UNIVERSITÄT
BERN

# Machine Learning Review

Paolo Favaro

# Contents

- Revision of basic concepts of Machine Learning
- Based on **Chapter 5** of Deep Learning by Goodfellow, Bengio, Courville

# Context

- A more complete introduction to Machine Learning through the following courses
  - Machine Learning @ UniBe
  - Machine Learning and Data Mining @ UniNe
  - Pattern Recognition @ UniFr
  - Statistical Learning Methods @ UniNe

# Resources

- Books and online material for further studies
  - Machine Learning @ Stanford (Andrew Ng)
  - **Pattern Recognition and Machine Learning**
    by Christopher M. Bishop
  - **Machine Learning: a Probabilistic Perspective** by Kevin P. Murphy

# Learning Pillars

- Supervised learning
- Semi-supervised learning
- Self-taught learning (unsupervised feature learning)
- Unsupervised learning
- Reinforcement learning

# Definition

- Mitchell (1997)
  *A computer program is said to learn from* ***experience E*** *with respect to some class of* ***tasks T*** *and* ***performance measure P****, if its performance at tasks in T, as measured by P, improves with experience E.*

# The Task T

- Example: if we want a robot to be able to walk, then **walking** is the task
- Approaches

1. We could directly input directives for how we think a robot should walk, or
2. We could provide examples of successful and unsuccessful walking (this is machine learning)

# The Task T

- Given an input x (e.g., a vector) produce a function f, such that f(x) = y (e.g., an integer, a probability vector)
- Examples
  - Classification
  - Regression
  - Machine translation
  - Denoising
  - Probability density estimation

# The Performance Measure P

- To evaluate a ML algorithm we need a way to measure how well it performs on the task
- It is measured on a separate set (**the test set**) from what we use to build the function f (**the training set**)
- Examples
  - Classification accuracy (portion of correct answers) or error rate (portion of incorrect answers)
  - Regression accuracy (e.g., least squares errors)

# The Experience E

- Specifies what data can be used to solve the task
- We can distinguish it based on the learning pillars
  - **Supervised**: data is composed of both the input x (e.g., features) and output y (e.g., labels/targets)
  - **Unsupervised**: data is composed of just x; here we typically aim for p(x) or a method to sample p(x)
  - **Reinforcement**: data is dynamically gathered based on previous experience

# Data

- We assume that all collected data samples in all datasets:

  1. come from the same distribution $\rightarrow p_{x^{(i)}}(x) = p_{x^{(j)}}(x)$

  2. are independent $\longrightarrow p\left(x^{(1)}, \ldots, x^{(m)}\right) = \prod_{i=1}^{m} p\left(x^{(i)}\right)$

