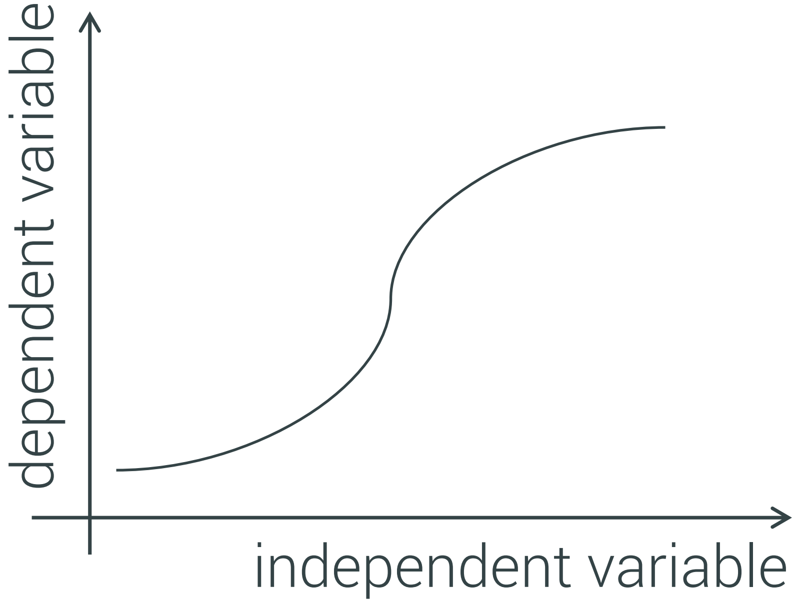
End-to-end machine learning

Lecture 02

Quiz

Common language



independent variable	**dependent** variable
input	output
predictor	response
feature	target
x	y

Supervised learning

Objective: create an algorithm that predicts well

Example:
Digits classification

New Data

$\boldsymbol{x}_{new}$

Trained Algorithm

$\hat{f}(\boldsymbol{x})$

Prediction

$\hat{y}_{new}$

Training Data
(examples)

$(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)$

$\boldsymbol{x}_i$:

y_i: 3 3 3 3 7 9 6 1 2 4 3 3

Supervised learning

Objective: create an algorithm that predicts well

Example:
Digits classification

New Data

$\boldsymbol{x}_{new}$

Trained Algorithm

$\hat{f}(\boldsymbol{x})$

Prediction

$\hat{y}_{new}$

Training Data

(examples)

$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$

$\boldsymbol{x}_i$:

y_i: 1 1 1 1 0 0 0 0 0 0 0 0

Simplification for example:
Is the digit a "3"?

This becomes a binary classification problem:
$y = 1$ implies a "3"
$y = 0$ implies another digit

Supervised machine learning model

We search for the model that best fits our data

$$y = f(x, \boldsymbol{w}) + \epsilon$$

Labels

Model

Input Data

Parameter(s)

