, we predict a probability for each class:
  - The prediction $\hat{y}^{(i)}$ becomes array of probabilities.
  - ${\hat{y}_k}^{(i)}$ is the probability of $\vec{x}^{(i)}$ being class k (e.g. $\hat{y}^{(i)} = [0.1 \quad 0.15 \quad 0.7 \quad 0.05]$)

  - Training samples: $y_k^{(i)} = \begin{cases} 1 & \text{if } \vec{x}^{(i)} \text{ is class } k \\ 0 & \text{otherwise} \end{cases}$ (e.g. $y^{(i)} = [0 \quad 0 \quad 1 \quad 0]$)

The predicted class is the one with the highest probability.

- **Multinomial Cross-Entropy**:

$$\mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right) = \sum_{k=1}^{K} y_k^{(i)} \log {\hat{y}_k}^{(i)}$$

What if we have multiple classes?

- If we have K classes, we predict a probability for each class:
  - The prediction $\hat{y}^{(i)}$ becomes array of probabilities.
  - $\hat{y}_k{}^{(i)}$ is the probability of $\vec{x}^{(i)}$ being class k (e.g. $\hat{y}^{(i)} = [0.1 \quad 0.15 \quad 0.7 \quad 0.05]$)
  - Training samples: $y_k^{(i)} = \begin{cases} 1 & \text{if } \vec{x}^{(i)} \text{ is class } k \\ 0 & \text{otherwise} \end{cases}$ (e.g. $y^{(i)} = [0 \quad 0 \quad 1 \quad 0]$)

The predicted class is the one with the highest probability.

- **Multinomial Cross-Entropy**:

$$\mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right) = \sum_{k=1}^{K} y_k^{(i)} \log \hat{y}_k{}^{(i)}$$

Only one term will be non-zero

What if we have multiple classes?

- We have a separate weight vector $\vec{w}_k$ for each class.
- Prediction is computed by applying **Softmax**:

$$\hat{y}_k = \frac{e^{\langle \vec{w}_k, \vec{x} \rangle}}{\sum_{j=1}^{K} e^{\langle \vec{w}_j, \vec{x} \rangle}}$$

Softmax is a generalization of the logistic function for n-dimensional inputs.

What if we have multiple classes?

- We have a separate weight vector $\vec{w}_k$ for each class.
- Prediction is computed by applying **Softmax**:

$$\hat{y}_k = \frac{e^{\langle \vec{w}_k, \vec{x} \rangle}}{\sum_{\mathrm{j}=1}^{\mathrm{K}} e^{\langle \vec{w}_j, \vec{x} \rangle}}$$

Softmax is a generalization of the logistic function for n-dimensional inputs.

- Even though we have a separate $\vec{w}_j$ for each class, there is only one loss which is minimized.
  - The weight vectors are obtained simultaneously.

Other strategies for multiclass classification

- Another method of having multiple classes is to fit multiple binary classifiers independently and combine their predictions.

Other strategies for multiclass classification

- Another method of having multiple classes is to fit multiple binary classifiers independently and combine their predictions.

- **One-versus-rest** (OVR), sometimes called (one-versus-all, OVA)
    - Train n classifiers, one for each class, where the negative examples are all the other classes.
    - At inference, run all classifiers and pick the class with the highest margin (most confident)

Other strategies for multiclass classification

- Another method of having multiple classes is to fit multiple binary classifiers independently and combine their predictions.

- **One-versus-rest** (OVR), sometimes called (one-versus-all, OVA)
  - Train n classifiers, one for each class, where the negative examples are all the other classes.
  - At inference, run all classifiers and pick the class with the highest margin (most confident)

- **One-versus-one** (OVO)
  - Train $n(n-1)/2$ classifiers, one for each pair of classes.
  - At inference, run all classifiers and pick the class which was selected by most of them

Logistic Regression in Python

```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(C = 10) # C is the inverse regularization strength 1/λ
clf.fit(X, y)

clf.predict([x]) # prediction for x
clf.predict_proba([x]) # predicted probability for x
clf.decision_function([x]) # value of the decision function before applying logistic: ⟨w⃗,x⃗⟩

clf.coef_ # w_1, w_2 ... w_n
clf.intercept_ # w_0

clf = LogisticRegression(multi_class = ‘multinomial’) # multi_class = {’multinomial’, ‘ovr’}
```

Conclusions

- **Logistic regression** uses a *logistic function* on top of a *linear function* to establish a relationship between a ***binary*** *dependent variable* and a number of *independent variables*.
- **Logistic Regression** is a method for **classification**.
    - Name is due to historical reasons and the relation to *linear regression*.
- The prediction can be interpreted as **probability.**
- The parameters of the model are obtained by minimizing the **cross-entropy loss** or *log-loss*.
    - Another possibility is to maximize the **log-likelihood** of the data
    - Since there is no *closed-form solution*, a convex optimization method is used.
- *Multinomial cross-entropy* can be used to achieve multiclass classification.
    - Other multiclass strategies are *OVR* and *OVO*.

History of Logistic Regression

- *"The regression analysis of binary sequences (with discussion)."*

  David Cox, 1958

- *"Estimation of the Probability of an Event as a Function of Several Independent Variables."*

  Strother H. Walker and David B. Duncan, 1967

Keywords

Logistic Regression
Sigmoid
Probability
Cross-entropy
Closed-form Solution
Log-likelihood
Convex Function
Optimization Method
Gradient Descent
Multinomial
Softmax
One-versus-rest (OVR)
One-versus-one (OVO)