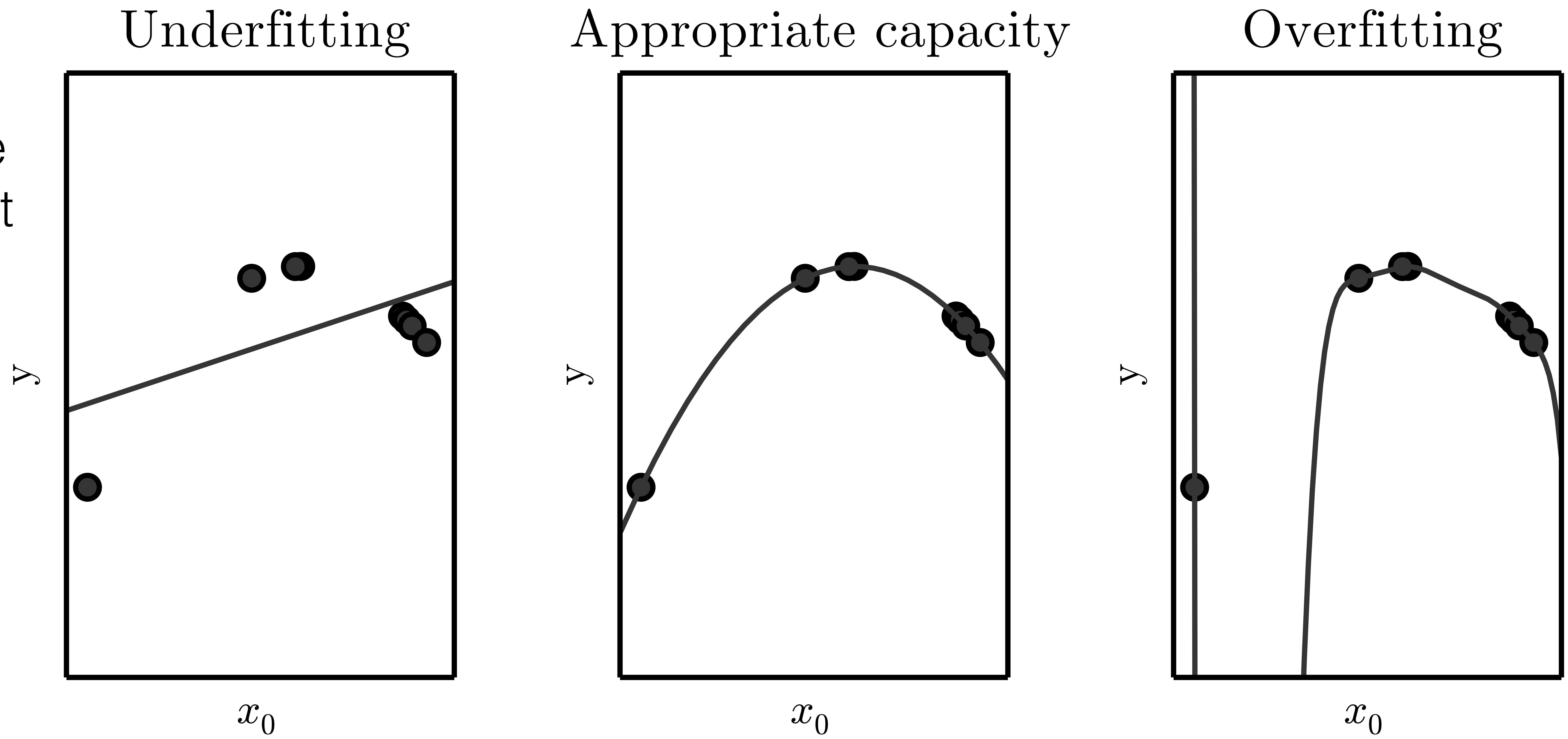
- This assumption is denoted **IID** (independent and identically distributed)

# Overfitting and Underfitting

- Performance P captures how well the learned model predicts new unseen data
- Ideally we want to select the predictor with the best performance
- What happens when we use predictors of different complexity/capacity?

# Overfitting and Underfitting

shown data is the training set




simple predictor | optimal predictor | complex predictor

# Loss function

- Define a **predictor** function $f : \mathcal{X} \mapsto \mathcal{Y}$
- Define a **loss** function $l : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ which measures how different the two inputs are
- Examples
  - 0-1 loss $$l(y, f(x)) = \begin{cases} 0 \text{ if } y = f(x) \\ 1 \text{ if } y \neq f(x) \end{cases}$$
  - Quadratic loss $$l(y, f(x)) = (y - f(x))^2$$

# Bayes Risk

- **Bayes risk** is defined as (average loss)

$$R(f) = E_{x,y}[l(f(x), y)] = \int l(f(x), y)p(x, y)dxdy$$

- The optimal predictor function is

$$f^* = \arg\min_f R(f)$$

# Empirical Risk

- Given $(x_i,y_i)$ with i = 1,…,m the **empirical risk** is

$$\hat{R}(f) = \frac{1}{m}\sum_{i=1}^{m} l(f(x_i), y_i)$$

- The empirical predictor is

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \hat{R}(f)$$

# Training, Validation and Test

- In alternative, collect samples into training set $D_{\text{train}}$, validation set $D_{\text{val}}$ and test set $D_{\text{test}}$