Random Noise

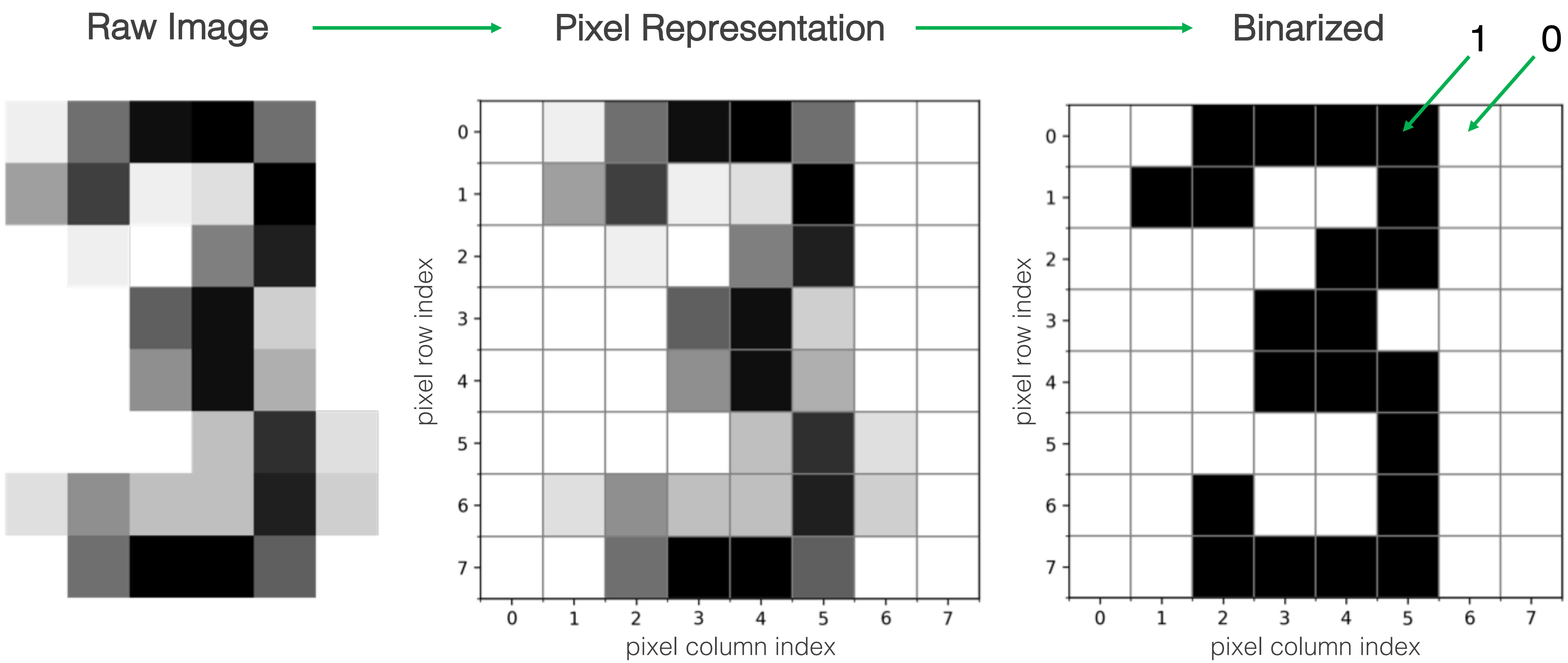
Hypothesis

Typically prevents perfect performance

Preprocessing

prepare your data for analysis

How do we make a prediction rule?

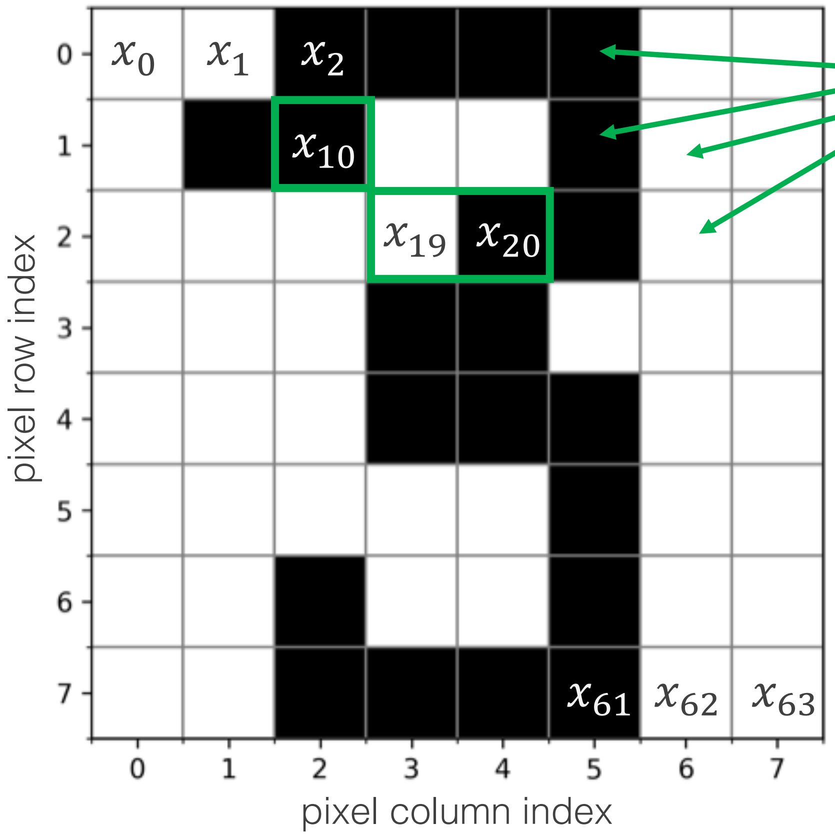


Choose your model

Which prediction rules will you consider?

How do we make a prediction rule?

$\boldsymbol{x}_i$: one sample (observation) of the data



Each pixel represents an element of $\boldsymbol{x}_i = [x_0, x_1, \ldots, x_{63}]$

64-dimensional binary data

We want to estimate a function that takes these pixel values and decides whether this is a '3' or not

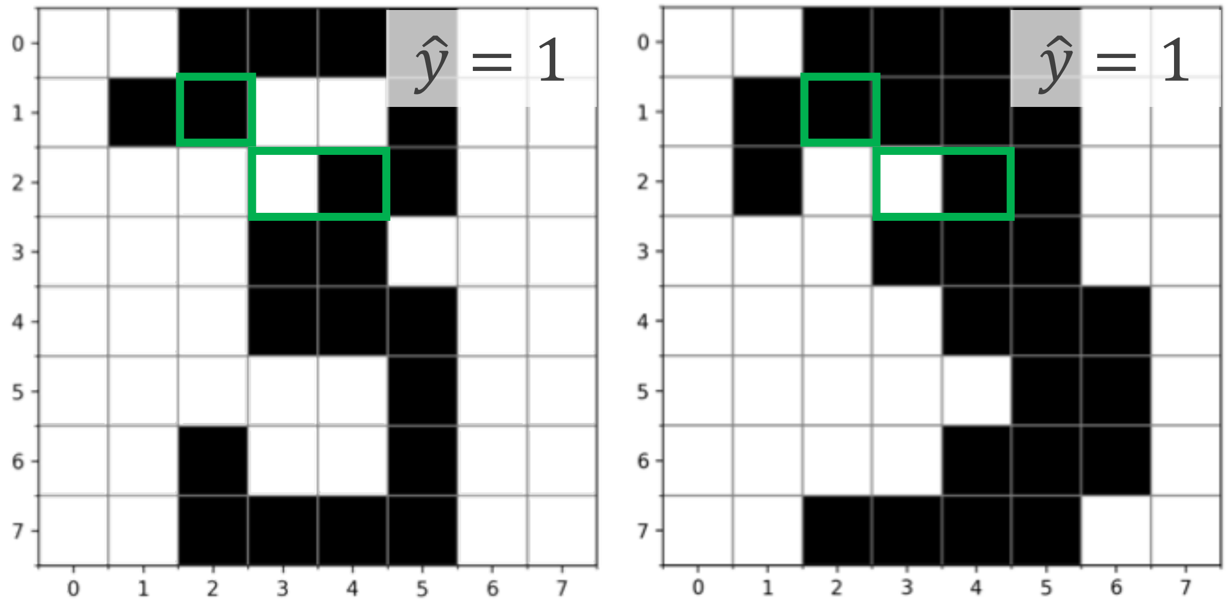
Example decision rule / prediction algorithm:

$$f(\boldsymbol{x}) = \begin{cases} 1 & x_{10}x_{20}(1 - x_{19}) > 0 \\ 0 & else \end{cases}$$

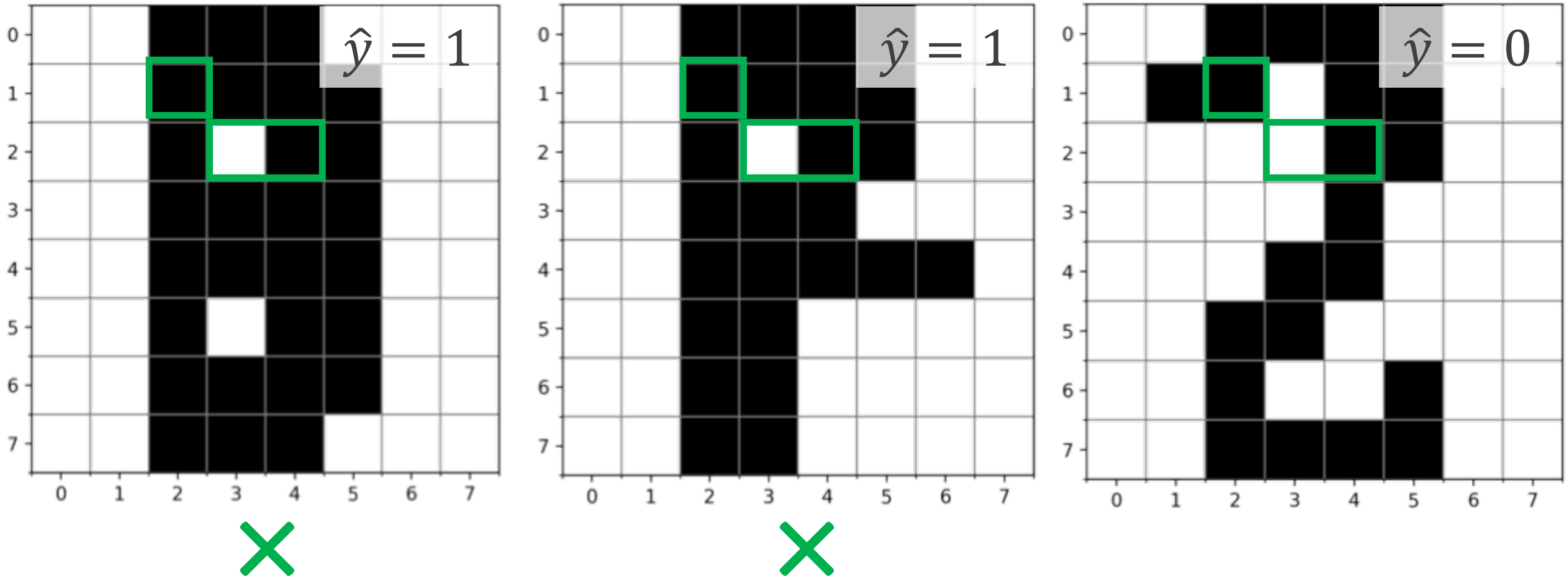
Does the rule work on other examples?

$$\hat{f}(\boldsymbol{x}) = \begin{cases} 1 & x_{10}x_{20}(1 - x_{19}) > 0 \\ 0 & else \end{cases}$$

