# Neural Networks and Deep Learning

## Probability and Statistics for Data Science

Carlos Fernandez-Granda

These slides are based on the book Probability and Statistics for Data Science by Carlos Fernandez-Granda, available for purchase here. A free preprint, videos, code, slides and solutions to exercises are available at https://www.ps4ds.net

# Regression

**Goal:** Estimate response from features

**Optimal estimator:** Conditional mean

**Problem:** Intractable to compute due to curse of dimensionality

# Linear regression

Response $y$ is approximated as an linear (affine) function of the features $x$

$$y \approx \sum_{i=1}^{d} \beta[i] x[i] + \alpha$$

**Assumption:** Response increases or decreases proportionally to each feature (if we fix other features)

# Example

Response: Temperature in Manhattan (Kansas)

Features:

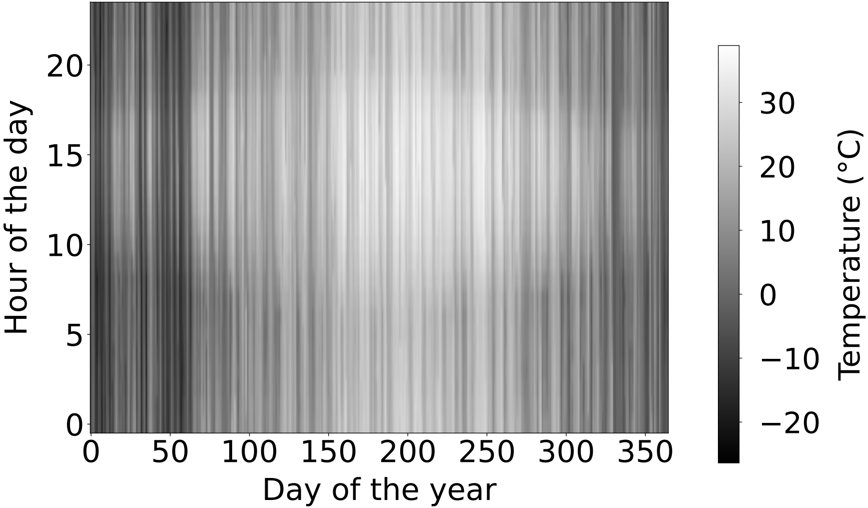
(1) Hour of the day (0-23)

(2) Day of the year (1-365)

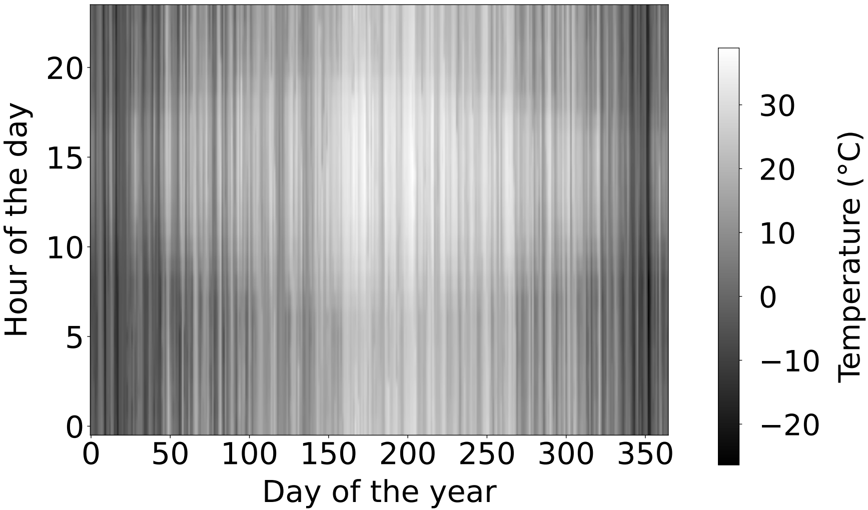
Training data: 2015

Test data: 2016

# Training data

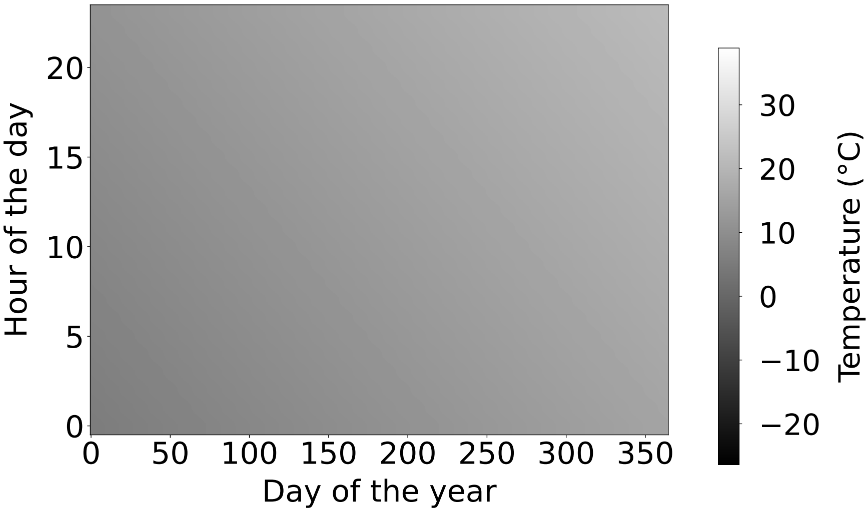


# Test data



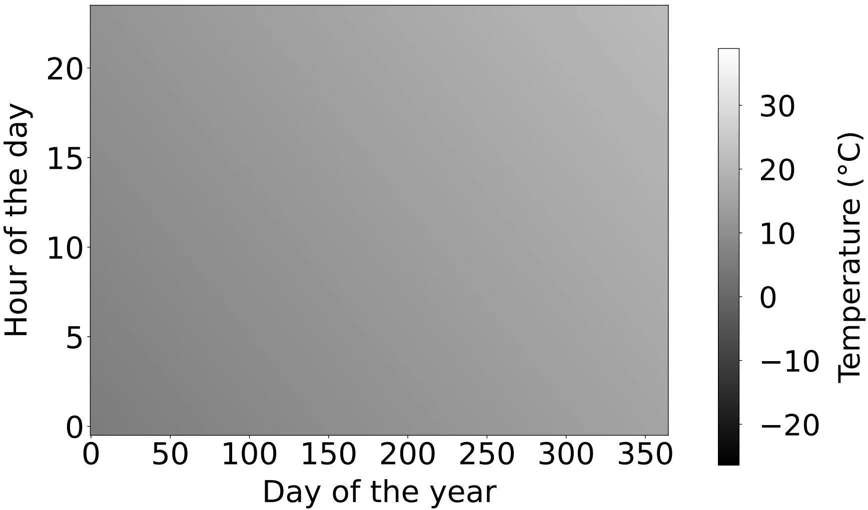
Hour of the day
20
15
10
5
0
0
50
100
150
200
250
300
350
Day of the year
30
20
10
0
−10
−20
Temperature (°C)