- Use the **training set** to define the optimal predictor $\hat{f}_\lambda = \arg\min_{f \in \mathcal{F}} \hat{R}_{D_{\text{train}}}(f)$

- Use the **validation set** to choose the capacity $\hat{\lambda} = \arg\min_{\lambda} \hat{R}_{D_{\text{val}}}(\hat{f}_\lambda)$

- Use the **test set** to evaluate the performance $\text{performance P} = R_{D_{\text{test}}}\left(\hat{f}_{\hat{\lambda}}\right)$

# Maximum Likelihood

- Given IID data samples $x^1, \dots, x^m \sim p_{\text{data}}(x)$
- Let $p_{\text{model}}(x;\theta)$ be a parametric family of probability density functions
- The **maximum likelihood estimator** of $\theta$ is

$$\begin{aligned}\theta_{\text{ML}} &= \arg\max_\theta \textstyle\prod_{i=1}^m p_{\text{model}}(x^i;\theta) \\ &= \arg\max_\theta \textstyle\sum_{i=1}^m \log p_{\text{model}}(x^i;\theta) \\ &= \arg\max_\theta E_{x\sim\hat{p}_{\text{data}}}[\log p_{\text{model}}(x;\theta)]\end{aligned}$$

with the empirical data distribution $\hat{p}_{\text{data}}(x)$

# Maximum Likelihood

- Maximum likelihood estimation can also be interpreted as fitting $p_{\text{model}}(x;\theta)$ to $p_{\text{data}}(x)$ via a Kullback-Leibler divergence minimization

$$\begin{aligned}
&\arg\max_{\theta} -D_{\text{KL}}(\hat{p}_{\text{data}}|p_{\text{model}}) \\
=&\arg\max_{\theta} -E_{x\sim\hat{p}_{\text{data}}}[\log\hat{p}_{\text{data}}(x) - \log p_{\text{model}}(x;\theta)] \\
=&\arg\max_{\theta} E_{x\sim\hat{p}_{\text{data}}}[\log p_{\text{model}}(x;\theta)]
\end{aligned}$$

# Maximum Likelihood

- Given IID input/output samples $(x^i, y^i) \sim p_{\text{data}}(x, y)$

  the conditional maximum likelihood estimate is

$$\begin{aligned}\theta_{\text{ML}} &= \arg\max_\theta \prod_{i=1}^{m} p_{\text{data}}(y^i | x^i; \theta) \\ &= \arg\max_\theta \sum_{i=1}^{m} \log p_{\text{data}}(y^i | x^i; \theta)\end{aligned}$$

# Maximum a Posteriori

- Given IID data samples $x^1, \ldots, x^m \sim p_{\text{data}}(x)$
- Let $p(x^1, \ldots, x^m|\theta)$ be the conditional probability density function
- Maximum a Posteriori aims at recovering

$$p(\theta|x^1, \ldots, x^m) = \frac{p(x^1, \ldots, x^m|\theta)p(\theta)}{p(x^1, \ldots, x^m)}$$

# Maximum a Posteriori

- The prior $p(\theta)$ can encode a preference for simpler or smoother models (acts as regularizer)

- Predictions could be obtained by marginalizing over the parameters (this corresponds to a quadratic loss function in Bayes risk)

$$p(x^{m+1}|x^1, \ldots, x^m) = \int p(x^{m+1}|\theta)p(\theta|x^1, \ldots, x^m)d\theta$$

# Maximum a Posteriori

- If we minimize Bayes risk with a 0-1 loss function, we obtain a point estimate

$$\theta_{\mathrm{MAP}} = \arg\max_{\theta} p(\theta|x) = \arg\max_{\theta} \log p(x|\theta) + \log p(\theta)$$

which is computationally feasible.