$y = 1$

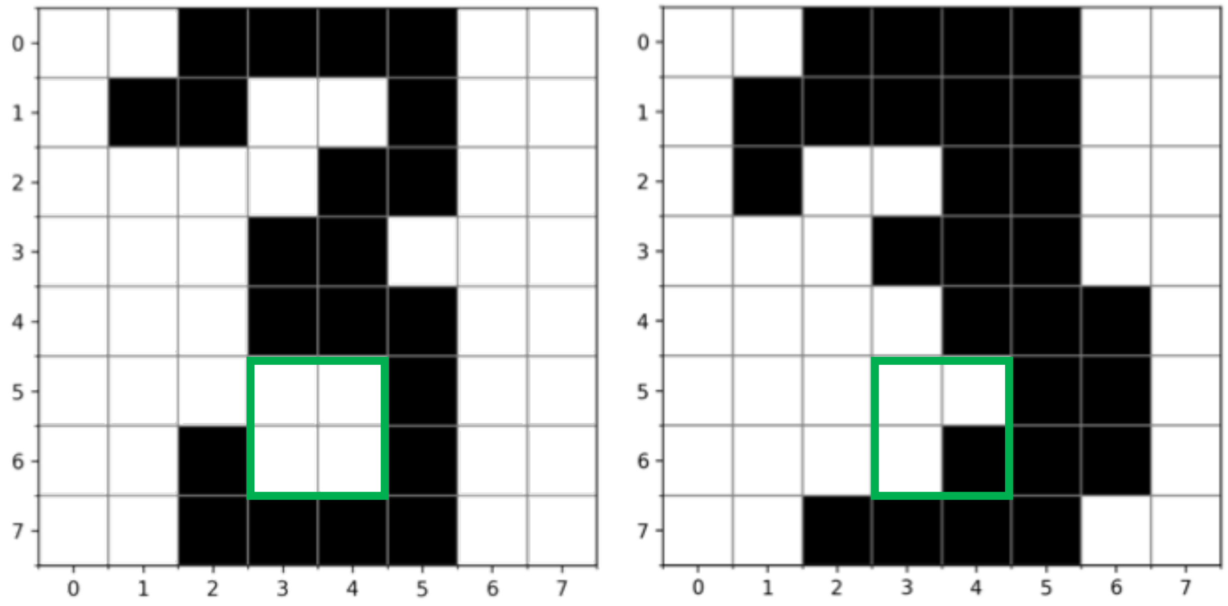


$y = 0$

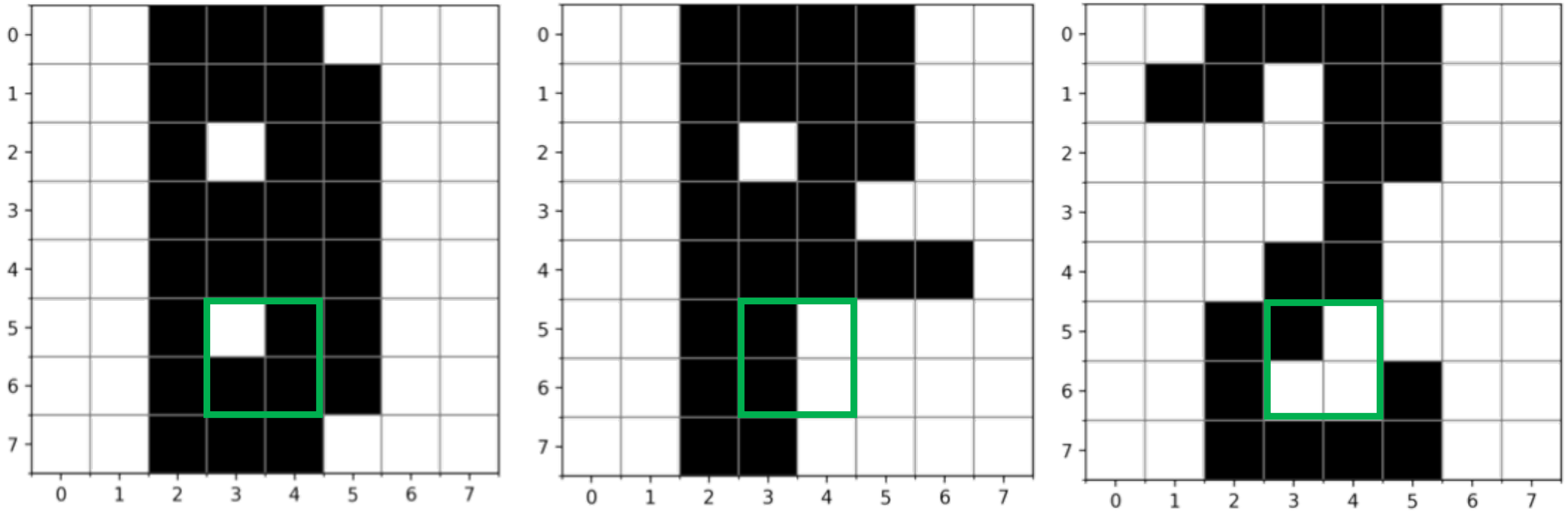


This is very hard to do manually, let's limit our search to a **subset** of the features…