# Linear model: 0.25 hour + 0.03 day + 5.85




Response increases or decreases proportionally to each feature (if we fix other features)

# Linear model: $0.25$ hour + $0.03$ day + $5.85$




Training error: $10.8°C$ Test error: $11.0°C$

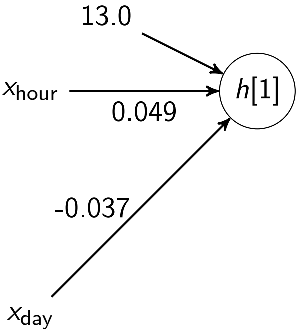
# Challenge

How to learn nonlinear model?

Neural network: Interleave

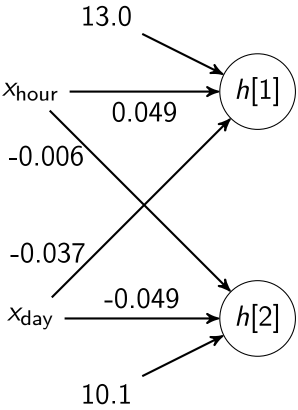
- Linear (affine) transformations
- Nonlinearity

# Hidden variables (a.k.a. feature or activation maps)



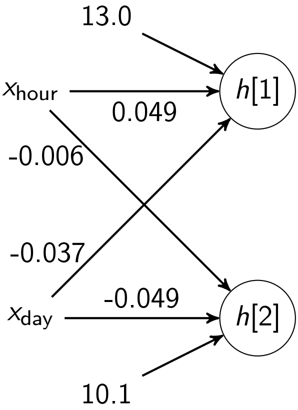

$$h[1] := 0.049x_{\text{hour}} - 0.037x_{\text{day}} + 13.0$$

# Hidden variables (a.k.a. feature or activation maps)




$$h[2] := -0.006x_{\text{hour}} - 0.049x_{\text{day}} + 10.1$$

# Hidden variables




$$h := \underbrace{\begin{bmatrix} 0.049 & -0.037 \\ -0.006 & -0.049 \end{bmatrix}}_{W_1} \begin{bmatrix} x_{\text{hour}} \\ x_{\text{day}} \end{bmatrix} + \underbrace{\begin{bmatrix} 13.0 \\ 10.1 \end{bmatrix}}_{\alpha_1}$$

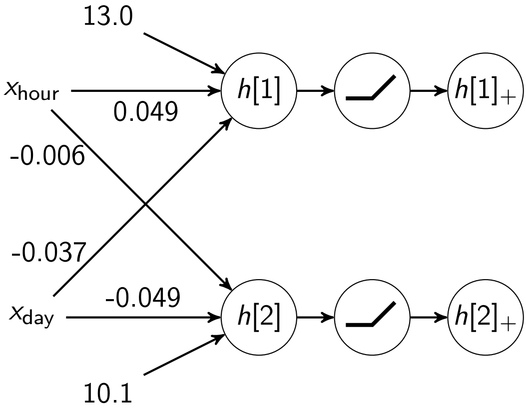
# Activation function

Nonlinearity applied to each hidden variable

Rectified Linear Unit (ReLU): Sets negative entries to zero

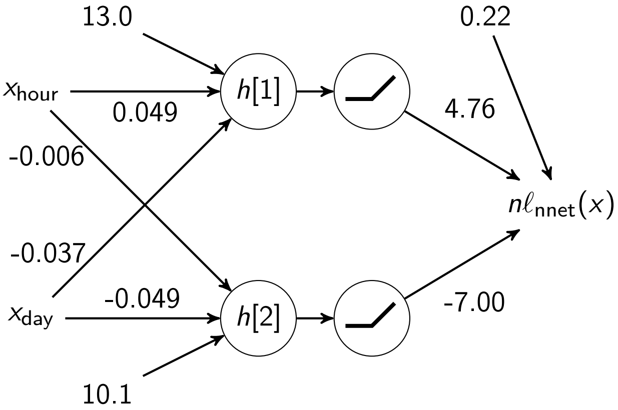
$$a_+ := \begin{cases} a & \text{if} \quad a \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

# ReLU



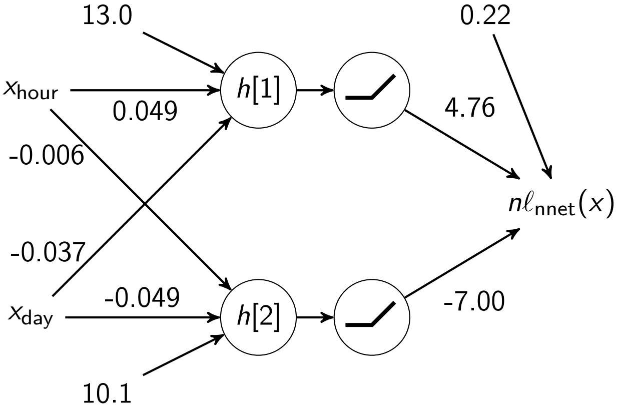

$$h_+ := \begin{bmatrix} h[1]_+ \\ h[2]_+ \end{bmatrix}$$

# Output layer




$$n\ell_{\text{nnet}}\left(\begin{bmatrix} x_{\text{hour}} \\ x_{\text{day}} \end{bmatrix}\right) := 4.76h[1]_+ - 7.00h[2]_+ + 0.22$$

# Output layer




$$n\ell_{\text{nnet}}\left(\begin{bmatrix} x_{\text{hour}} \\ x_{\text{day}} \end{bmatrix}\right) := \underbrace{\begin{bmatrix} 4.76 & -7.00 \end{bmatrix}}_{W_2} h_+ + \underbrace{0.22}_{\alpha_2}$$

