Machine learning crash course

Content draws on material by Nina Balcan, Zico Kolter, and Aditi Raghunathan

Outline

1. **What is machine learning?**
2. Regression
3. Classification
4. (Simple) neural networks

Why machine learning?

The task: write a program where

- **Input:** 28x28 grayscale image of a digit
- **Output:** number in the image

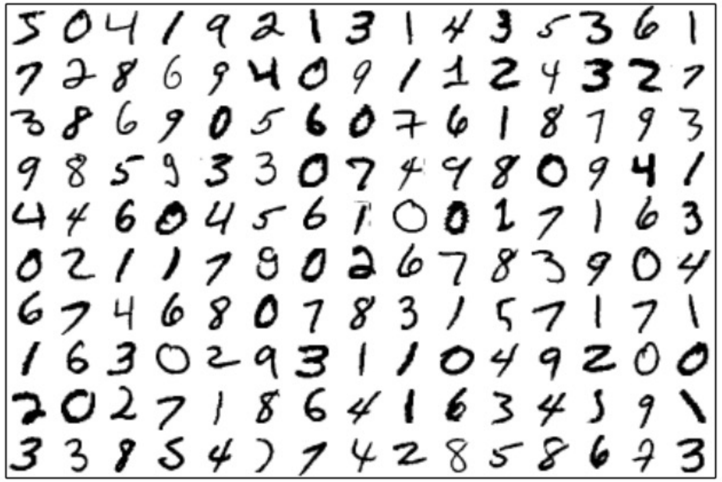


Image: digits from the MNIST data set (http://yann.lecun.com/exdb/mnist/)

Approaches

 → **8**

Approach 1:

- Write a program by hand
- Use your *a priori* knowledge about what numbers look like

Approach 2 (the machine learning approach):

- Collect a dataset of images & their corresponding numbers
- Let the computer "write its own program"
  - Maps these images to their corresponding number

Types of learning

Supervised learning:

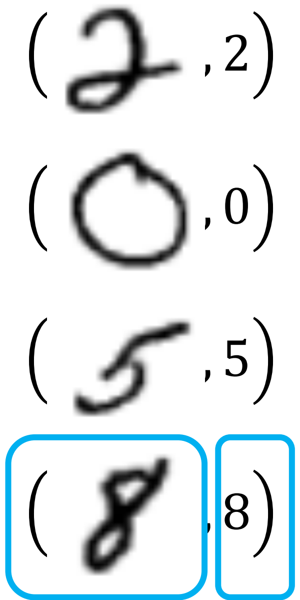
Learn from (input, output) pairs

Unsupervised learning:

Detecting patterns from inputs alone (for e.g. clustering)

Supervised learning pipeline

Training data



Example $x^{(i)} \in \mathcal{X}$

Label $y^{(i)} \in \mathcal{Y}$

$\longrightarrow$ **Machine learning algorithm** $\longrightarrow$

Hypothesis function

$h: \mathcal{X} \to \mathcal{Y}$ such that
$y^{(i)} \approx h(x^{(i)}), \forall i$

On new data $x' \in \mathcal{X}$,
make prediction
$y' = h(x')$

Outline

1. What is machine learning
2. **Regression**
   a. **Linear regression**
   b. Non-linear regression
3. Classification
4. (Simple) neural networks

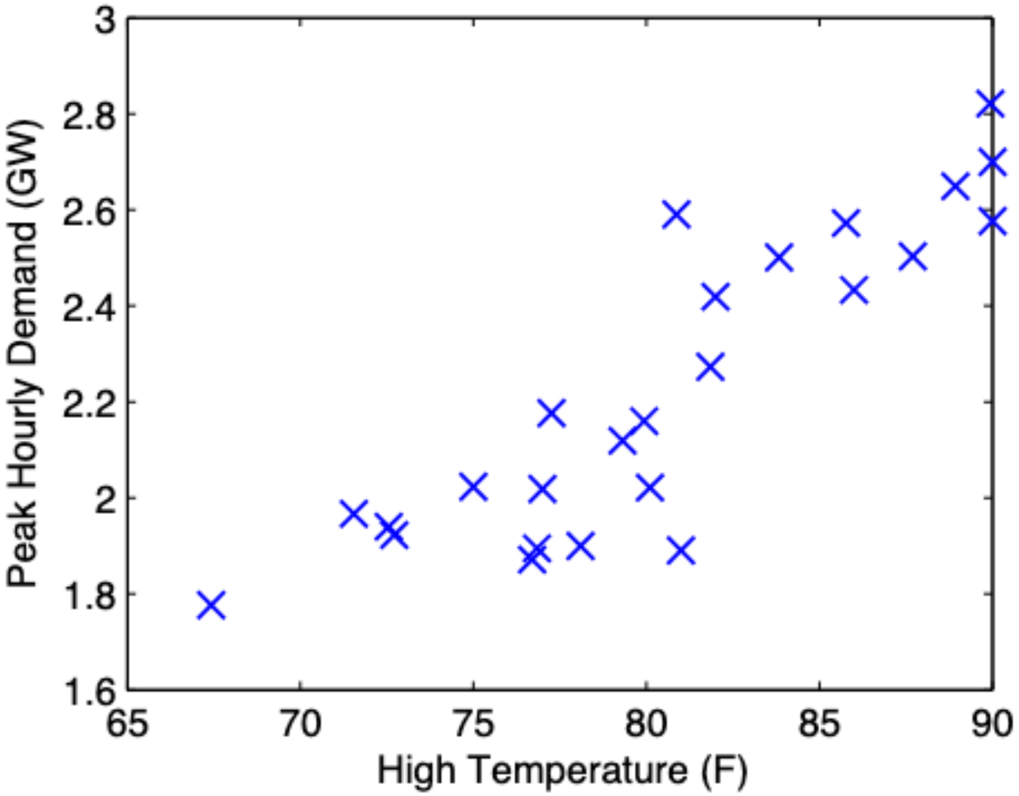
Example: predicting electricity use

- What will peak power consumption be tomorrow?
- Difficult to build an “a priori” model from first principles
- Easy to record past days of consumption
  - Also record additional features that affect consumption (i.e., weather)

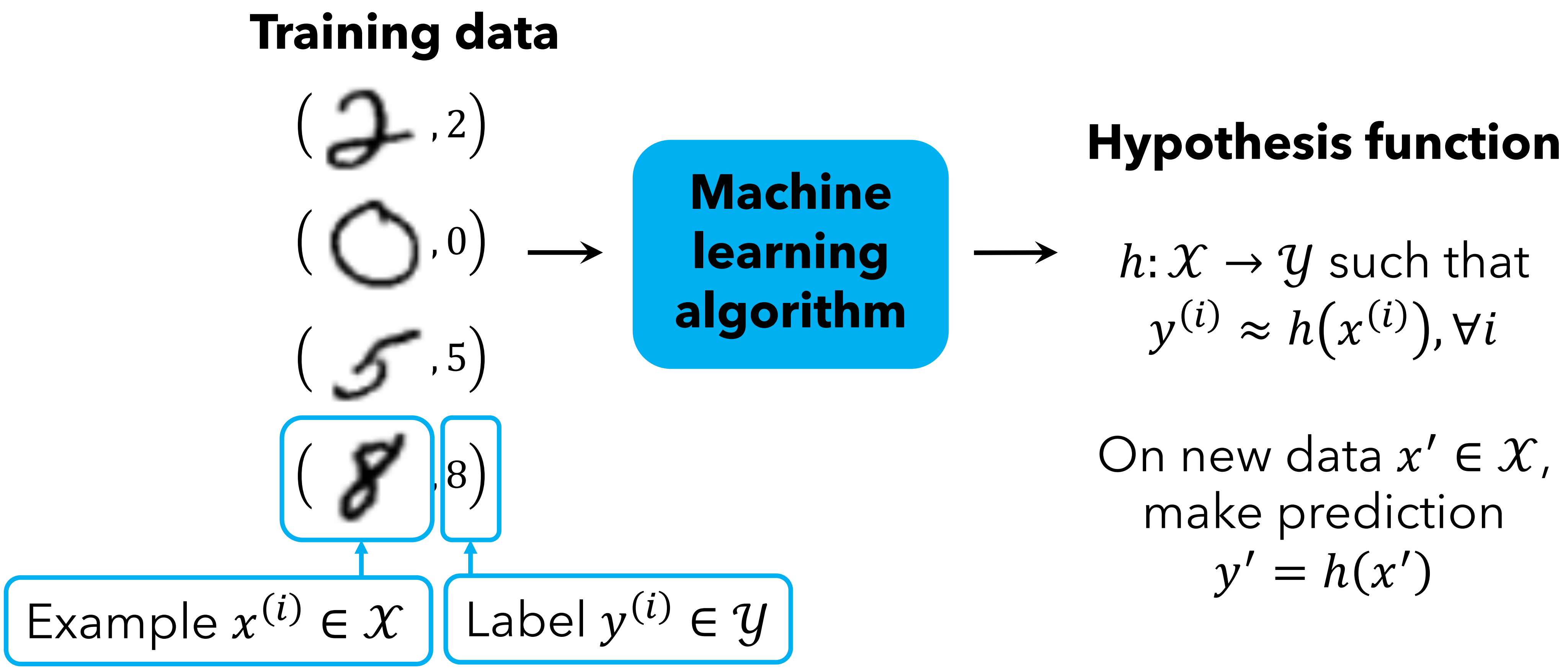
Date	High Temperature (F)	Peak Demand (GW)
2011-06-01	84.0	2.651
2011-06-02	73.0	2.081
2011-06-03	75.2	1.844
2011-06-04	84.9	1.959
…	…	…

Plot of consumption vs. temperature

Several days of peak demand vs. high temperature

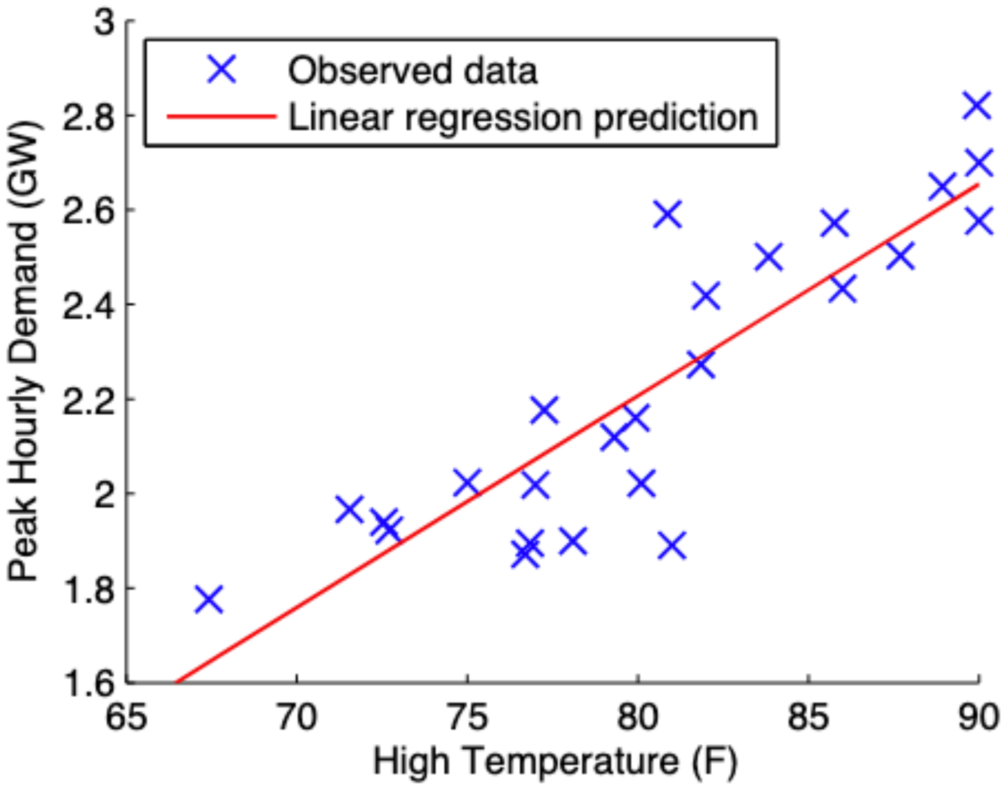


Supervised learning pipeline



Predictions

Predicting is equivalent to “drawing line through data”



Hypothesis: linear model

Suppose peak demand approximately fits a *linear model*

$$\text{Peak\_Demand} \approx \theta_1 \cdot \text{High\_Temperature} + \theta_2$$

- θ_1 is the "slope" of the line
- θ_2 is the intercept

Given forecast of tomorrow's weather, can predict demand

Machine learning notation

Input features: $\boldsymbol{x}^{(i)} \in \mathbb{R}^n, i = 1, \ldots, m$

E.g., $x^{(i)} = \begin{bmatrix} \text{High\_Temperature}^{(i)} \\ 1 \end{bmatrix}$

Outputs: $y^{(i)} \in \mathbb{R}$ (regression task)

E.g., $y^{(i)} \in \mathbb{R} = \text{Peak\_Demand}^{(i)}$

Training data

Hypothesis function: $h_{\boldsymbol{\theta}}: \mathbb{R}^n \to \mathbb{R}$, predicts output given input

E.g., : $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x} = \sum_{j=1}^{n} \theta_j \cdot x_j$

Model parameters: $\boldsymbol{\theta} \in \mathbb{R}^k$ (for linear models $k = n$)

How to obtain best hypothesis?