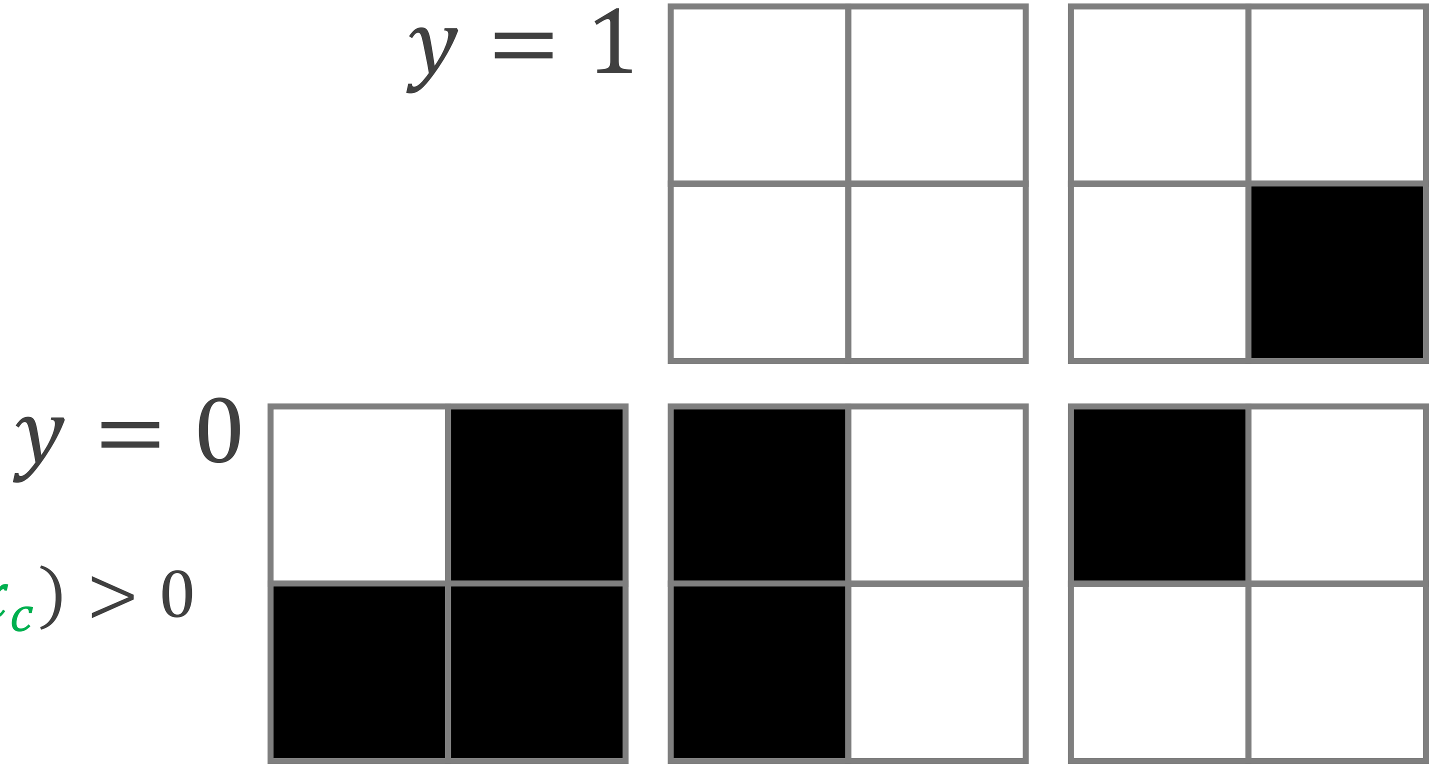
$y = 1$



$y = 0$



What decision rules work?



Option 1:

$$\hat{f}_1(\boldsymbol{x}) = \begin{cases} 1 & (1 - x_a)(1 - x_b)(1 - x_c) > 0 \\ 0 & else \end{cases}$$

Option 2:

$$\hat{f}_2(\boldsymbol{x}) = \begin{cases} 1 & (1 - x_a)(1 - x_b) > 0 \\ 0 & else \end{cases}$$

For simplicity, we refer to these x_i values as:

x_a	x_b
x_c	x_d

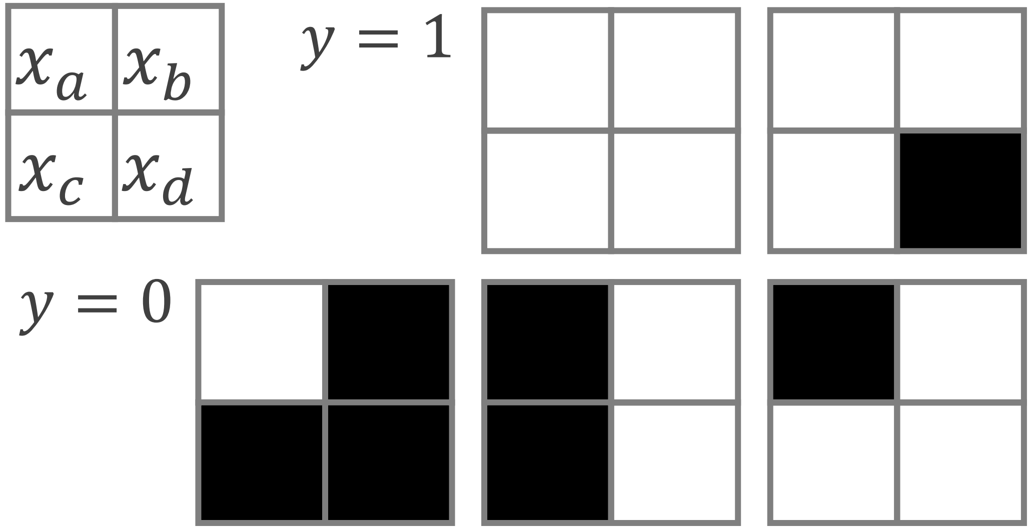
Option 3 (exclude our negative examples):

$$\hat{f}_3(\boldsymbol{x}) = \begin{cases} 1 & 1 - [(1 - x_a)x_b x_c x_d + x_a(1 - x_b)x_c(1 - x_d) + x_a(1 - x_b)(1 - x_c)(1 - x_d)] > 0 \\ 0 & else \end{cases}$$