# What is the function implemented by the neural network?

Affine transformation that adapts to different inputs

Depends on *which ReLUs are active*

# Which ReLUs are active?

$$h[1] := 0.049x_{\text{hour}} - 0.037x_{\text{day}} + 13.0$$

is positive if

$$x_{\text{day}} < 351 + 1.32x_{\text{hour}}$$

$$h[2] := -0.006x_{\text{hour}} - 0.049x_{\text{day}} + 10.1$$

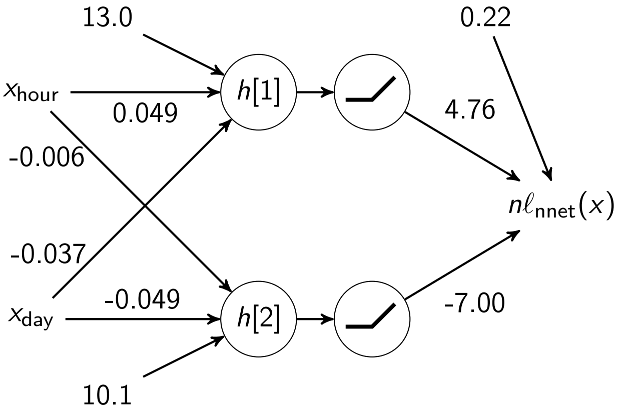
is positive if

$$x_{\text{day}} < 206 - 0.12x_{\text{hour}}$$

Since $x_{\text{hour}} \geq 0$

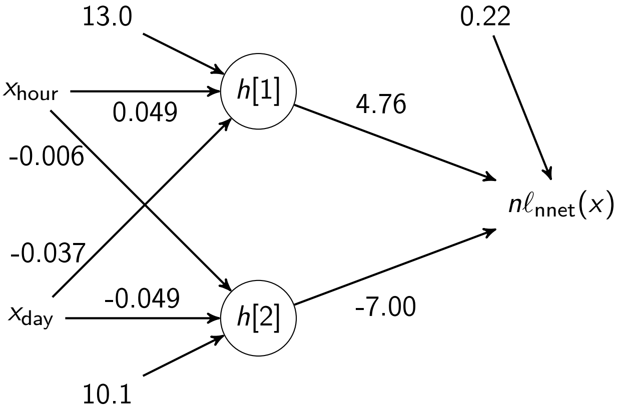
$$206 - 0.12x_{\text{hour}} < 351 + 1.32x_{\text{hour}}$$

$x_{\text{day}} < 206 - 0.12 x_{\text{hour}}$




Both ReLUs are active

$x_{\text{day}} < 206 - 0.12x_{\text{hour}}$




$$n\ell_{\text{nnet}}\left(\begin{bmatrix} x_{\text{hour}} \\ x_{\text{day}} \end{bmatrix}\right) = \mathbf{0.28}\boldsymbol{x}_{\textbf{hour}} + \mathbf{0.17}\boldsymbol{x}_{\textbf{day}} - \mathbf{8.6}$$

$$351 + 1.32x_{\text{hour}} > x_{\text{day}} > 206 - 0.12x_{\text{hour}}$$

$$h[1] := 0.049x_{\text{hour}} - 0.037x_{\text{day}} + 13.0$$

is positive if

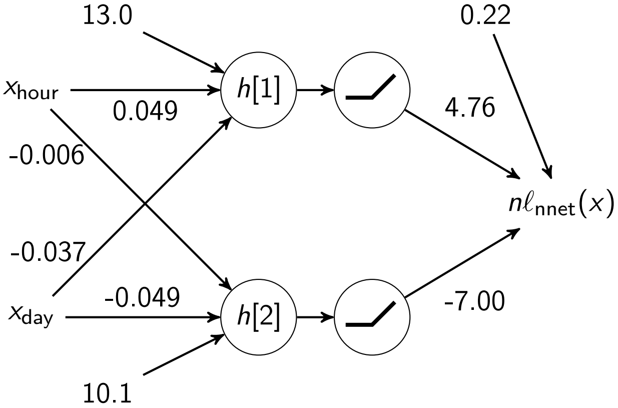
$$x_{\text{day}} < 351 + 1.32x_{\text{hour}}$$

$$h[2] := -0.006x_{\text{hour}} - 0.049x_{\text{day}} + 10.1$$

is negative if

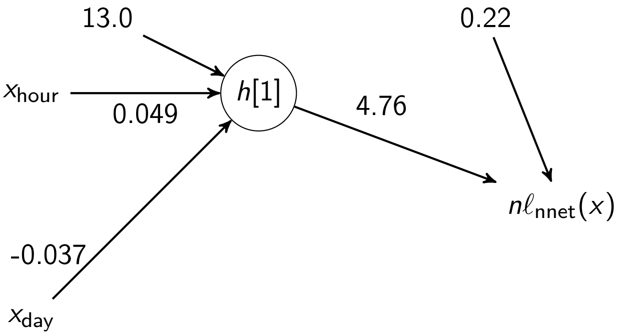
$$x_{\text{day}} > 206 - 0.12x_{\text{hour}}$$

$$351 + 1.32x_{\text{hour}} > x_{\text{day}} > 206 - 0.12x_{\text{hour}}$$




Only the first ReLU is active

$$351 + 1.32x_{\text{hour}} > x_{\text{day}} \geq 206 - 0.12x_{\text{hour}}$$




$$n\ell_{\text{nnet}}\left(\begin{bmatrix} x_{\text{hour}} \\ x_{\text{day}} \end{bmatrix}\right) = \mathbf{0.23}\boldsymbol{x}_{\textbf{hour}} - \mathbf{0.18}\boldsymbol{x}_{\textbf{day}} + \mathbf{62.1}$$

$x_{\text{day}} > 351 + 1.32x_{\text{hour}}$

$$h[1] := 0.049x_{\text{hour}} - 0.037x_{\text{day}} + 13.0$$

is negative if

$$x_{\text{day}} > 351 + 1.32x_{\text{hour}}$$

$$h[2] := -0.006x_{\text{hour}} - 0.049x_{\text{day}} + 10.1$$

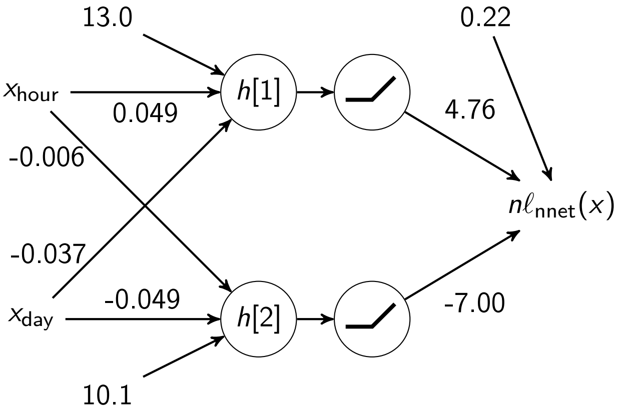
is negative if

$$x_{\text{day}} > 206 - 0.12x_{\text{hour}}$$

Since $x_{\text{hour}} \geq 0$

$$206 - 0.12x_{\text{hour}} < 351 + 1.32x_{\text{hour}}$$

$x_{\text{day}} > 351 + 1.32x_{\text{hour}}$




Neither ReLU is active

$x_{\text{day}} > 351 + 1.32 x_{\text{hour}}$

$$0.22 \longrightarrow n\ell_{\text{nnet}}(x)$$

$$n\ell_{\text{nnet}}\left(\begin{bmatrix} x_{\text{hour}} \\ x_{\text{day}} \end{bmatrix}\right) = \mathbf{0.22}$$