How good is a hypothesis function?

- Typically done by introducing a **loss function** $\ell \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$
- $\ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$ = how far apart prediction is from actual output
- E.g., common loss function for linear regression is squared error:

$$\ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y)^2$$

Optimization:

Find ***best*** hypothesis (i.e. with smallest loss) on training data

Canonical machine learning problem

Input: Set of input features and outputs $\left(\boldsymbol{x}^{(i)}, y^{(i)}\right), i = 1, \ldots, m$

Task: find the parameters that minimize the sum of losses

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \sum_{i=1}^{m} \ell\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right) \qquad \text{TrainLoss}(\boldsymbol{\theta})$$

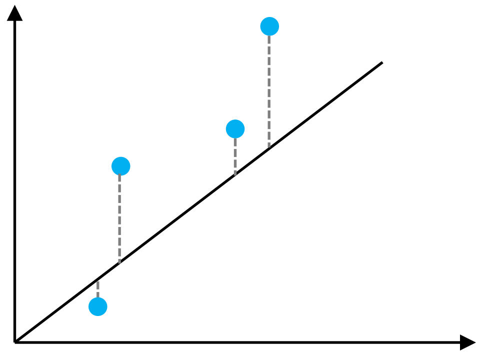
Need to specify:

- What's the hypothesis function?
- What's the loss function?
- How do we solve the optimization problem?

Least squares

In this notation

- **Hypothesis function:** $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x}$
- **Squared loss:** $\ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y)^2$



Leads to ML optimization problem

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \sum_{i=1}^{m} \ell\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right) \equiv \underset{\boldsymbol{\theta}}{\text{minimize}} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - y^{(i)}\right)^2$$

Solution via gradient descent

Gradient descent to solve optimization problem

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - y^{(i)}\right)^2$$

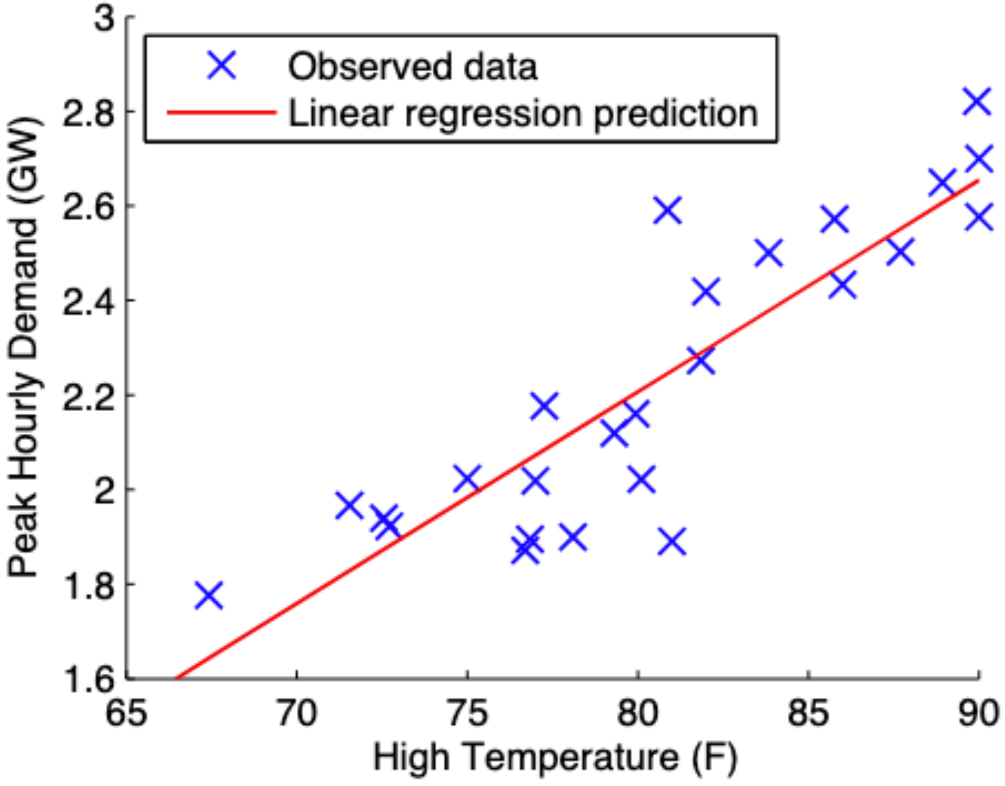
Gradient given by

$$\nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - y^{(i)}\right)^2 = 2 \sum_{i=1}^{m} \boldsymbol{x}^{(i)} \left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - y^{(i)}\right)$$

Gradient descent: repeat $\boldsymbol{\theta} \rightarrow \boldsymbol{\theta} - \eta \sum_{i=1}^{m} \boldsymbol{x}^{(i)} (\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - y^{(i)})$

Least squares solution

Gradient descent gives coefficients θ_1, θ_2 leading to the fit:

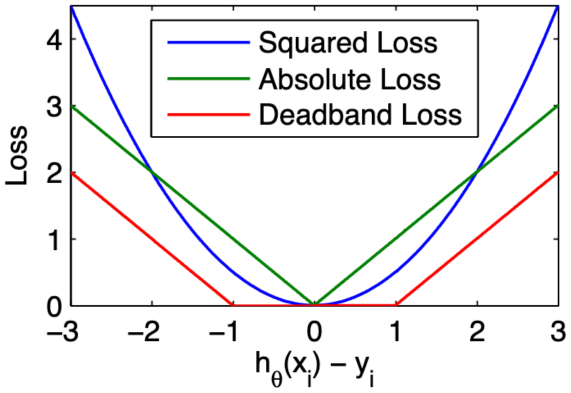


Alternative loss functions

Why did we pick the squared loss $\ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y)^2$?

Some other alternatives:

- Absolute loss: $\ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = |h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y|$
- Deadband loss: $\ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = \max\{0, |h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y| - \epsilon\}$, $\epsilon \in \mathbb{R}$

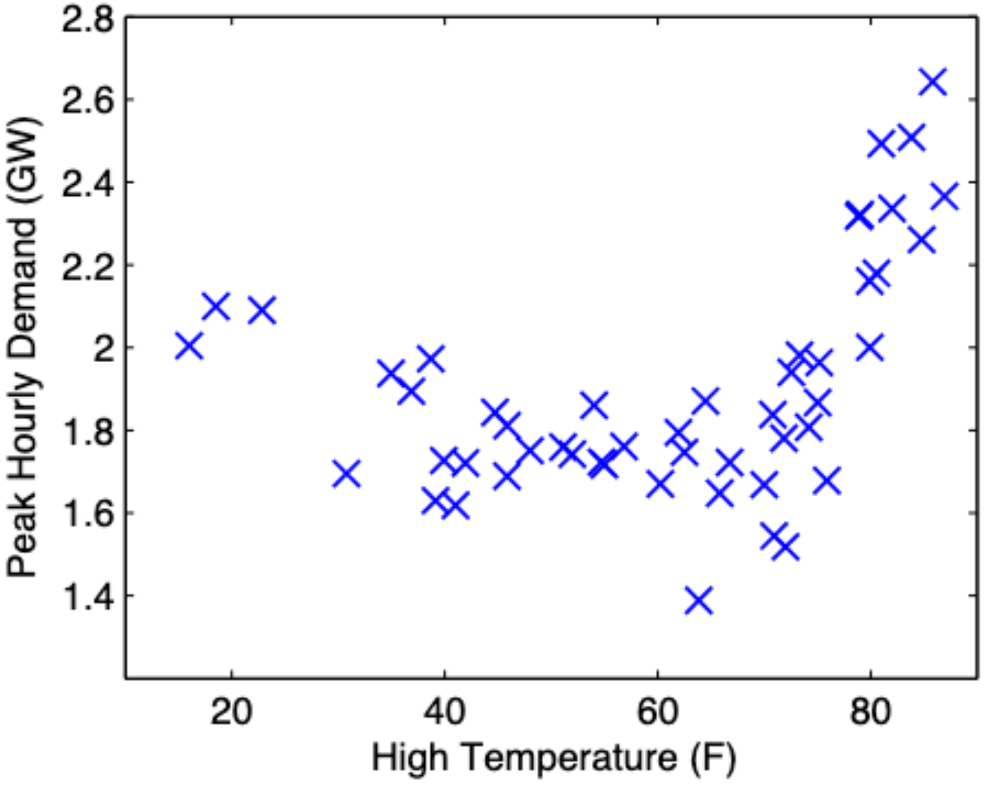


Outline

1. What is machine learning
2. Regression
   a. Linear regression
   b. **Non-linear regression**
3. Classification
4. (Simple) neural networks

Plot of consumption vs. temperature

Several days of peak demand vs. high temperature: all months



"Non-linear" regression

Linear regression applied to non-linear features of input, e.g.:

$$\boldsymbol{x}^{(i)} = \begin{bmatrix} \left(\text{High\_Temperature}^{(i)}\right)^2 \\ \text{High\_Temperature}^{(i)} \\ 1 \end{bmatrix}$$

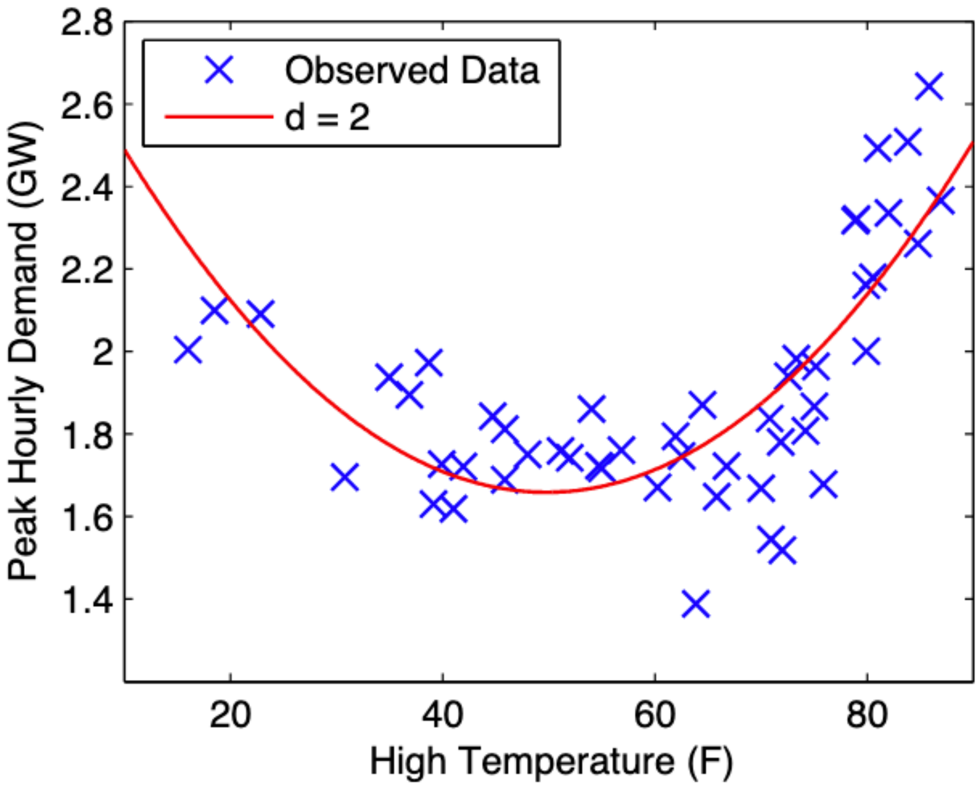
Same hypothesis class as before $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x}$