Learning: train your model

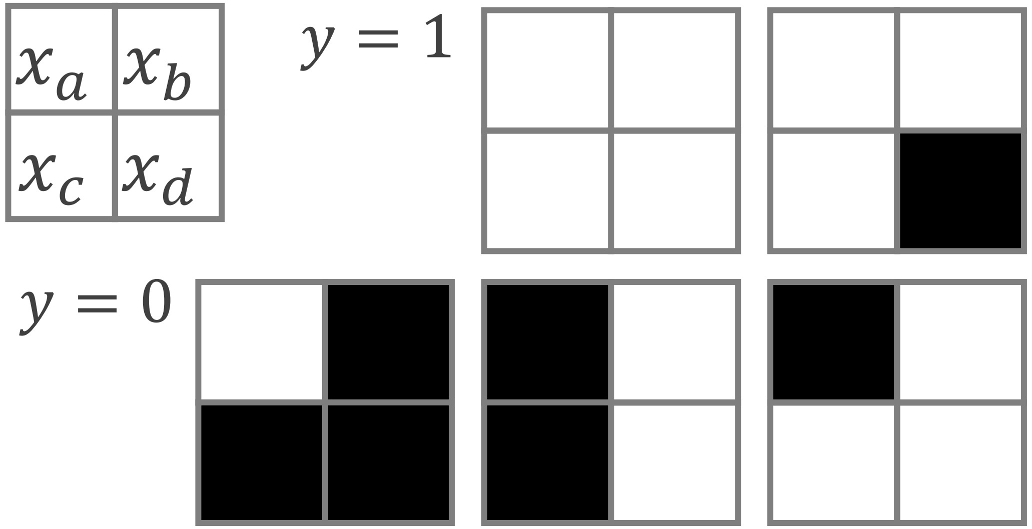
Select the best rule from the options available

x_a	x_b	x_c	x_d	y
0	0	0	0	**1**
0	0	0	1	**1**
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	**0**
1	0	0	0	**0**
1	0	0	1	
1	0	1	0	**0**
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	



What are all the possible combinations?

x_a	x_b	x_c	x_d	y	$\hat{f}_1$	$\hat{f}_2$	$\hat{f}_3$
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1
0	0	1	0		0	1	1
0	0	1	1		0	1	1
0	1	0	0		0	0	1
0	1	0	1		0	0	1
0	1	1	0		0	0	1
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	1		0	0	1
1	0	1	0	0	0	0	0
1	0	1	1		0	0	1
1	1	0	0		0	0	1
1	1	0	1		0	0	1
1	1	1	0		0	0	1
1	1	1	1		0	0	1



Option 1:

$$\hat{f}_1(\boldsymbol{x}) = 1 \text{ if } (1 - x_a)(1 - x_b)(1 - x_c) > 0$$

Option 2:

$$\hat{f}_2(\boldsymbol{x}) = 1 \text{ if } (1 - x_a)(1 - x_b) > 0$$

Option 3 (exclude our negative examples):

$$\hat{f}_3(\boldsymbol{x}) = 1 \text{ if } 1 - [(1 - x_a)x_b x_c x_d + x_a(1 - x_b)x_c(1 - x_d) + x_a(1 - x_b)(1 - x_c)(1 - x_d)] > 0$$

Features: $n = 4$

Decision rules: $N = 2^{2^n} = 65{,}536$

Possible observations $= 2^n = 16$

x_a	x_b	x_c	x_d	y	$\hat{h}_0$	$\hat{h}_1$	$\hat{h}_2$	…	$\hat{h}_{N-1}$	$\hat{h}_N$
0	0	0	0	**1**	0	1	0		1	1
0	0	0	1	**1**	0	0	1		1	1
0	0	1	0		0	0	0		1	1
0	0	1	1		0	0	0		1	1
0	1	0	0		0	0	0		1	1
0	1	0	1		0	0	0		1	1
0	1	1	0		0	0	0		1	1
0	1	1	1	**0**	0	0	0		1	1
1	0	0	0	**0**	0	0	0		1	1
1	0	0	1		0	0	0		1	1
1	0	1	0	**0**	0	0	0		1	1
1	0	1	1		0	0	0		1	1
1	1	0	0		0	0	0		1	1
1	1	0	1		0	0	0		1	1
1	1	1	0		0	0	0		1	1
1	1	1	1		0	0	0		0	1

We have $n = 4$ features: x_a, x_b, x_c, x_d

They produce $2^n = \mathbf{16}$ **possible observations** (i.e. unique feature combinations)…

…yielding $N = 2^{2^4} = \mathbf{65{,}536}$ **unique functions** to predict y!

There are 5 labeled observations, so 16 – 5 = 11 unlabeled observations. So there are $2^{11} = \mathbf{2{,}048}$ **unique functions that all correctly predict** $\boldsymbol{y}$ for our 5 observations

We cannot try all possible decision rules for features…

Number of binary features	Unique Decision Rules
1	$2^{2^1} = 4$
2	$2^{2^2} = 16$
3	$2^{2^3} = 256$
4	$2^{2^4} = 65{,}536$
5	$2^{2^5} = 4{,}294{,}967{,}296$
6	$2^{2^6} = 18{,}446{,}744{,}073{,}709{,}551{,}616$
7	$2^{2^7} = 340{,}282{,}366{,}920{,}938{,}463{,}463{,}374{,}607{,}431{,}768{,}211{,}456$

1 year of processing on the world's most powerful computer

For our digits data, we have **64 features**…

For continuous features, the number of unique rules is **infinite**

Key Challenges

Possible Solutions

Manually selecting rules is typically impractical

Supervised learning searches a **subset** of all possible decision rules to “learn” (choose) a rule

We **cannot test all possible decision rules**

How do we choose the best rule given the data?