# Function implemented by the neural network

Affine transformation that adapts to different inputs

- If $x_{\text{day}} < 206 - 0.12x_{\text{hour}}$

$$n\ell_{\text{nnet}}\left(\begin{bmatrix} x_{\text{hour}} \\ x_{\text{day}} \end{bmatrix}\right) = 0.28x_{\text{hour}} + \mathbf{0.17}\boldsymbol{x}_{\textbf{day}} - 8.6$$

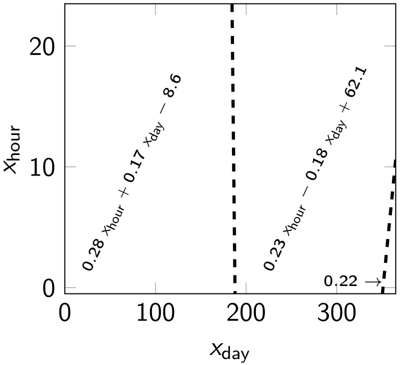
- If $206 - 0.12x_{\text{hour}} \leq x_{\text{day}} < 351 + 1.32x_{\text{hour}}$

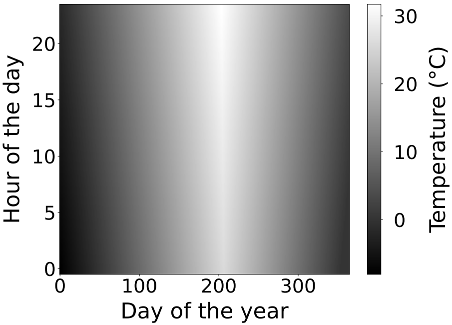
$$n\ell_{\text{nnet}}\left(\begin{bmatrix} x_{\text{hour}} \\ x_{\text{day}} \end{bmatrix}\right) = 0.23x_{\text{hour}} - \mathbf{0.18}\boldsymbol{x}_{\textbf{day}} + 62.1$$

- If $x_{\text{day}} > 351 + 1.32x_{\text{hour}}$

$$n\ell_{\text{nnet}}\left(\begin{bmatrix} x_{\text{hour}} \\ x_{\text{day}} \end{bmatrix}\right) = 0.22$$

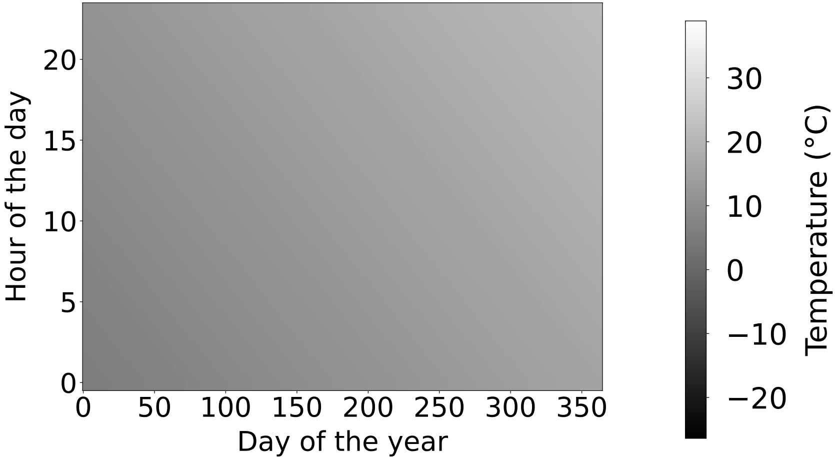
# Neural network




Training error: 6.32°C    Test error: 6.25°C

# Linear model




Training error: 10.8°C    Test error: 11.0°C

# How to train a neural network

Learn parameters $\Theta$ (weights / biases of linear layers) from training set of feature vectors / responses: $(x_1, y_1)$, $(x_2, y_2)$, ...

By minimizing residual sum of squares (RSS)

$$\text{RSS}(\Theta) := \sum_{i=1}^{n} \left( y_i - n\ell_{\text{nnet}}^{[\Theta]}(x_i) \right)^2$$

Nonconvex function of network parameters

General strategy: Minimize iteratively via gradient descent

# Stochastic gradient descent

Problem: Training sets are usually too large

Solution: Separate into batches of size $n_B$

Compute gradient of batch RSS (via backpropagation)

$$\mathrm{RSS}(\Theta) := \sum_{i=1}^{n_B} \left( y_i - n\ell_{\mathrm{nnet}}^{[\Theta]}(x_i) \right)^2$$

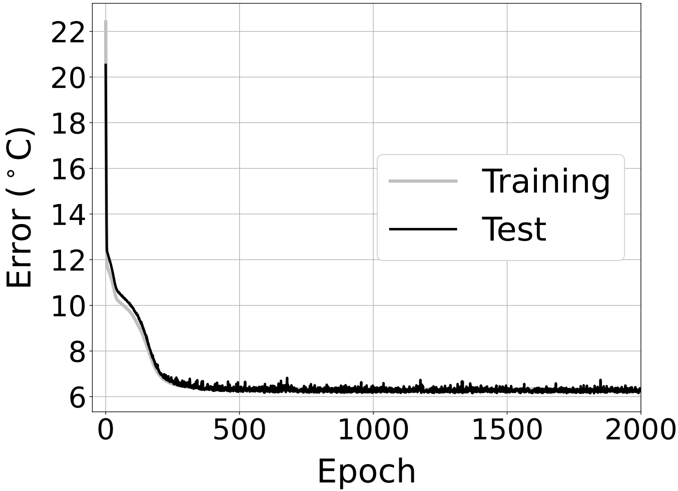
Take a step in the direction opposite to the gradient

$$\Theta_b := \Theta_{b-1} - \eta \nabla_\Theta \, \mathrm{RSS}\left(\Theta_{b-1}\right)$$