- This is the **Maximum a Posteriori (MAP)** estimate

# Supervised Learning

- Make a prediction of an output $y$ given an input $x$
- Boils down to determining the conditional probability

$$p(y|x)$$

- Formulate problem as that of finding $\theta$ for a parametric family (Maximum Likelihood)

$$p(y|x;\theta)$$

# Supervised Learning

- **Example**: Binary classification $y \in \{0, 1\}$

- We aim at determining $p(y=1|x;\theta) = \sigma(\theta^\top x)$

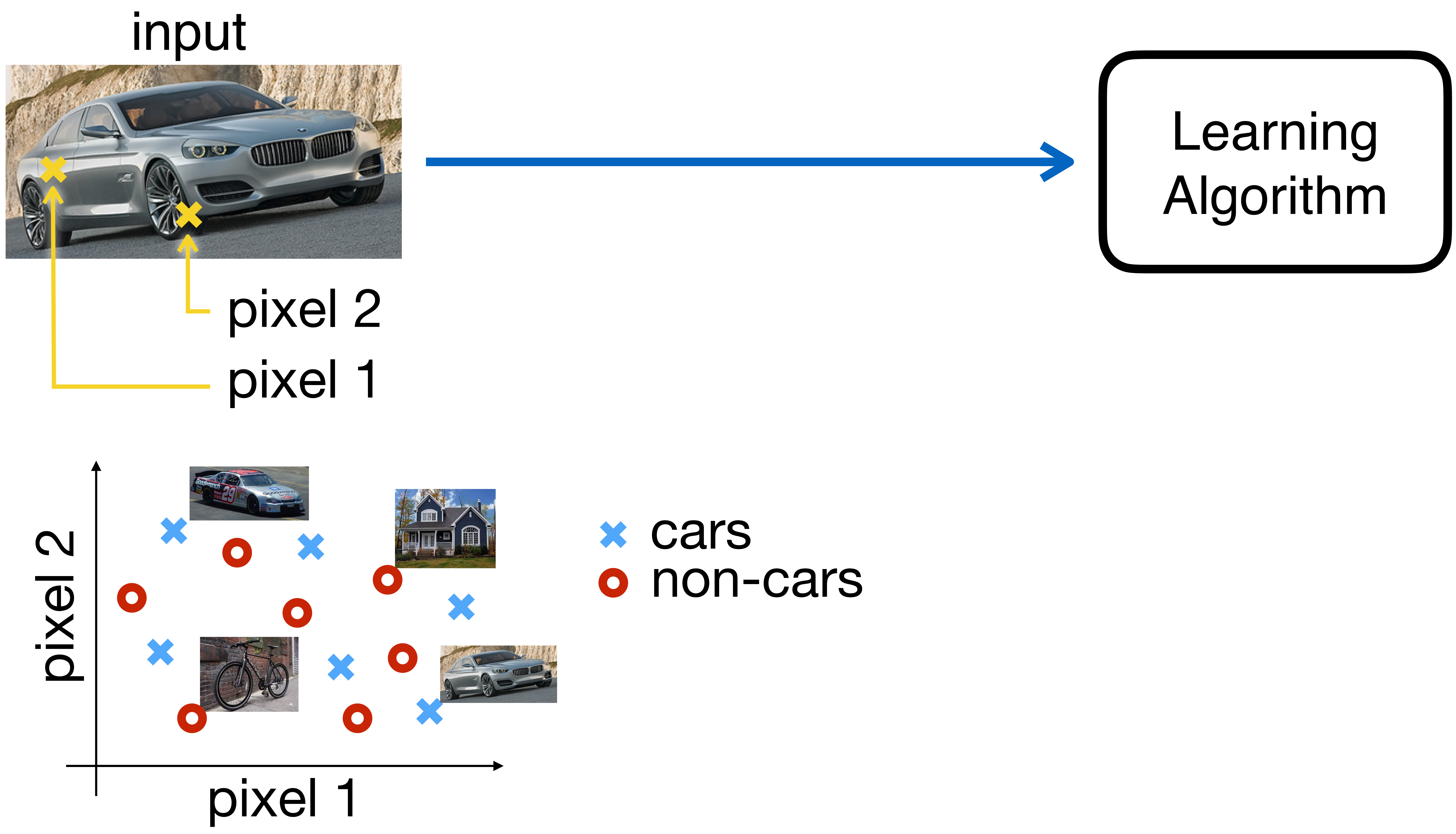
  where $\sigma(z) = \dfrac{1}{1+e^{-z}}$ is the sigmoid function