Pick the one that **generalizes** best – often **simpler** (regularized) rules. This may mean allowing for some mistakes (tradeoff).

Evaluate performance

See how well your learned model works on new data (i.e. generalizes)

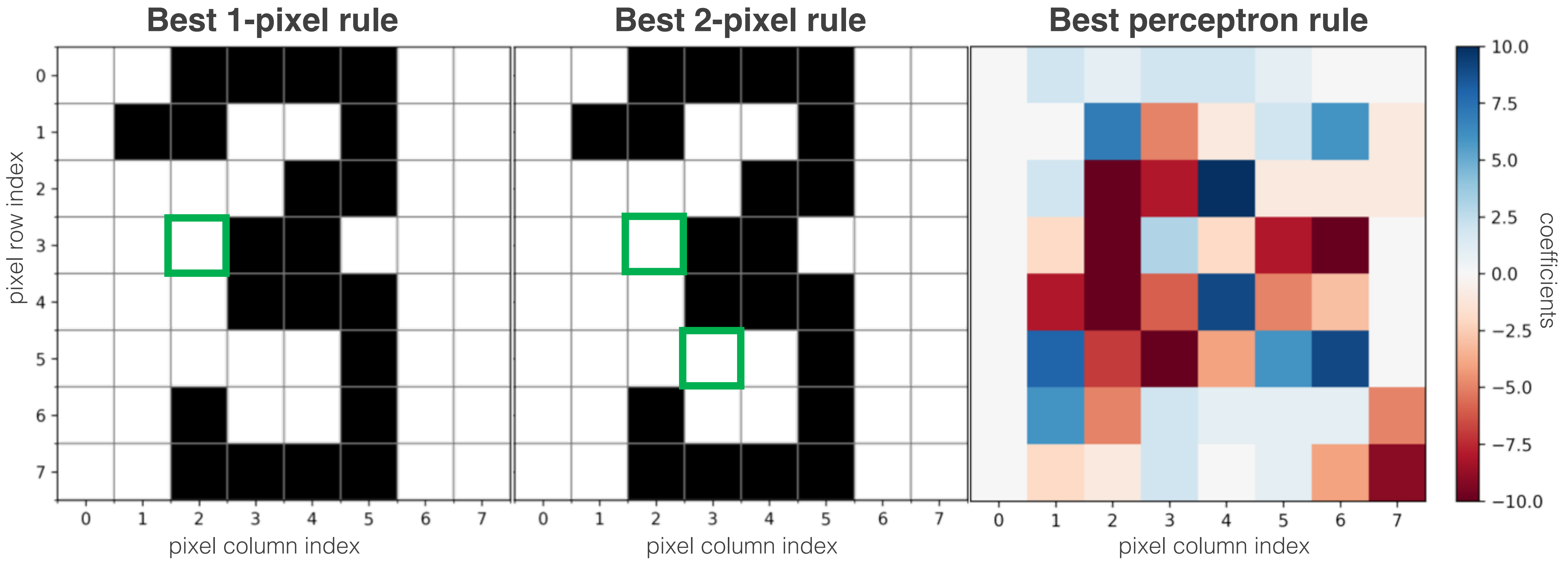
Let's test some rules

We divide the data we have into **training** and **validation** sets so we **never** train and test on the same data

We limit our search for rules to three **hypothesis sets** (possible rules to search) using different **learning algorithms** (search for best rule)

	Hypothesis sets (of rules)	Learning algorithm	Max # of features in rule
1.	The best **one-pixel** rule	exhaustive search	1
2.	The best **two-pixel** rule	exhaustive search	2
3.	A **perceptron** rule (linear classifier)	stochastic gradient descent	64
	models	**optimization strategy**	