Adjusting learning rate $\eta$ is *crucial*

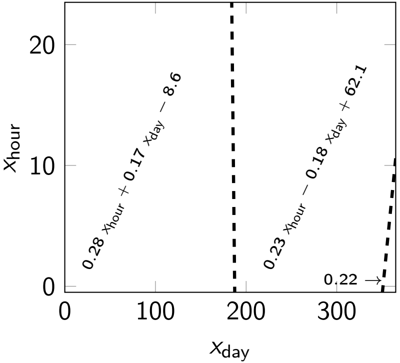
Automatic adaption via e.g. Adam (Kingma and Ba, 2014) is often helpful

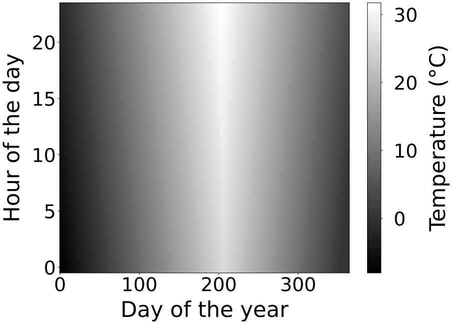
# Training



Error (°C)
22
20
18
16
14
12
10
8
6
0
500
1000
1500
2000
Epoch
Training
Test

# How can we improve the model?




Training error: 6.32°C  Test error: 6.25°C

Make network deeper!

# 4-layer network

$$h^{[1]} := W_1 x + \alpha_1 \qquad W_1 \in \mathbb{R}^{100\times 2}, \alpha_1 \in \mathbb{R}^{100}$$

$$h^{[2]} := W_2 h_+^{[1]} + \alpha_2 \qquad W_2 \in \mathbb{R}^{100\times 100}, \alpha_2 \in \mathbb{R}^{100}$$

$$h^{[3]} := W_3 h_+^{[2]} + \alpha_3 \qquad W_3 \in \mathbb{R}^{100\times 100}, \alpha_3 \in \mathbb{R}^{100}$$

$$n\ell_{\text{nnet}}^{[\Theta]}(x) := W_4 h_+^{[3]} + \alpha_4 \qquad W_4 \in \mathbb{R}^{1\times 100}, \alpha_4 \in \mathbb{R}$$

Network parameters: $\Theta := \{W_1, W_2, W_3, W_4, \alpha_1, \alpha_2, \alpha_3, \alpha_4\}$

# Deep learning

Number of data: 8,760

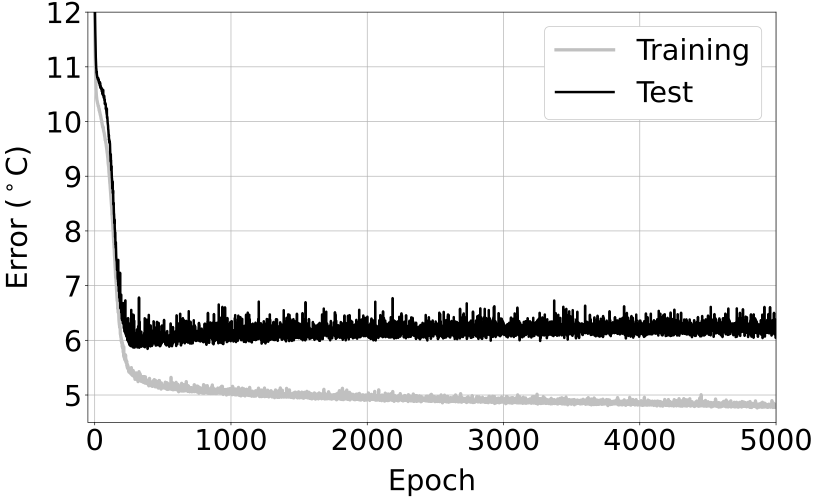
Number of parameters in 4-layer network: 20,601!

Overparametrization

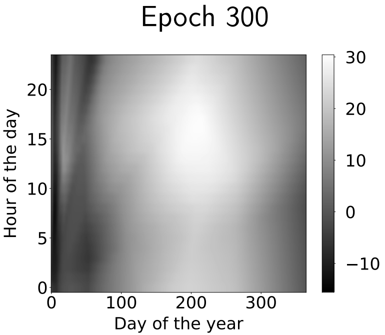
Large vision models have up to billions of parameters

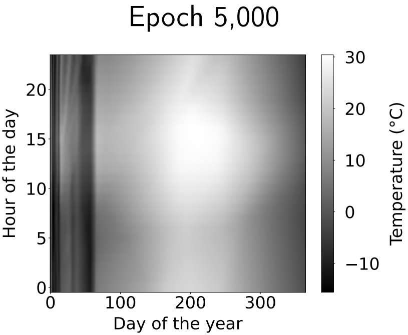
Large language models up to trillions

# But what about overfitting?



# Early stopping mitigates overfitting




Training error: 5.30°C
Test error: 6.06°C

Training error: 4.78°C
Test error: 6.25°C

2-layer network: Training error: 6.32°C Test error: 6.25°C

# What function does the 4-layer network implement?

$$h^{[1]} := W_1 x + \alpha_1 \qquad W_1 \in \mathbb{R}^{100\times 2}, \alpha_1 \in \mathbb{R}^{100}$$

$$h^{[2]} := W_2 h_+^{[1]} + \alpha_2 \qquad W_2 \in \mathbb{R}^{100\times 100}, \alpha_2 \in \mathbb{R}^{100}$$

$$h^{[3]} := W_3 h_+^{[2]} + \alpha_3 \qquad W_3 \in \mathbb{R}^{100\times 100}, \alpha_3 \in \mathbb{R}^{100}$$

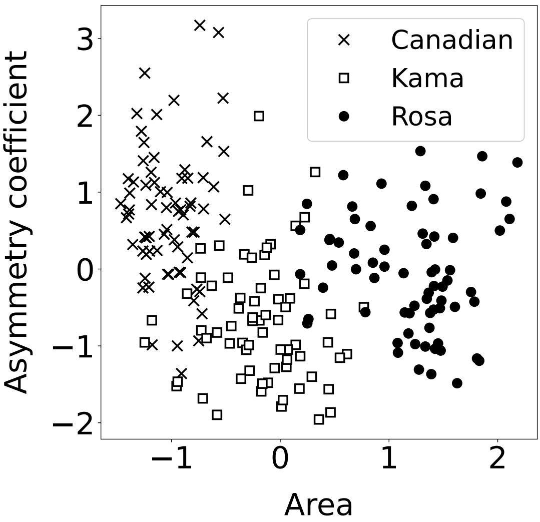
$$n\ell_{\text{nnet}}^{[\Theta]}(x) := W_4 h_+^{[3]} + \alpha_4 \qquad W_4 \in \mathbb{R}^{1\times 100}, \alpha_4 \in \mathbb{R}$$

300 ReLUs

Possible activation states? $2^{300} > 10^{90}$!