Prediction will be non-linear (i.e., quadratic) function of base input

Same solution method as before, e.g., gradient descent

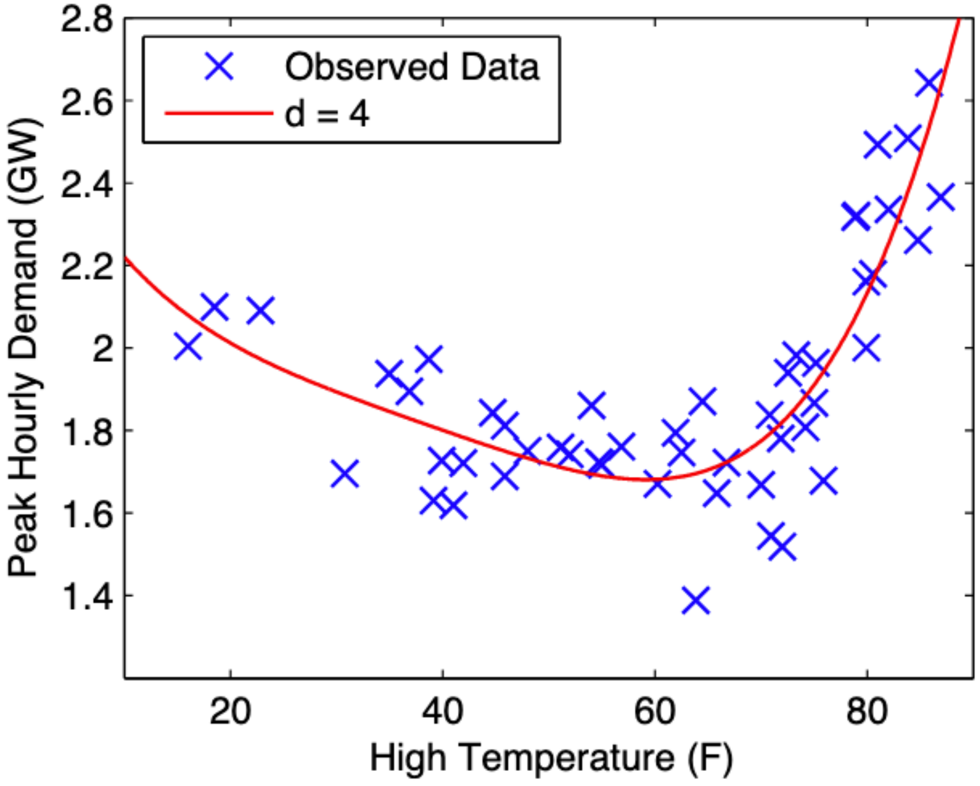
“Non-linear” regression

Linear regression with 2nd degree polynomial features



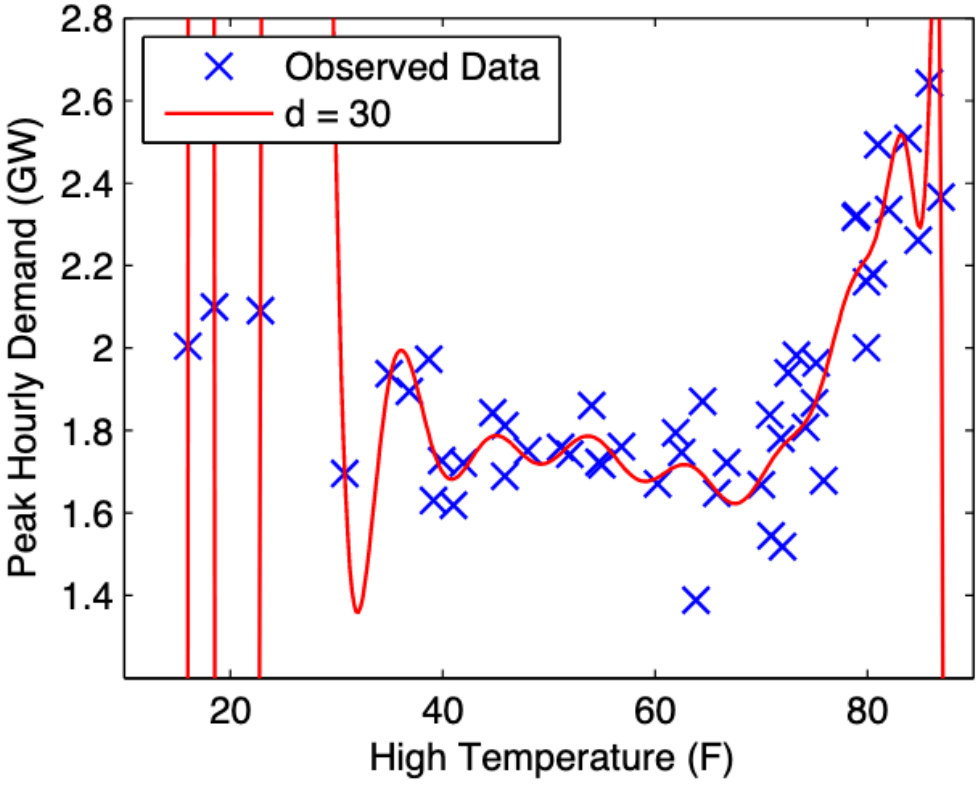
“Non-linear” regression

Linear regression with 4th degree polynomial features



"Non-linear" regression

Linear regression with 30th degree polynomial features



Training and validation loss

Fundamental problem: find parameters that optimize

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \sum_{i=1}^{m} \ell\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right)$$

But what we really care about:

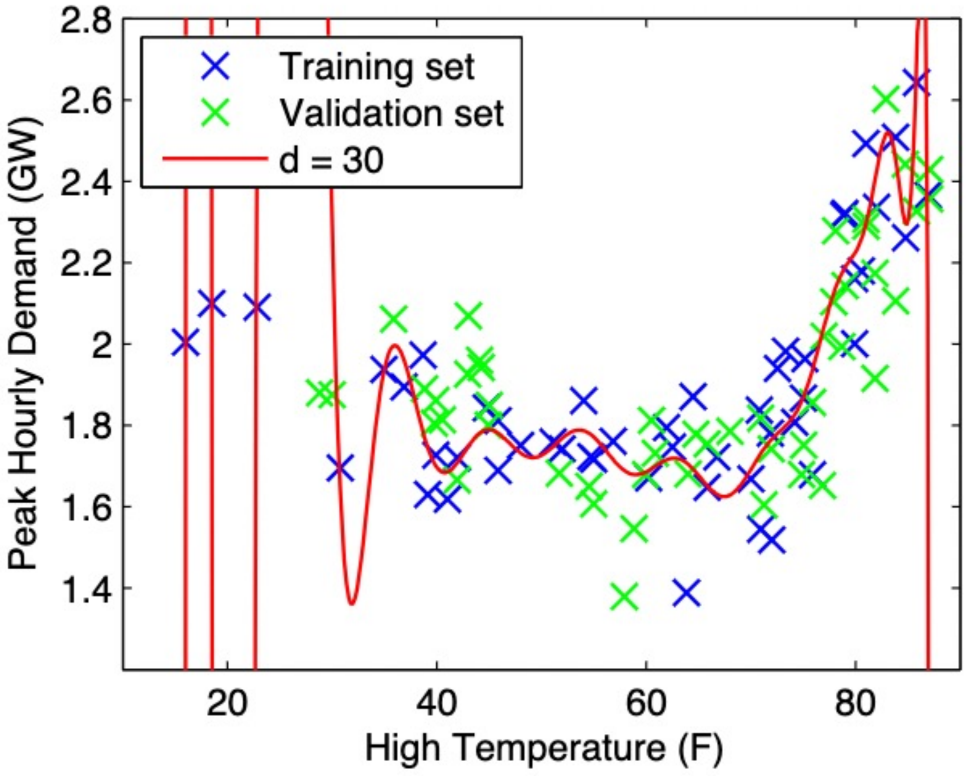
$\ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}'), y')$ on new examples $(\boldsymbol{x}', y')$ ("generalization error")

Divide data into:

- Training set: used to find parameters $\boldsymbol{\theta}$ for fixed hypothesis class H
- Validation set: used to choose H (e.g., polynomial degree)

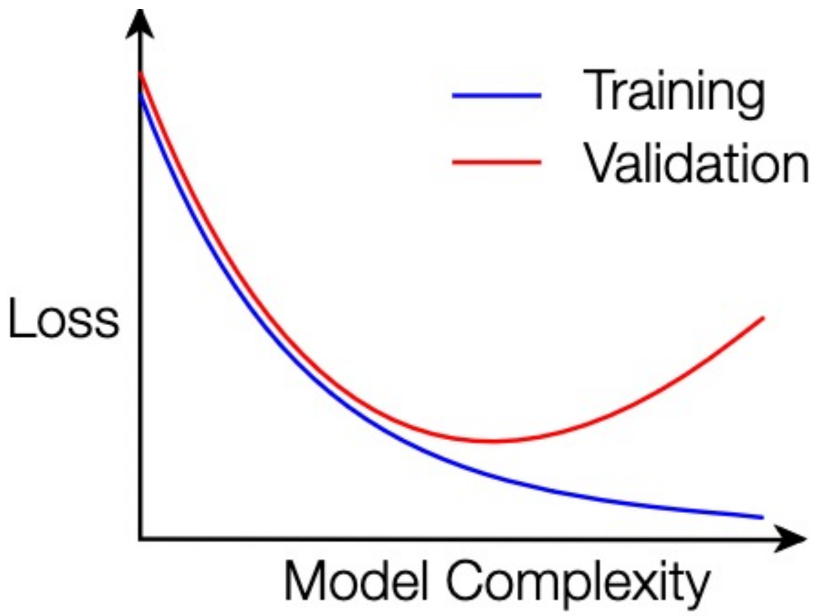
Training and validation loss

Training set and validation set; 30th degree polynomial features



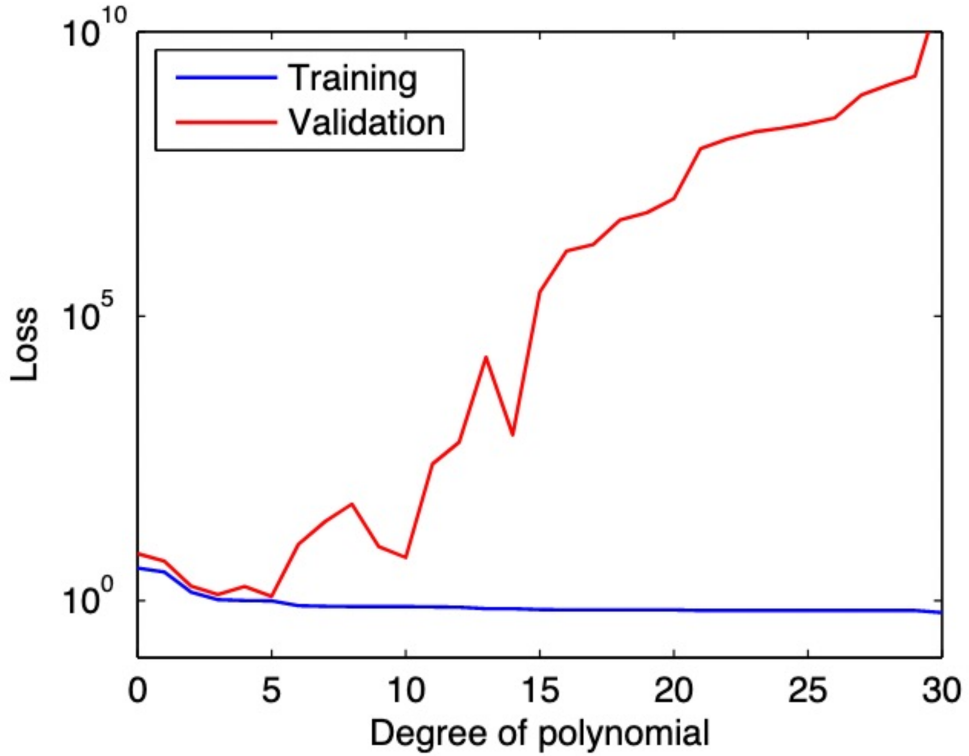
Training and validation loss

General intuition:



Would like hypothesis class that minimizes validation loss

Training and validation loss



Training and validation loss on peak demand prediction

Model complexity and regularization

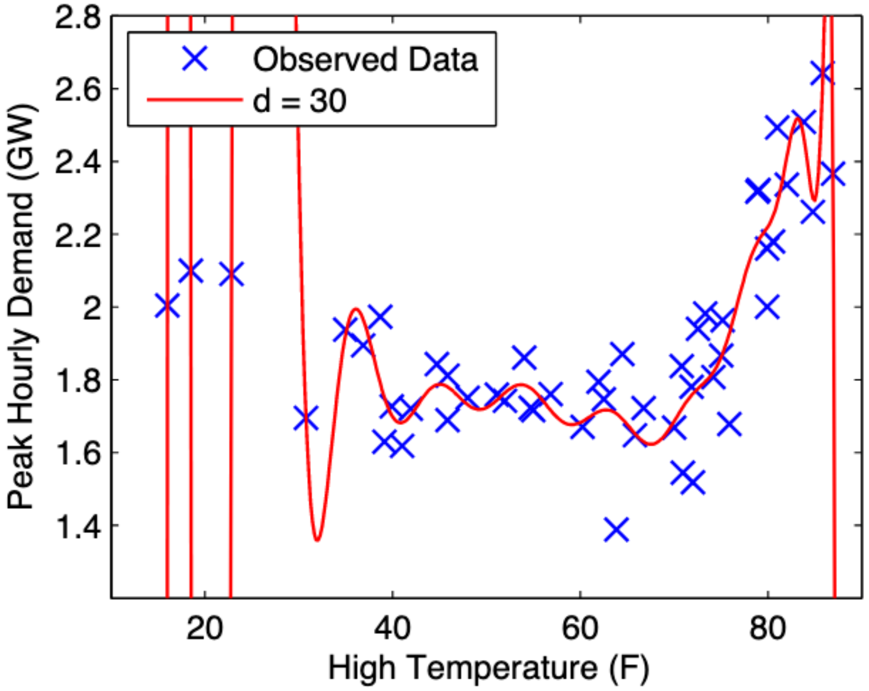
Many different ways to control **“model complexity”**

Obvious one: keep **# of features** (# of parameters) low

Less obvious method: keep **magnitude** of the parameters small

Regularization intuition

If 30th degree polynomial that passes through many points
⇒ Requires very large entries in $\boldsymbol{\theta}$



Regularization

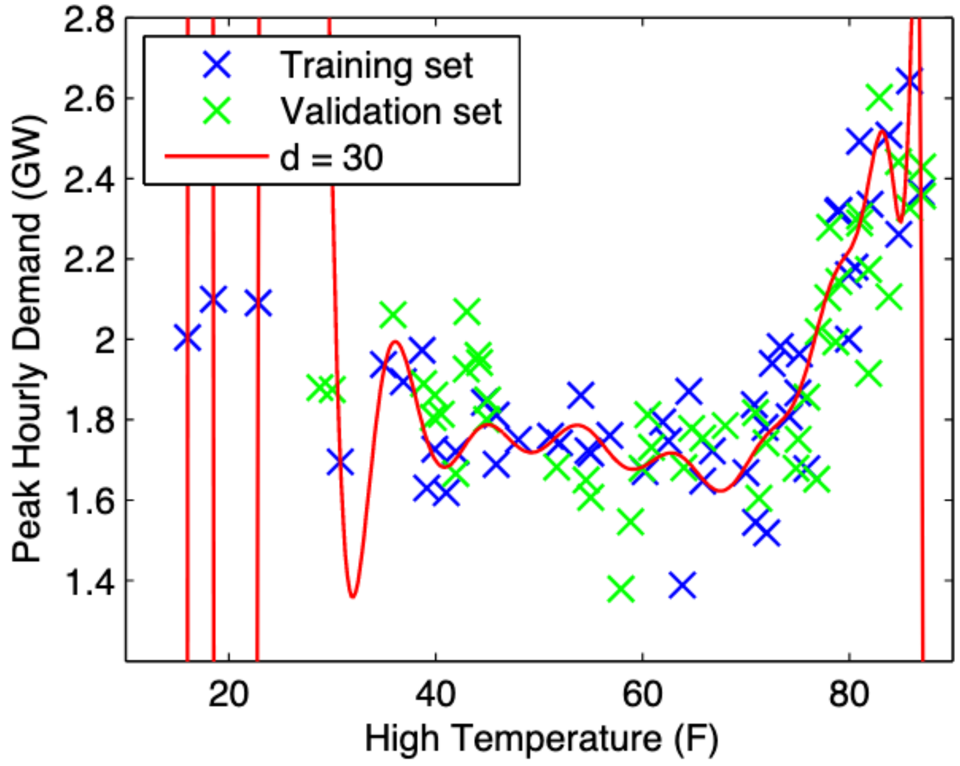
Prevent large entries in $\boldsymbol{\theta}$ by penalizing magnitude of its entries

Leads to **regularized loss minimization** problem

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \sum_{i=1}^{m} \ell\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right) + \lambda \sum_{i=1}^{n} \theta_i^2$$

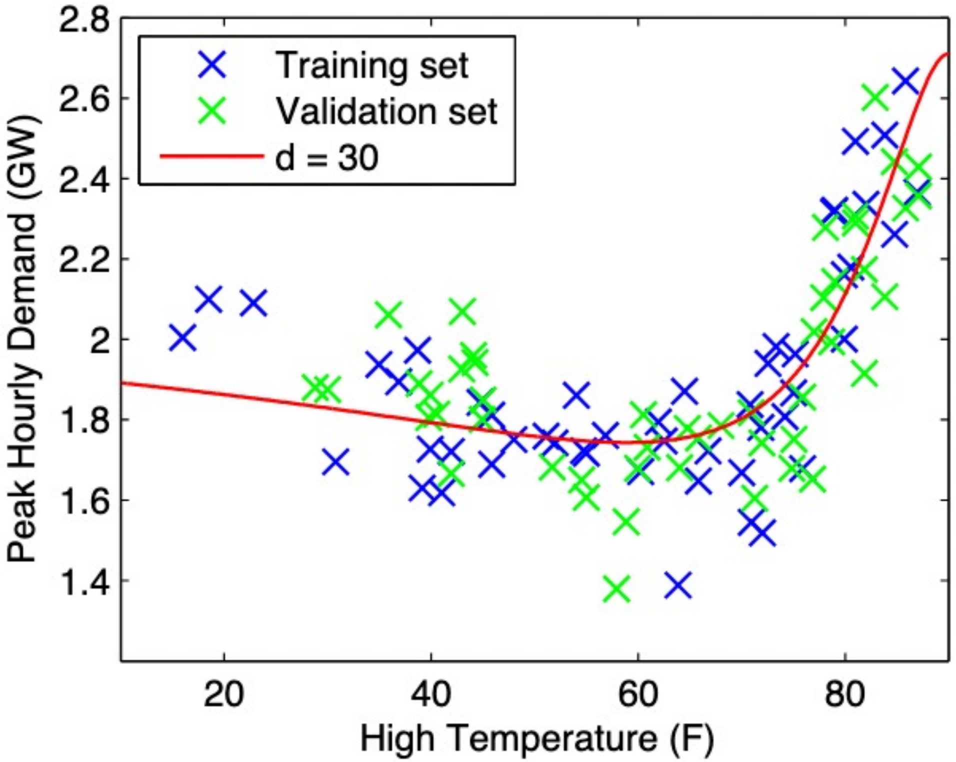
$\lambda > 0$ is a **regularization parameter**

Regularized loss minimization



Degree 30 polynomial, $\lambda = 0$ (unregularized)

Regularized loss minimization



Degree 30 polynomial, $\lambda = 1$

Evaluating ML algorithms

The proper way to evaluate an ML algorithm:

1. Break all data into training/testing sets
   - E.g., 70%/30%
2. Break training set into training/validation set
   - E.g., 70%/30%
3. Choose hyperparameters using validation set
4. (Optional) Retrain using all the training set
5. Evaluate performance on the testing set

Outline

1. What is machine learning
2. Regression
3. **Classification**
4. (Simple) neural networks

Classification problems

Task: predict **discrete** outputs (rather than continuous)

Is the email spam or not? (YES/NO)

What digit is in this image? (0/1/2/3/4/5/6/7/8/9)

Outline

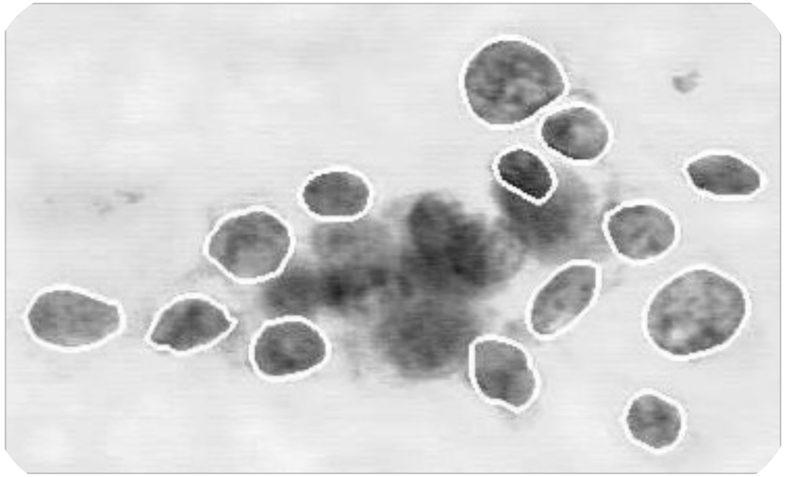
1. What is machine learning
2. Regression
3. Classification
    a. **Linear classification**
    b. Decision trees
    c. Multi-class classification
4. (Simple) neural networks

Example: breast cancer classification

Task: Diagnose whether a tumor is benign or malignant

Doctor's procedure:

1. Extract a sample of fluid from tumor
2. Stains cell
3. Outline several cells

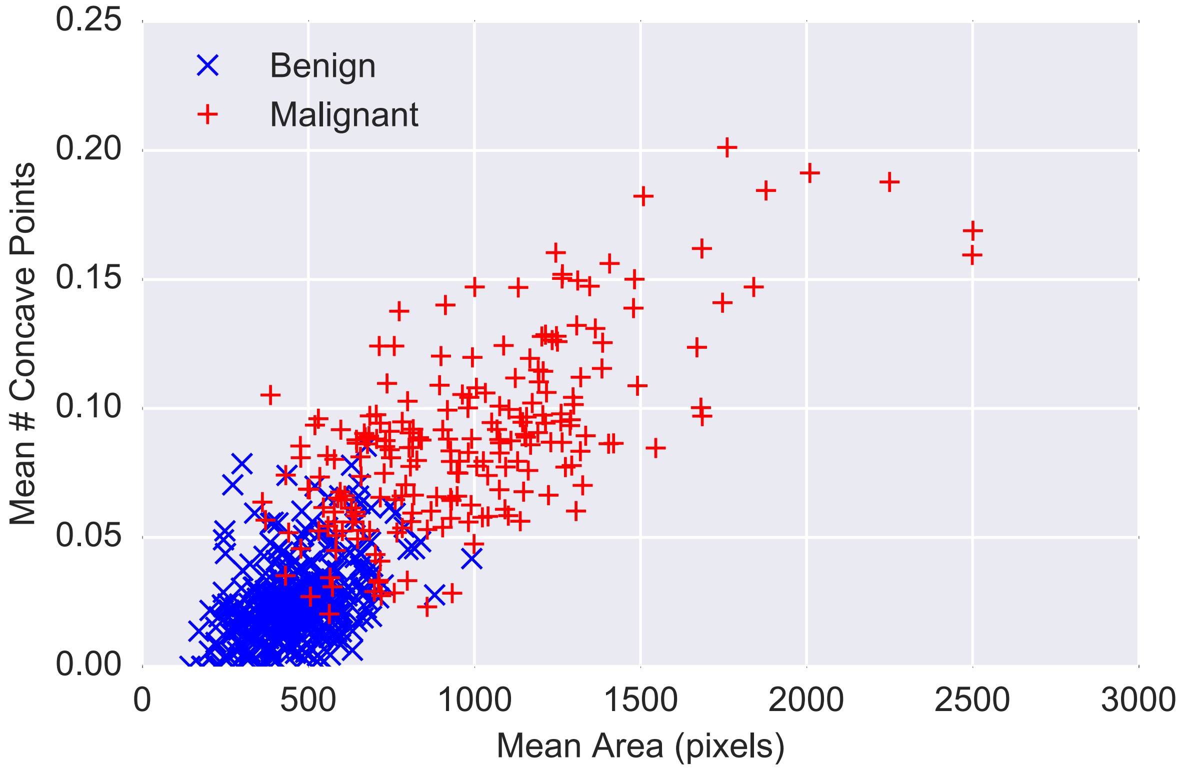


Features for each cell:

Area, perimeter, concavity, texture, …

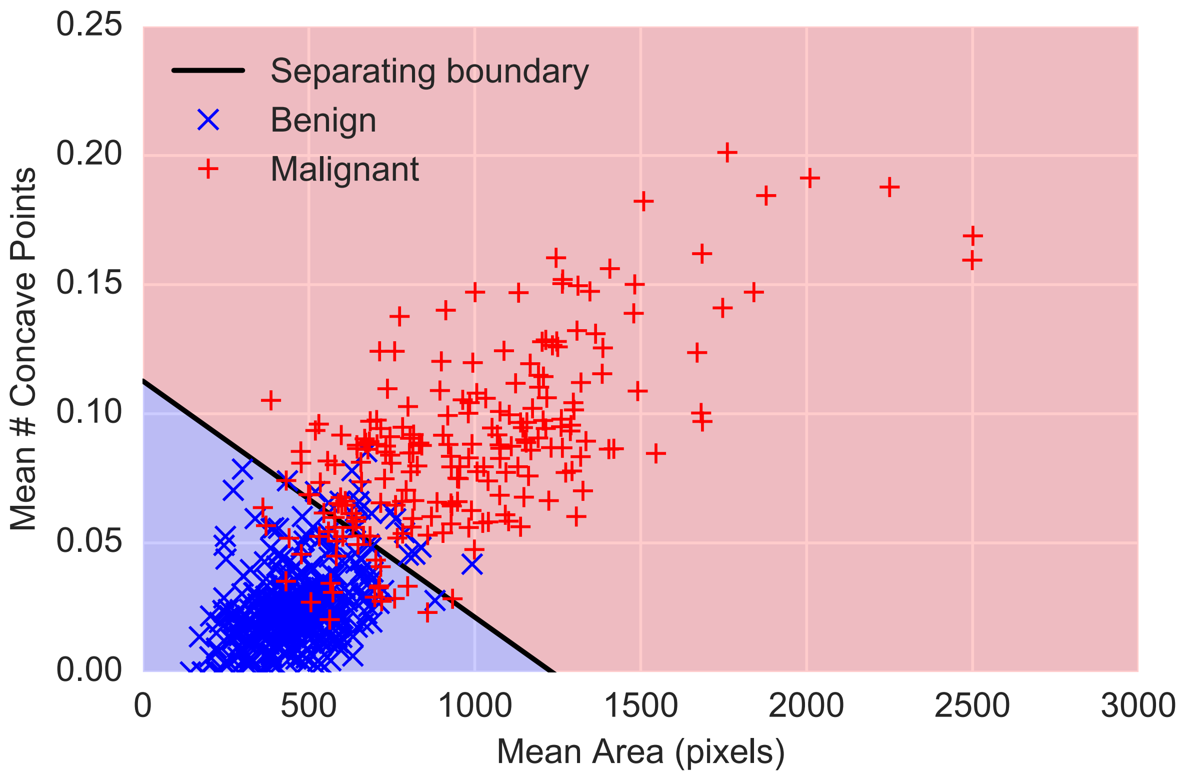
Example: breast cancer classification

Plot of two features: mean area vs. mean concave points



Linear classification example

Linear classification ≡ "class separator is linear"



Formal setting

- **Input features:** $\boldsymbol{x}^{(i)} \in \mathbb{R}^n, i = 1, \dots, m$

  E.g., $\boldsymbol{x}^{(i)} = \begin{bmatrix} \text{Mean\_Area}^{(i)} \\ \text{Mean\_Concave\_Points}^{(i)} \\ 1 \end{bmatrix}$

- **Outputs:** $y^{(i)} \in \{-1, +1\},\ i = 1, \dots, m$

  E.g., $y^{(i)} \in \{-1 \text{ (benign)}, +1 \text{ (malignant)}\}$

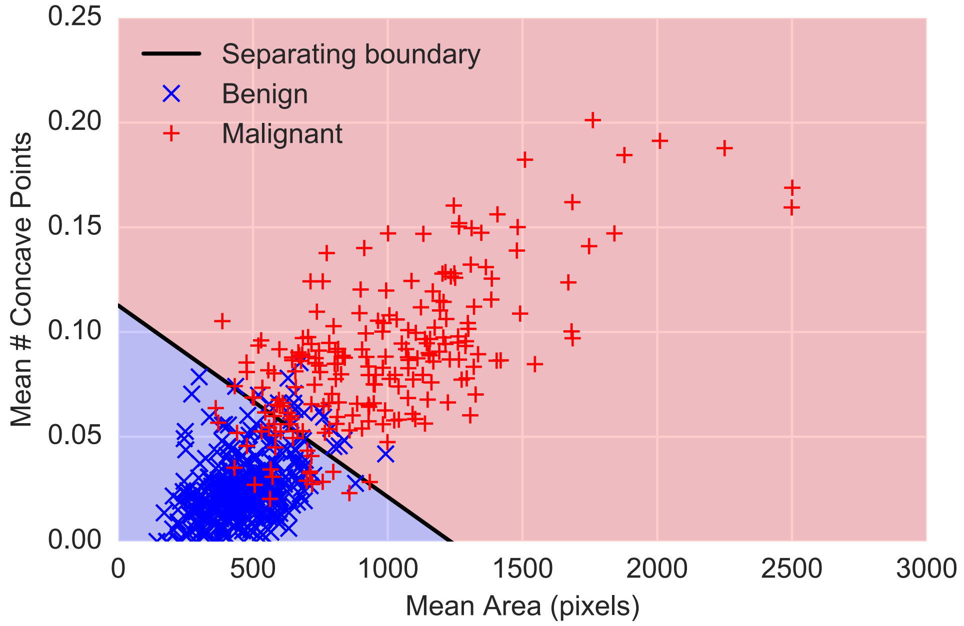
- **Model parameters:** $\boldsymbol{\theta} \in \mathbb{R}^n$

- **Hypothesis:** $h_{\boldsymbol{\theta}}: \mathbb{R}^n \to \mathbb{R}$, aims for same *sign* as output

  E.g., $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x},\ \hat{y} = \operatorname{sign}\big(h_{\boldsymbol{\theta}}(\boldsymbol{x})\big)$

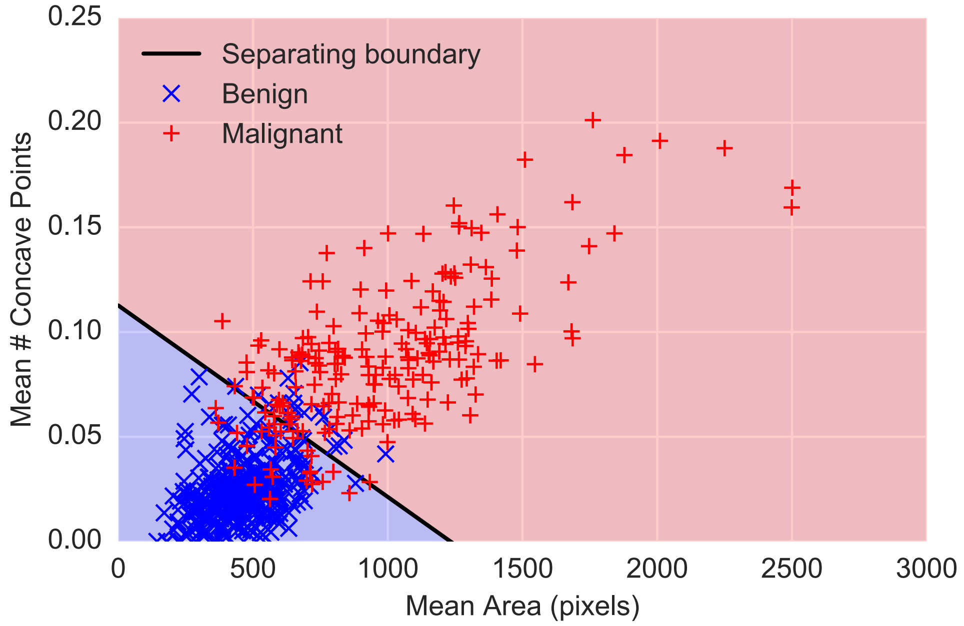
Linear classification diagrams

- Color shows regions where $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ is positive
- Separating boundary is given by the equation $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0$



Linear classification diagrams

- As we move away from decision boundary, $|h_{\boldsymbol{\theta}}(\boldsymbol{x})|$ increases
- $|h_{\boldsymbol{\theta}}(\boldsymbol{x})| = |\boldsymbol{\theta}^T \boldsymbol{x}|$ measures model's "confidence" on input