- Class y=1 can be picked when

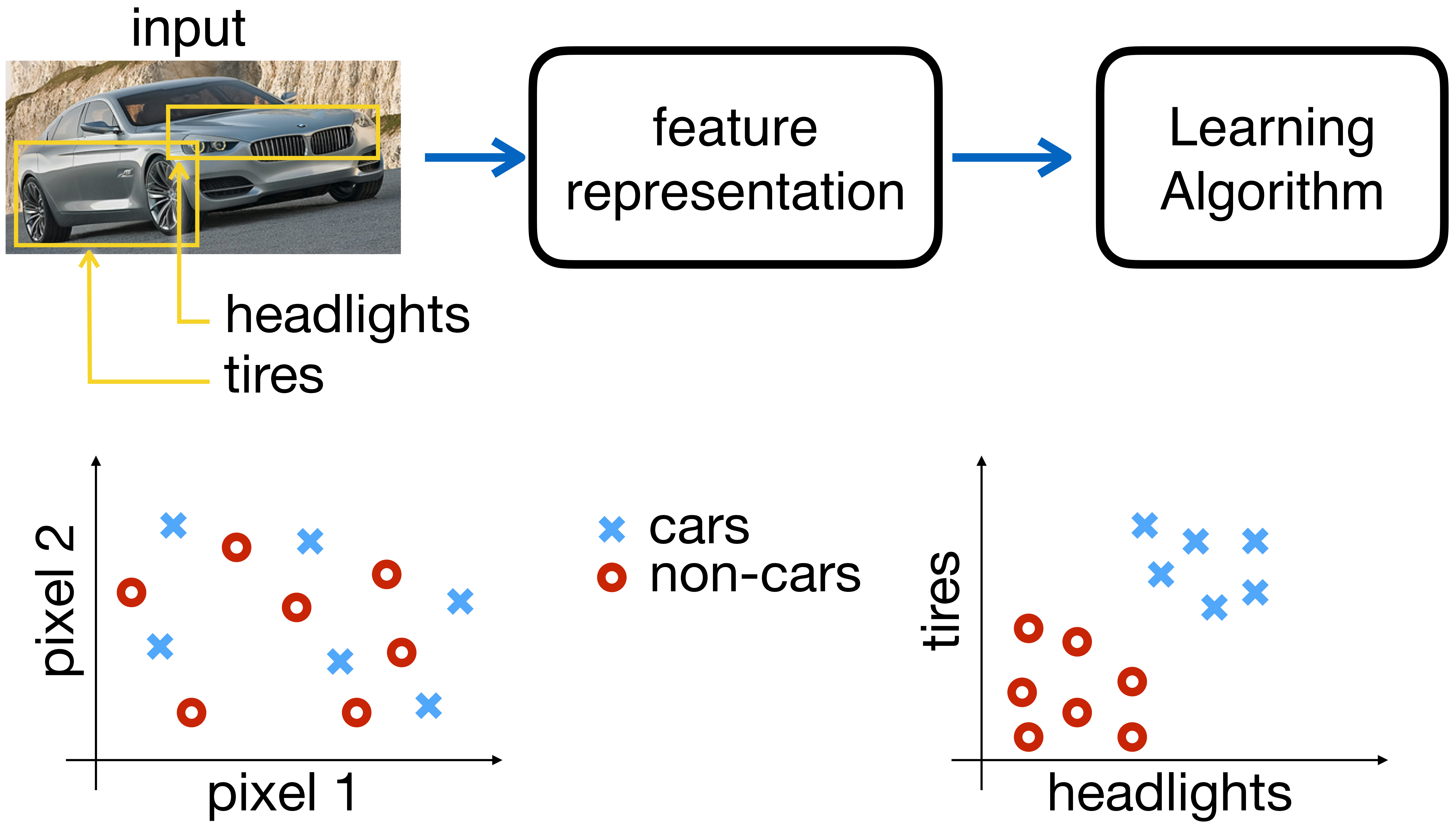
$$p(y=1|x;\theta) > p(y=0|x;\theta)$$

  which is equivalent to $\theta^\top x > 0$

# Features


input
Learning
Algorithm
pixel 2
pixel 1
pixel 2
pixel 1
cars
non-cars

# Features


input
headlights
tires
feature representation
Learning Algorithm
pixel 2
pixel 1
cars
non-cars
tires
headlights