No idea...

# Classification



# Classification

Data: $(x_1, y_1)$, $(x_2, y_2)$, ..., $(x_n, y_n)$

Each feature $x_i$ is a $d$-dimensional vector

The label $y_i$ indicates the class (e.g. *Canadian*, *Kama*, or *Rosa*)

Goal: Assign class to new data

# Probabilistic modeling

Model features as random vector $\tilde{x}$ and label as random variable $\tilde{y}$

For new data vector $x$:

$$\hat{y} := \arg \max_{y \in \{1,2,\ldots,c\}} p_{\tilde{y} \mid \tilde{x}}(y \mid x)$$

Is classification easy? No, due to curse of dimensionality!

# Discriminative classification

Goal: Approximate $p_{\tilde{y} \mid \tilde{x}}(k \mid x)$ for $1 \leq k \leq c$

Logistic and softmax regression:

Linear function of features mapped to probabilities

Neural network:

Nonlinear function of features mapped to probabilities

# Neural-network classifier

Softmax regression

$$p_{\Theta}(x)_k := \frac{\exp\left(\beta_k^T x + \alpha_k\right)}{\sum_{l=1}^{c} \exp\left(\beta_l^T x + \alpha_l\right)} \qquad 1 \leq k \leq c$$

Neural network

$$p_{\Theta}(x)_k := \frac{\exp\left(n\ell_{\text{nnet}}(x)[k]\right)}{\sum_{l=1}^{c} \exp\left(n\ell_{\text{nnet}}(x)[l]\right)} \qquad 1 \leq k \leq c$$

# Likelihood

We model $i$th feature and label as random variables $\tilde{x}_i$ and $\tilde{y}_i$

Assumption 1:
Labels are conditionally independent given the features

Assumption 2:
$\tilde{y}_i$ is conditionally independent from $\{\tilde{x}_m\}_{m \neq i}$ given $\tilde{x}_i$

$$
\begin{aligned}
\mathcal{L}_{XY}(\Theta) &:= \mathrm{P}\left(\tilde{y}_1 = y_1, \dots, \tilde{y}_n = y_n \,|\, \tilde{x}_1 = x_1, \dots, \tilde{x}_n = x_n\right) \\
&= \prod_{i=1}^{n} \mathrm{P}\left(\tilde{y}_i = y_i \,|\, \tilde{x}_1 = x_1, \dots, \tilde{x}_n = x_n\right) \\
&= \prod_{i=1}^{n} \mathrm{P}\left(\tilde{y}_i = y_i \,|\, \tilde{x}_i = x_i\right) \\
&= \prod_{k=1}^{c} \prod_{\{i : y_i = k\}} p_\Theta\left(x_i\right)_k, \quad \Theta\text{: neural network parameters}
\end{aligned}
$$

# Likelihood and log-likelihood

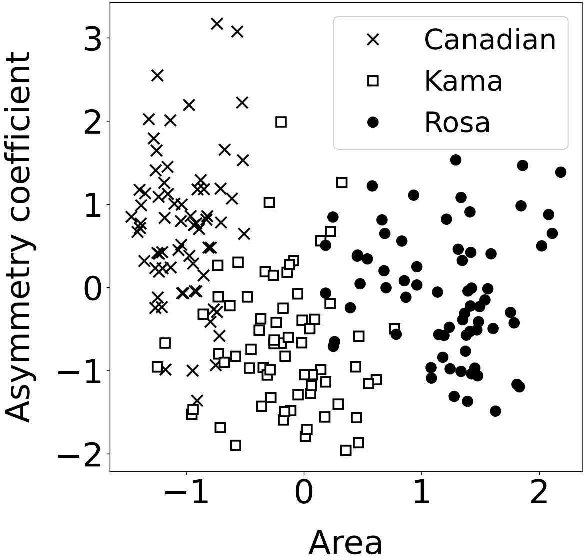
$$\mathcal{L}_{XY}(\Theta) = \prod_{k=1}^{c} \prod_{\{i:y_i=k\}} p_{\Theta}(x_i)_k$$

$$\log \mathcal{L}_{XY}(\Theta) = \sum_{k=1}^{c} \sum_{\{i:y_i=k\}} \log p_{\Theta}(x_i)_k$$

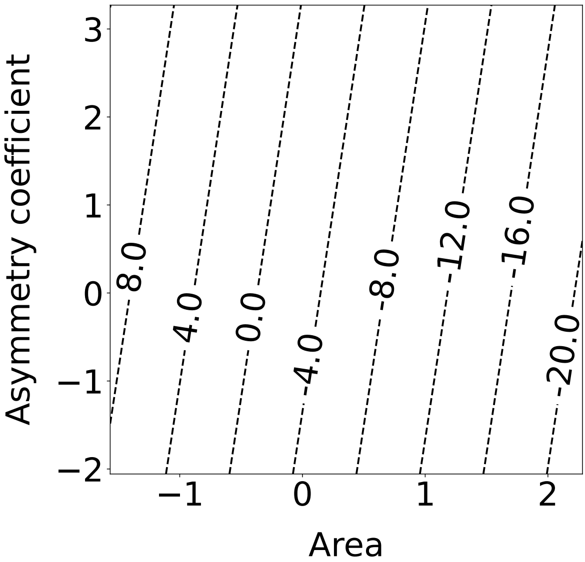
Maximized via stochastic gradient ascent