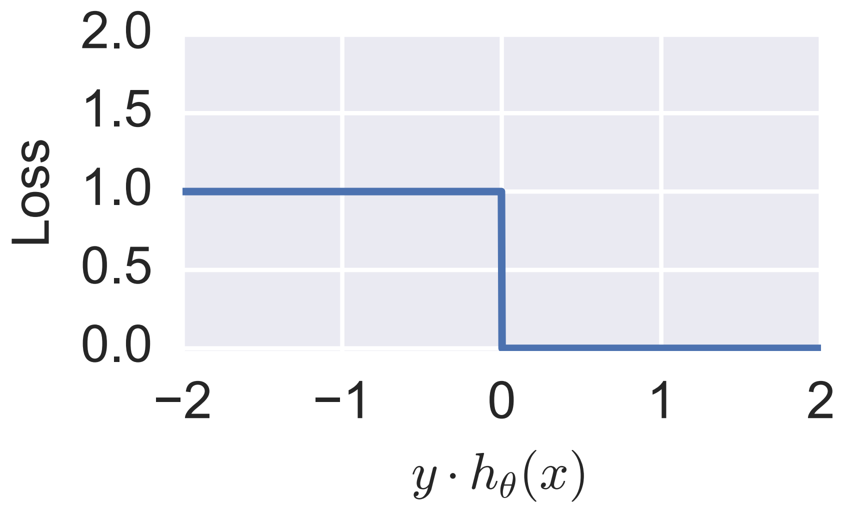


Loss functions

The loss we would like to minimize (0/1 loss, or just "error"):

$$\ell_{0/1}(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = \begin{cases} 0 & \text{if } \operatorname{sign}\big(h_{\boldsymbol{\theta}}(\boldsymbol{x})\big) = y \\ 1 & \text{otherwise} \end{cases}$$

$$= \mathbf{1}\{y \cdot h_{\boldsymbol{\theta}}(\boldsymbol{x}) \le 0\}$$



Alternative losses

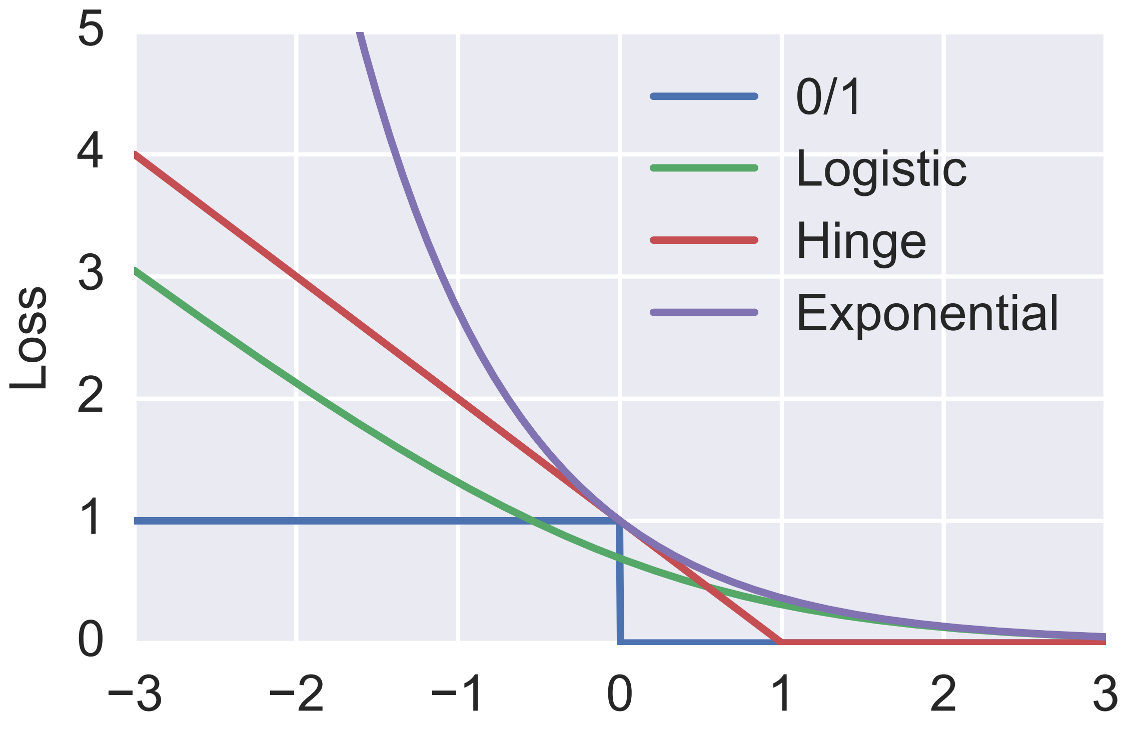
Optimization is hard: $\ell_{0/1}(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y)$ isn't convex

Gradient is zero almost everywhere

Alternative losses for classification are typically used instead

Loss functions depend on $yh_{\boldsymbol{\theta}}(\boldsymbol{x})$ (larger the better)

Alternative losses



More confident and wrong $y \cdot h_\theta(x)$ *More confident and right*

$$\ell_{0/1} = 1\{y \cdot h_\theta(x) \leq 0\}$$
$$\ell_{\text{logistic}} = \log\big(1 + \exp\big(-y \cdot h_\theta(x)\big)\big)$$
$$\ell_{\text{hinge}} = \max\{1 - y \cdot h_\theta(x), 0\}$$
$$\ell_{\text{exp}} = \exp(-y \cdot h_\theta(x))$$

Support vector machine (SVM)

SVM uses regularized **hinge loss** and a **linear** hypothesis

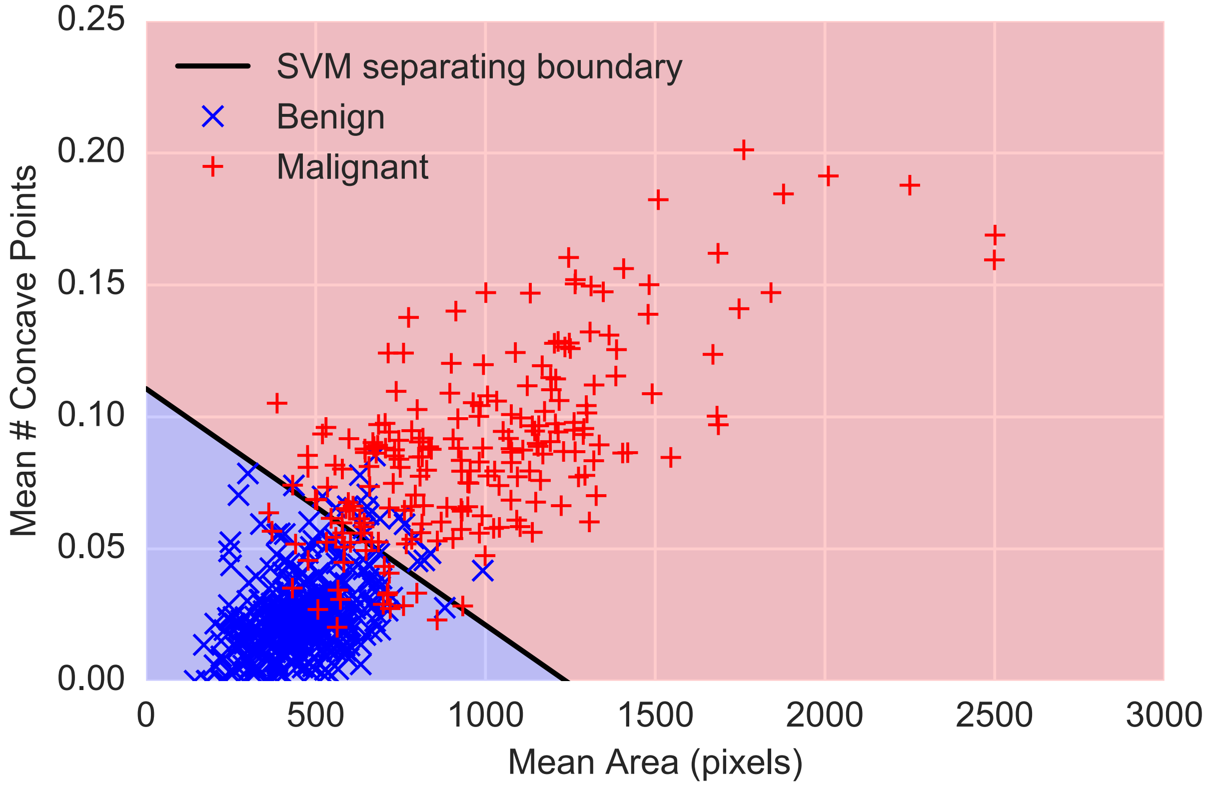
$$\underset{\boldsymbol{\theta}}{\text{minimize}} \sum_{i=1}^{m} \underbrace{\max\{1 - y^{(i)} \cdot \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}, 0\}}_{\text{Linear penalty if wrong and no penalty if confidently correct}} + \lambda \sum_{i=1}^{n} \theta_i^2$$

Updates using gradient descent:

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \eta \sum_{i=1}^{m} -y^{(i)} \boldsymbol{x}^{(i)} \mathbf{1}\{ y^{(i)} \cdot \boldsymbol{\theta}^T \boldsymbol{x}^{(i)} \leq 1\}$$

Support vector machine example

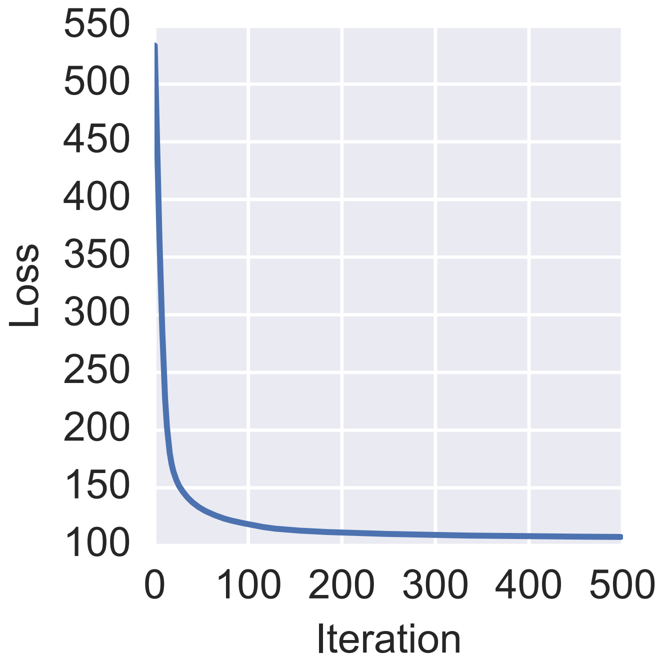
Running support vector machine on cancer dataset



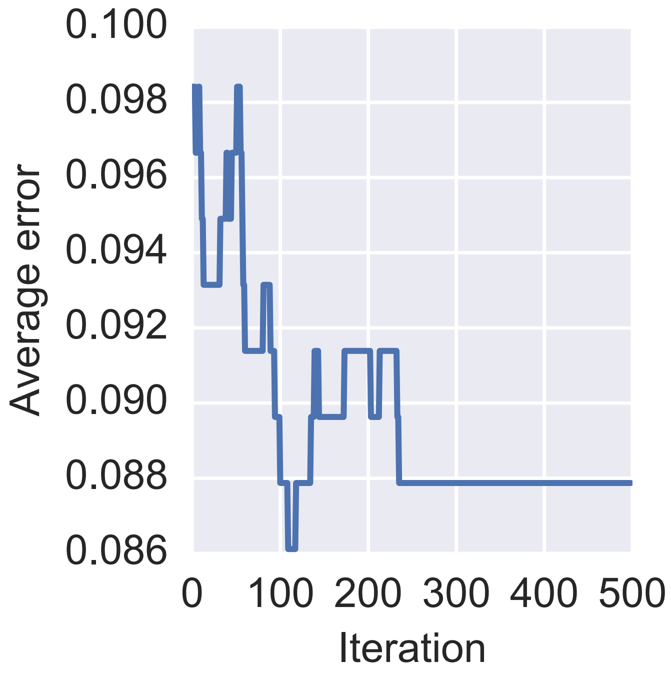
$$\theta = \begin{bmatrix} 1.456 \\ 1.848 \\ -0.189 \end{bmatrix}$$

SVM optimization progress

Optimization objective & error versus gradient descent iteration



Hinge loss



0-1 error

Outline

1. What is machine learning
2. Regression
3. Classification
    a. Linear classification
    b. **Decision trees**
    c. Multi-class classification
4. (Simple) neural networks

Supervised learning example

Example: learn if our friend will play tennis on a given day

Simple training dataset:

	Day	Outlook	Temperature	Humidity	Wind	Play tennis	
example	D1	Sunny	Hot	Normal	Weak	**Yes**	label
	D2	Sunny	Hot	High	Weak	**No**	
	D3	Overcast	Hot	High	Weak	**Yes**	
	D4	Rain	Mild	High	Strong	**No**	
	D5	Rain	Cool	Normal	Weak	**Yes**	

Supervised learning example

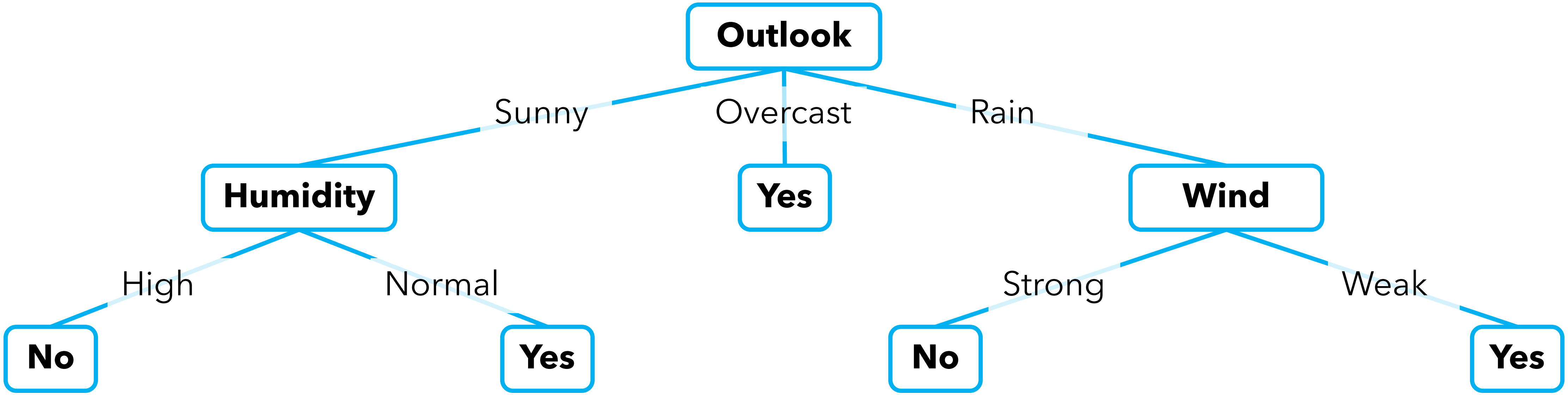
"Labeled example" $(\boldsymbol{x}, y)$ where:

- $\boldsymbol{x} = (x_1, x_2, x_3, x_4) = ($ $\underbrace{\text{Sunny}}_{\text{Outlook}}$ $\underbrace{\text{Hot}}_{\text{Temp}}$ $\underbrace{\text{Normal}}_{\text{Humidity}}$ $\underbrace{\text{Weak}}_{\text{Wind}}$ $)$
- $y =$ "Yes"

	Day	Outlook	Temperature	Humidity	Wind	Play tennis	
example	D1	Sunny	Hot	Normal	Weak	**Yes**	label
	D2	Sunny	Hot	High	Weak	**No**	
	D3	Overcast	Hot	High	Weak	**Yes**	
	D4	Rain	Mild	High	Strong	**No**	
	D5	Rain	Cool	Normal	Weak	**Yes**	

Decision tree learning

- Each **internal node**: test one (discrete-valued) attribute x_i
- Each **branch** from a node: corresponds to a value for x_i
- Each **leaf**: predict y



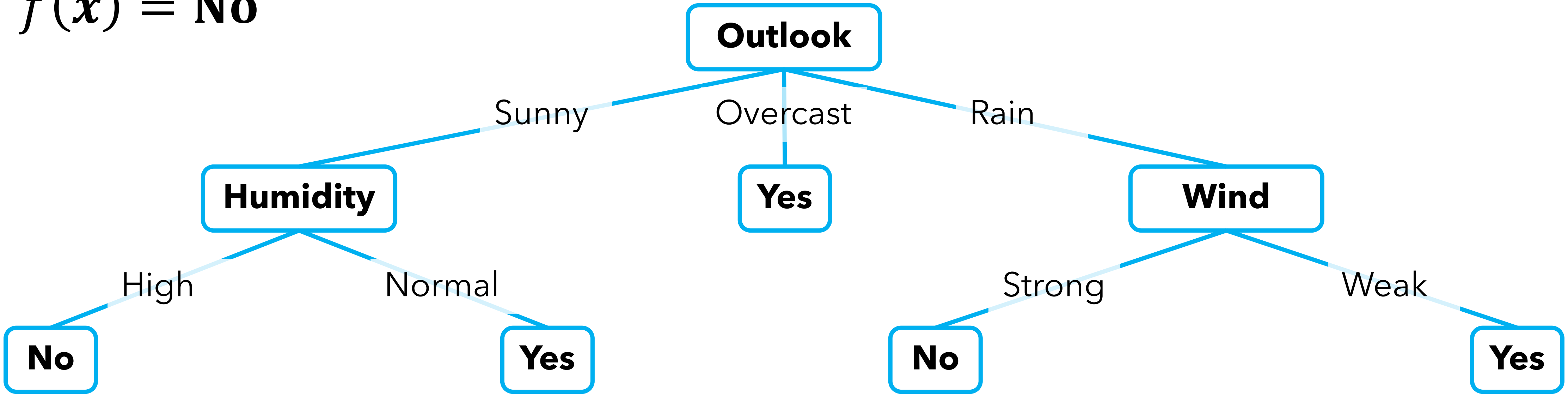
Decision tree learning

$f: <\text{Outlook}, \text{Temperature}, \text{Humidity}, \text{Wind}> \rightarrow \text{PlayTennis?}$

$\boldsymbol{x} = (x_1, x_2, x_3, x_4) =$ (Sunny Hot High Weak)

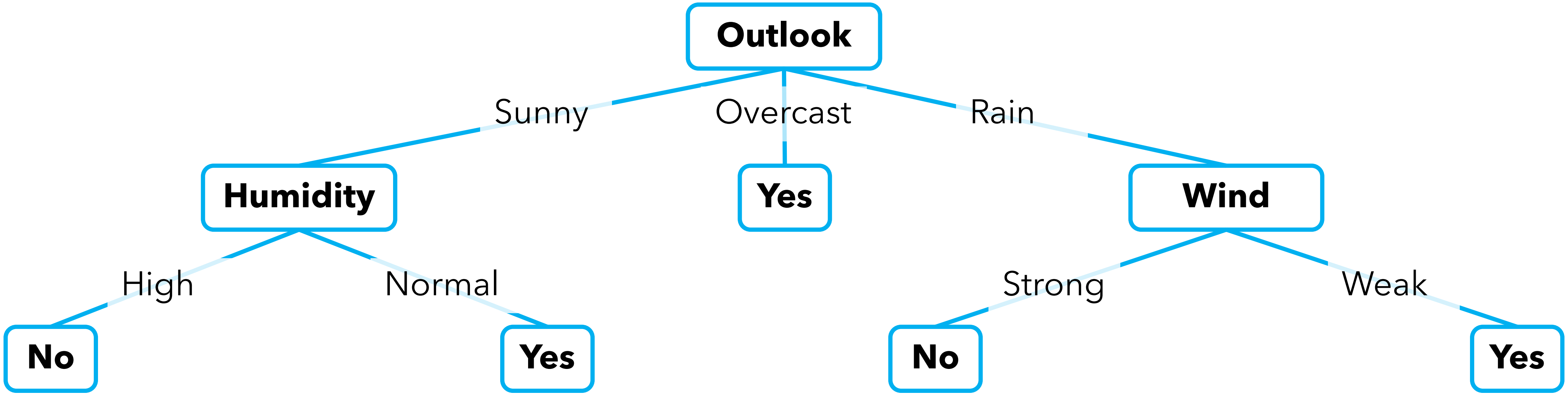
Sunny	Hot	High	Weak
Outlook	Temp	Humidity	Wind

$f(\boldsymbol{x}) = \textbf{No}$



Top-down induction of decision trees

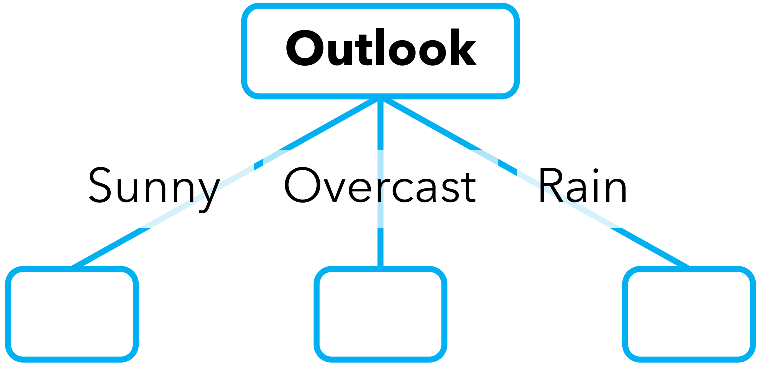
- Grow tree from the root to the leaves
- Repeatedly replacing an existing leaf with an internal node



Top-down induction of decision trees

Main loop:

1. $A \leftarrow$ "best" decision attribute for next node
2. For each value of A, create new descendent of node
3. Sort training examples to leaf nodes

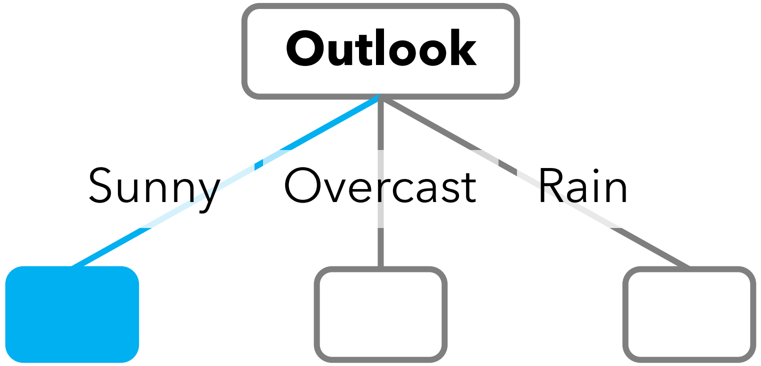


Day	Outlook	Temp	Humidity	Wind	Tennis
D1	Sunny	Hot	Normal	Weak	**Yes**
D2	Sunny	Hot	High	Strong	**No**
D3	Overcast	Hot	High	Weak	**Yes**
D4	Rain	Mild	High	Weak	**No**
D5	Rain	Cool	Normal	Weak	**Yes**

Top-down induction of decision trees

Main loop:

1. $A \leftarrow$ "best" decision attribute for next node
2. For each value of A, create new descendent of node
3. Sort training examples to leaf nodes

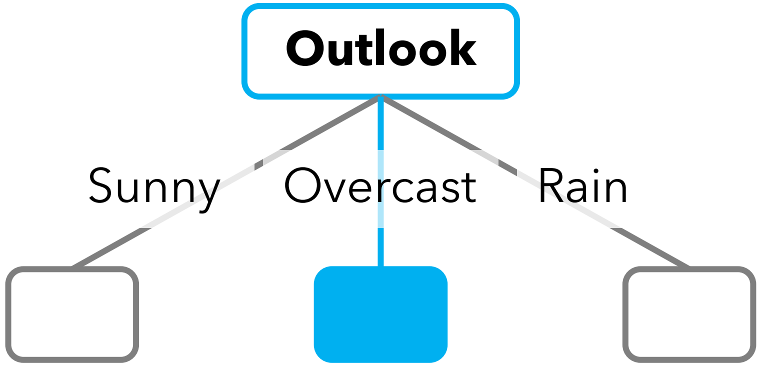


Day	Outlook	Temp	Humidity	Wind	Tennis
D1	Sunny	Hot	Normal	Weak	**Yes**
D2	Sunny	Hot	High	Weak	**No**
D3	Overcast	Hot	High	Weak	**Yes**
D4	Rain	Mild	High	Strong	**No**
D5	Rain	Cool	Normal	Weak	**Yes**