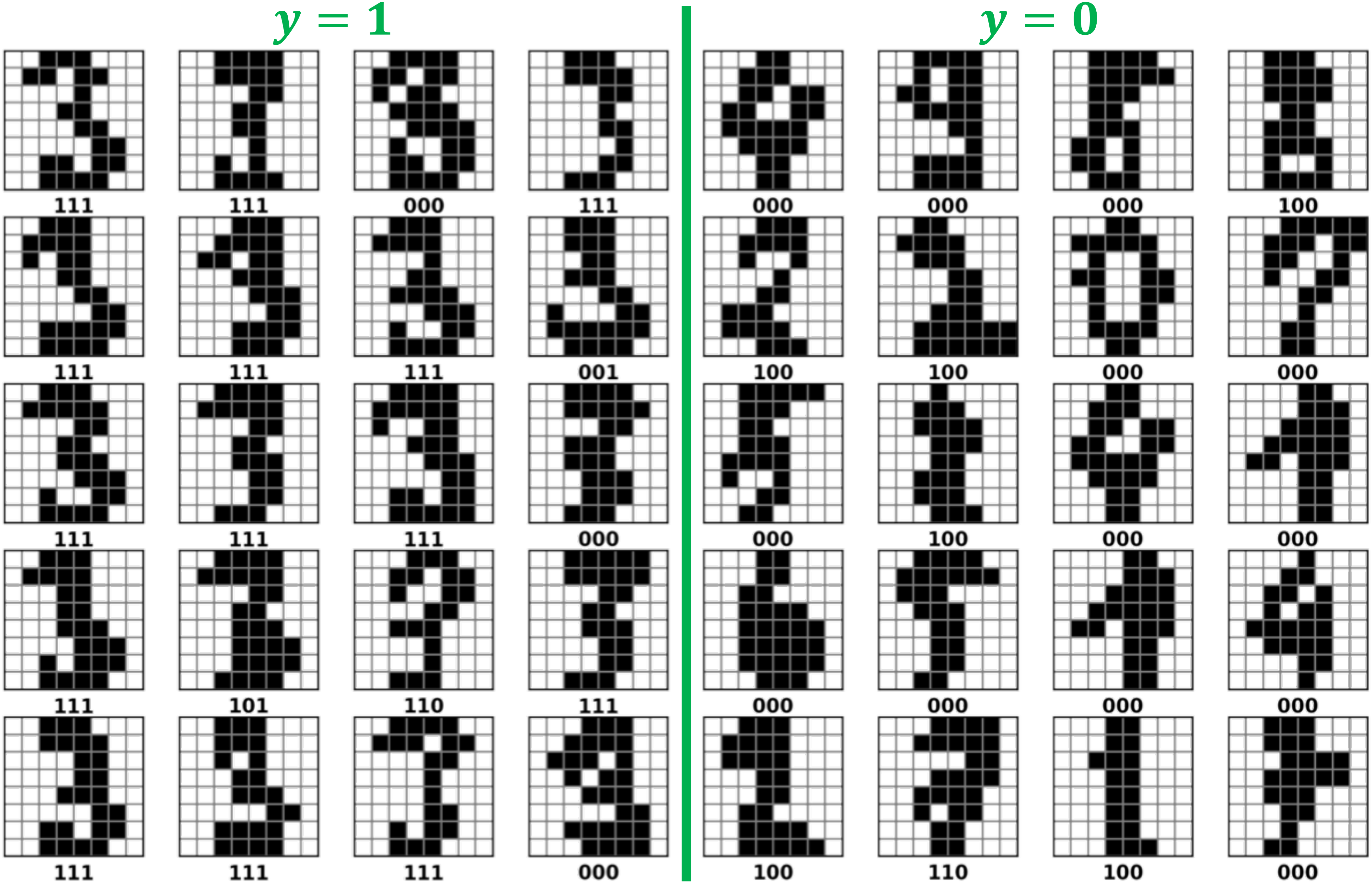
Our results:



Validation performance: **F_1-score** (higher is better)

0.400	**0.691**	**0.844**

Example Results



1-pixel rule

2-pixel rule

perceptron

010 Key

Summarizing supervised learning

Let’s review and bring it all together

Components of supervised learning

Input	$\boldsymbol{x}$	
Output	y	
Training Data	$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)$	
Target function	$f(\boldsymbol{x}) \rightarrow y$	This is unknown, but the best you could ever do
Hypothesis set	$f_i(\boldsymbol{x}) \rightarrow \hat{y}$	Functions to consider in trying to approximate $f(x)$
Learning algorithm	Optimization technique that searches the hypothesis set for the function f_i that best approximates f (typically by choosing parameters in a model)	

Supervised Learning

Unobservable

Observable

Data Generating Process

$p(X, Y)$

Training Data

$(x_1, y_1), \ldots, (x_N, y_N)$

Learning Algorithm
Chooses a hypothesis, $\hat{f} = f_i$ based on the training data such that $\hat{f}(x) \approx f(x)$

Final Hypothesis

$\hat{f}(x) \rightarrow \hat{y}$

predictions

Target Function
The best function predicting y from x

$f(x) \rightarrow y$

Hypothesis Functions Set

$f_1, f_2, f_3, \ldots$

- Need to select the hypothesis functions (models to train)
- Need to select the learning algorithm (for fitting the models to the data)

Supervised learning in practice

Preprocess

Data Visualization and Exploration

Identify patterns that can be leveraged for learning

Data Cleaning

- Missing data
- Noisy data
- Erroneous data

Normalization

Prepare data for use in scale-dependent algorithms.

Feature Extraction

Dimensionality reduction eliminates redundant information

Learning the Model

Training

Select the "best" hypothesis function by choosing model parameters

Apply the Model

Prediction

Predict a categorical (classification) or numerical (regression) target function

Evaluate Performance

Cross-Validation

Metrics

Classification
Precision, Recall, F_1, ROC Curves (Binary), Confusion Matrices (Multiclass)

Regression
MSE, explained variance, R^2

References

Further reading:

Abu-Mostafa, Yaser S., Malik Magdon-Ismail, and Hsuan-Tien Lin. Learning from data. Vol. 4. New York, NY, USA:: AMLBook, 2012.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. Vol. 1. New York: Springer series in statistics, 2001.

Moore, Cristopher, and Stephan Mertens. The nature of computation. OUP Oxford, 2011.

Videos/presentations that inspired this lecture:

Learning from Data, Yaser Abu-Mostafa, Caltech

Learning to See, Welch Labs