$-\mathcal{L}_{XY}(\Theta)$ often referred to as cross-entropy loss

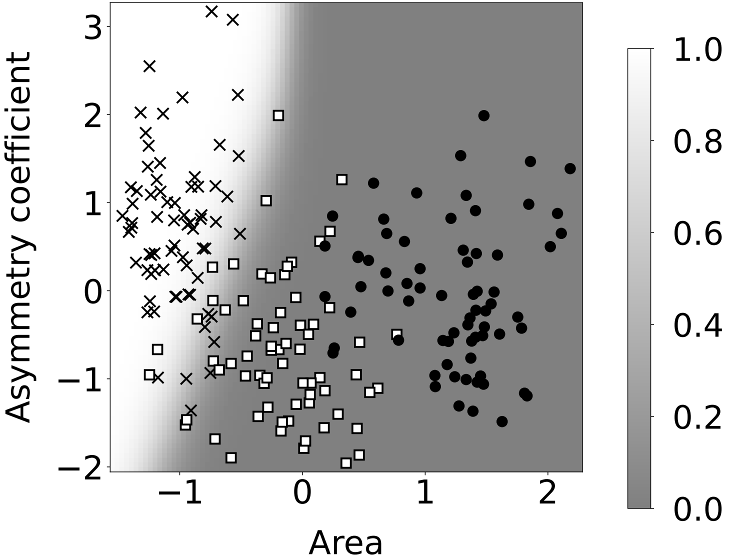
# Wheat varieties



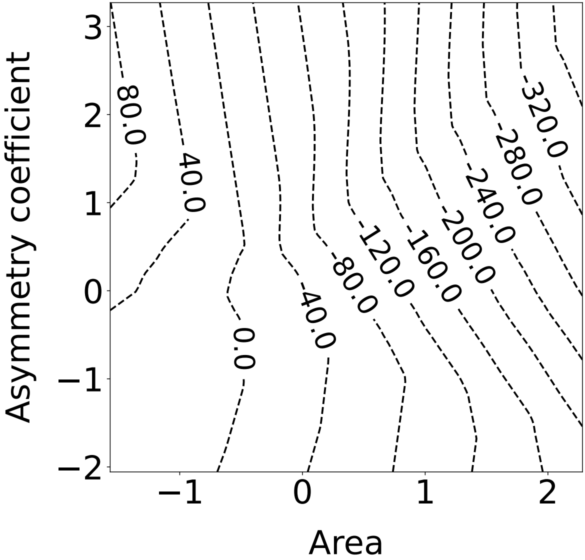
# Linear logits: Canadian



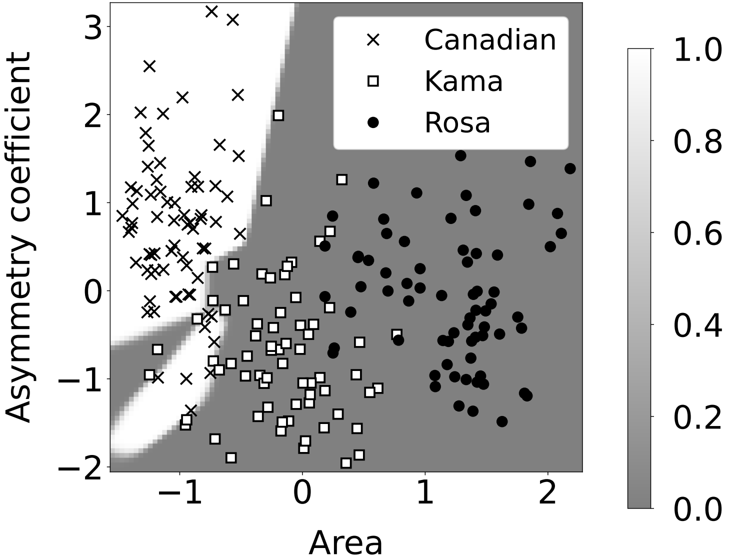
# Estimated probability: Canadian



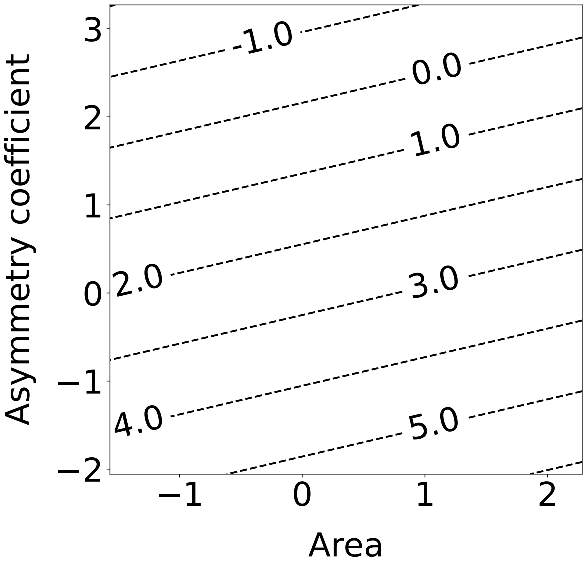
# Nonlinear logits: Canadian



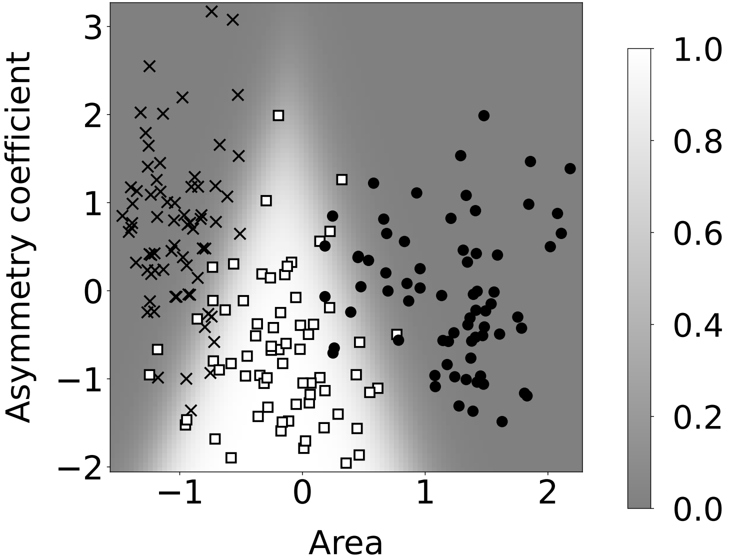
# Estimated probability: Canadian



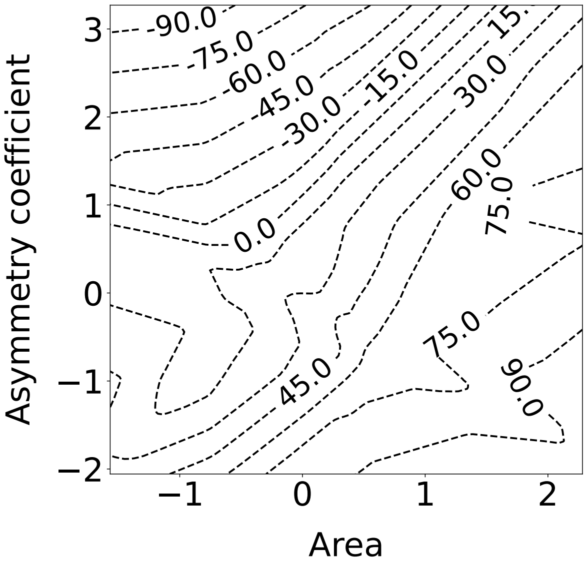
# Linear logits: Kama



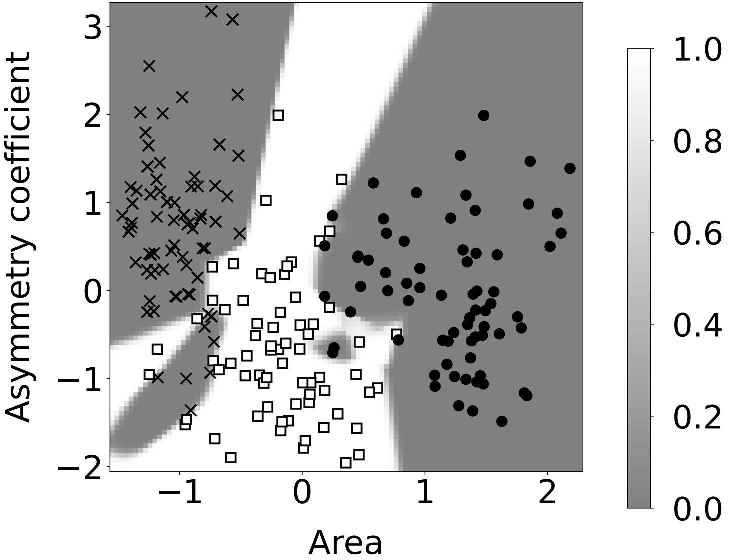