# Building a
# Machine Learning Algorithm

1. Build a dataset
2. Define a model
3. Define a cost function
4. Define an optimization procedure

# Example (revisited)

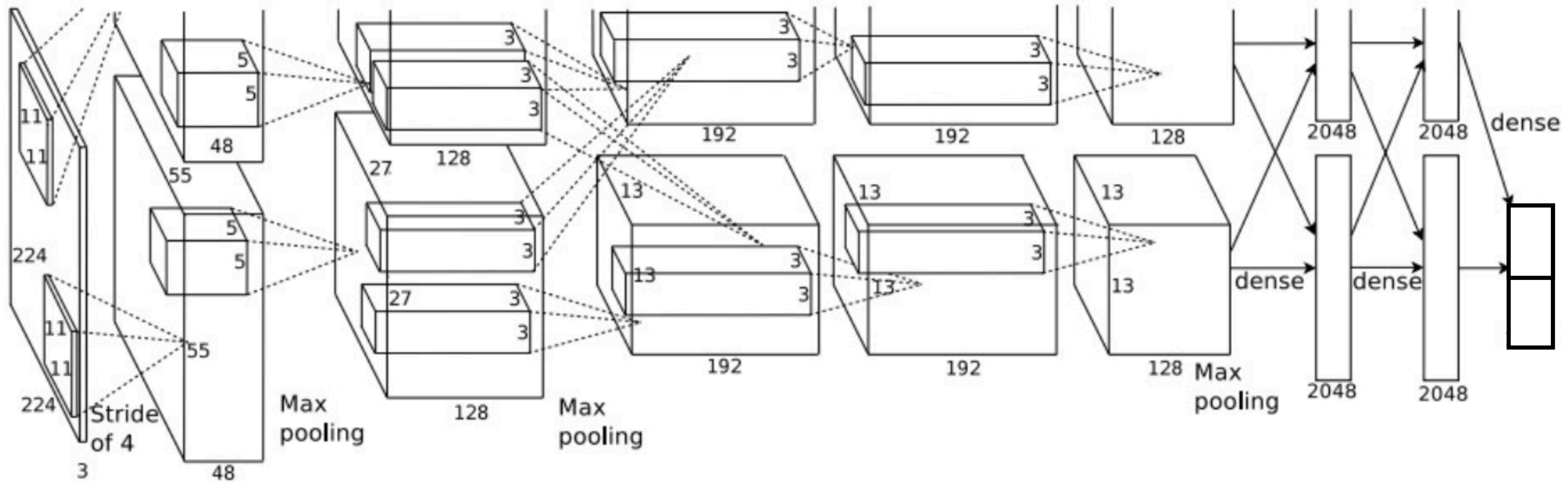
- **Dataset** of cars and non-cars (>1M samples)

| | | |
|---|---|---|
| $x$ |   |   |
| | $y = 1$ (cars) | $y = 0$ (non-cars) |

# Example (revisited)

- **Model**

$$p(y|x;\theta) =$$




binary
output

convolutional neural network

# Example (revisited)

- **Cost function**

$$\sum_{i=1}^{m} \log p(y^i | x^i; \theta)$$

negative cross entropy (maximum likelihood)

# Example (revisited)

- **Optimization procedure**

$$\theta_{t+1} = \theta_t + \alpha_t \frac{\nabla_\theta p(y^i|x^i;\theta_t)}{p(y^i|x^i;\theta_t)} \qquad i \sim U[1,m]$$

stochastic gradient ascent