Top-down induction of decision trees

Main loop:

1. $A \leftarrow$ "best" decision attribute for next node
2. For each value of A, create new descendent of node
3. Sort training examples to leaf nodes

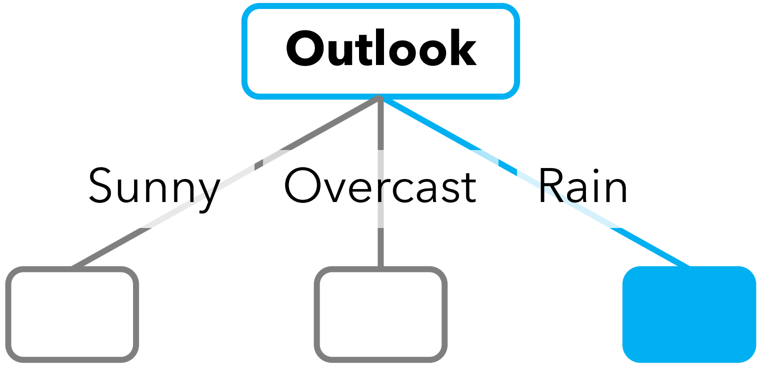


Day	Outlook	Temp	Humidity	Wind	Tennis
D1	Sunny	Hot	High	Weak	**No**
D2	Sunny	Hot	High	Strong	**No**
D3	Overcast	Hot	High	Weak	**Yes**
D4	Rain	Mild	High	Weak	**No**
D5	Rain	Cool	Normal	Weak	**Yes**

Top-down induction of decision trees

Main loop:

1. $A \leftarrow$ "best" decision attribute for next node
2. For each value of A, create new descendent of node
3. Sort training examples to leaf nodes



Day	Outlook	Temp	Humidity	Wind	Tennis
D1	Sunny	Hot	High	Weak	**No**
D2	Sunny	Hot	High	Strong	**No**
D3	Overcast	Hot	High	Weak	**Yes**
D4	Rain	Mild	High	Weak	**No**
D5	Rain	Cool	Normal	Weak	**Yes**

Top-down induction of decision trees

Main loop:

1. $A \leftarrow$ "best" decision attribute for next node
2. For each value of A, create new descendent of node
3. Sort training examples to leaf nodes
4. If training examples perfectly classified, then STOP,
   Else iterate over new leaf nodes

Top-down induction of decision trees

Main loop:

1. $A \leftarrow$ $\boxed{\text{“best”}}$ decision attribute for next node
2. For each value of A, create new descendent of node

Many different heuristics can be used to choose attribute

E.g., entropy (ID3)

Else iterate over new leaf nodes

Outline

1. What is machine learning
2. Regression
3. Classification
   a. Linear classification
   b. Decision trees
   c. **Multi-class classification**
4. (Simple) neural networks

Multiclass classification

Label $y \in \{1, \dots, k\}$ (e.g., digit classification)

Approach 1:

- Build k different binary classifiers $h_{\boldsymbol{\theta}_i}$
- $h_{\boldsymbol{\theta}_i}$ predicts class i vs all others
- Output predictions $\hat{y} = \operatorname*{argmax}_i h_{\boldsymbol{\theta}_i}(\boldsymbol{x})$

Multiclass classification

Label $y \in \{1, \dots, k\}$ (e.g., digit classification)

Approach 2:

- Use a hypothesis function $h_{\boldsymbol{\theta}}: \mathbb{R}^n \to \mathbb{R}^k$
- Define a loss function $\ell: \mathbb{R}^k \times \{1, \dots, k\} \to \mathbb{R}_+$
- E.g., softmax loss (also called cross entropy loss):

$$\ell(h_{\boldsymbol{\theta}}(\boldsymbol{x}), y) = \log \sum_{j=1}^{k} \exp\left(h_{\boldsymbol{\theta}}(\boldsymbol{x})_j\right) - h_{\boldsymbol{\theta}}(\boldsymbol{x})_y$$

Outline

1. What is machine learning
2. Regression
3. Classification
4. **(Simple) neural networks**

Neural networks for machine learning

3 components of ML algorithms:

1. Hypothesis class: set of functions we consider
2. Loss function: measures how good a hypothesis is
3. Optimization: how we find a hypothesis with low loss

Neural network: a type of hypothesis class

Composed non-linear functions

Any loss function and optimization approach could be used

Linear hypotheses and feature learning

Until now, mostly analyzed linear hypotheses $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{x}$

Performance depends on coming up with good features $\boldsymbol{x}$

Key question:

Can we automatically *learn* the features from raw data?

Feature learning, take one

Two-stage hypothesis class where:

1. One linear function creates the features, and
2. Another produces the final hypothesis

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = W_2\phi(\boldsymbol{x}) + \boldsymbol{b}_2 = W_2(W_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2$$

where $\boldsymbol{\theta} = \{W_1 \in \mathbb{R}^{k\times n}, \boldsymbol{b}_1 \in \mathbb{R}^k, W_2 \in \mathbb{R}^{1\times k}, \boldsymbol{b}_2 \in \mathbb{R}\}$

Neural networks

Neural networks are a simple extension of this idea

Apply a **non-linear function** after each linear transformation

$$h_\theta(x) = f_2(W_2 f_1(W_1 x + b_1) + b_2)$$

$f_1, f_2: \mathbb{R} \to \mathbb{R}$ are non-linear functions (applied elementwise)

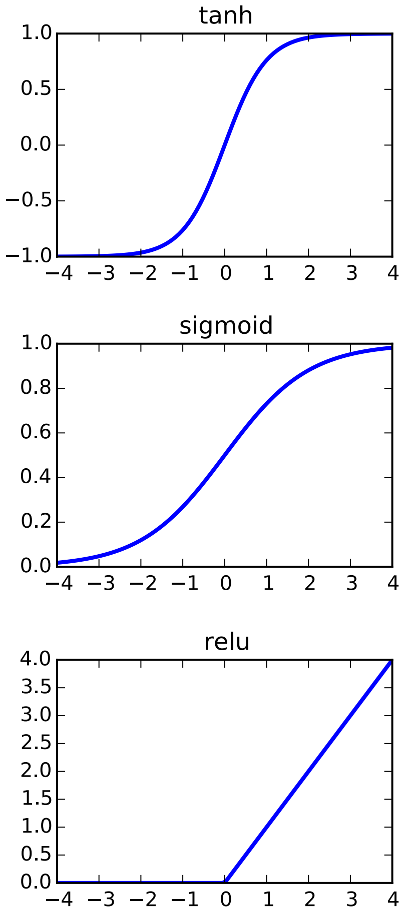
Neural networks

Common choices of f_i:

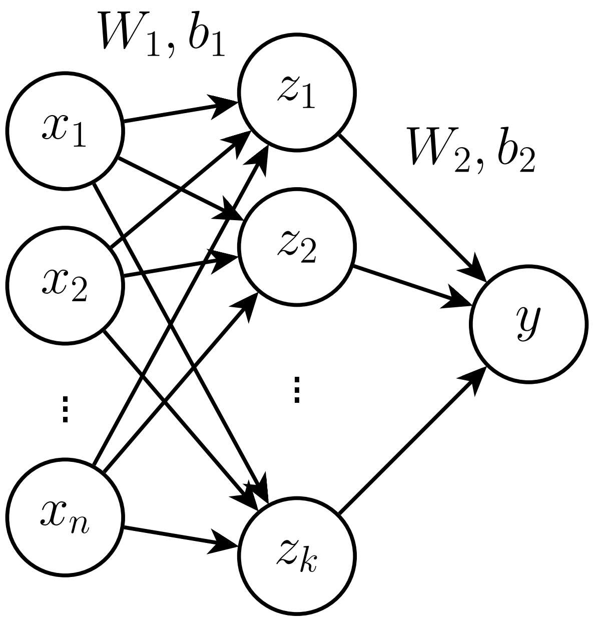
Hyperbolic tangent: $f(x) = \tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$

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

Rectified linear unit (ReLU): $f(x) = \max\{x, 0\}$



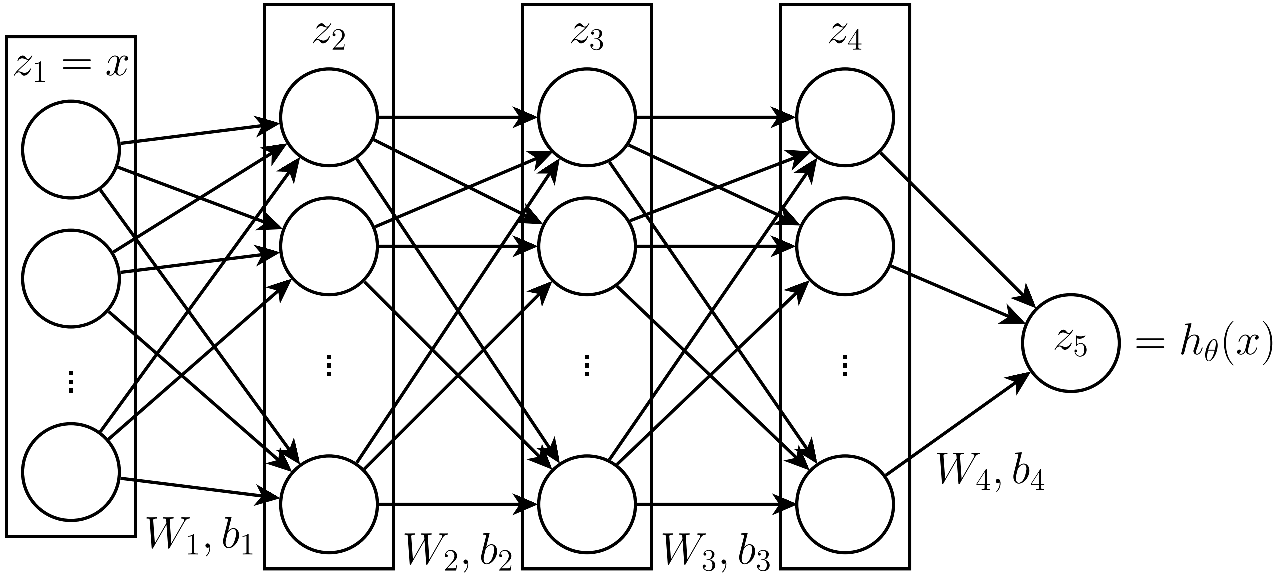
Illustrating neural networks



Middle layer *z* is referred to as the *hidden layer* or *activations*

- These are the **learned features**
- Nothing in the data prescribed what values they should take

Deep learning



"Multi-layer perceptron"

Hypothesis function for k-layer network

$$\boldsymbol{z}_{i+1} = f_i(W_i \boldsymbol{z}_i + \boldsymbol{b}_i), \qquad \boldsymbol{z}_1 = \boldsymbol{x}, \qquad h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{z}_k$$

($\mathbf{z}_i$ here refers to a vector, not an entry in a vector)

Training neural networks

Gradient descent, repeat:

- For $i = 1, \ldots, m$: $\boldsymbol{g}^{(i)} \leftarrow \nabla_{\boldsymbol{\theta}} \ell\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right)$
- Update parameters: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \sum_{i=1}^{m} \boldsymbol{g}^{(i)}$

Stochastic gradient descent (SGD), repeat:

- For $i = 1, \ldots, m$: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \ell\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right)$

In practice, SGD uses a small “minibatch” of samples

Overview

1. What is machine learning?
2. Regression
3. Classification
4. (Simple) neural networks

Next class:

- Linear programming relaxations
- Integer programming solvers
- SAT solvers