# Estimated probability: Kama



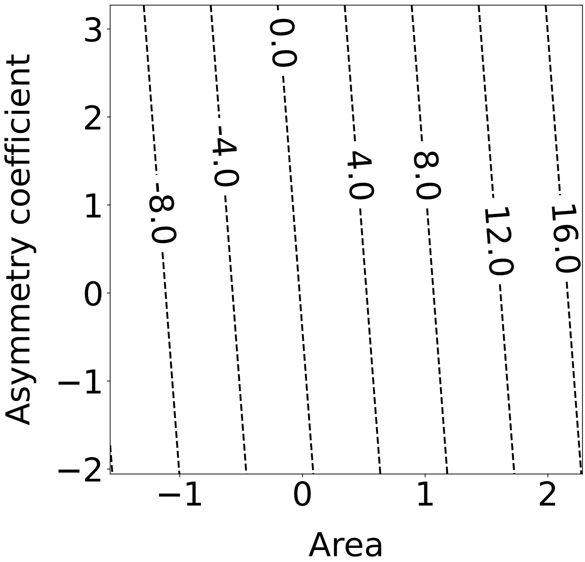
# Nonlinear logits: Kama



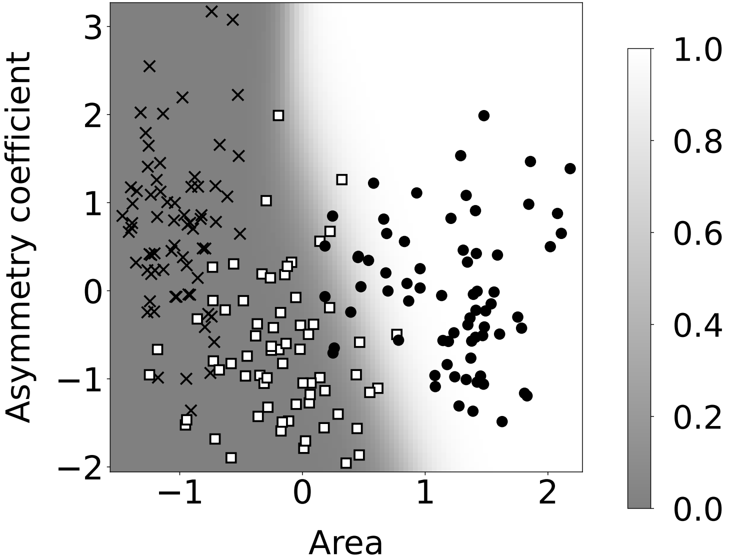
# Estimated probability: Kama



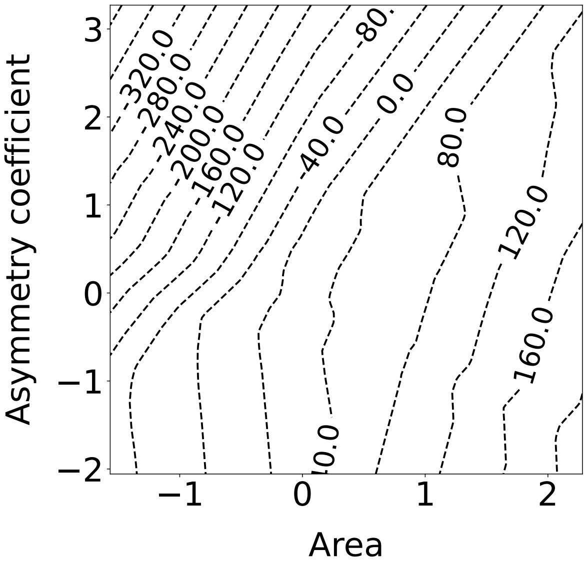
# Linear logits: Rosa



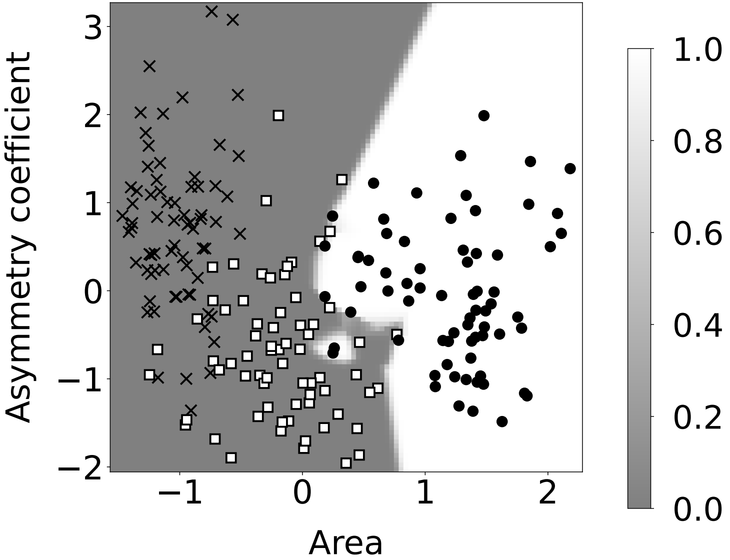
# Estimated probability: Rosa



# Nonlinear logits: Rosa



# Estimated probability: Rosa



# What have we learned?

How to build nonlinear regression and classification models using neural networks

Framework for training neural networks

The power of overparametrization

To be careful about overfitting!