Week 1: Visual Recognition & Machine Learning

CMPUT 328 – Nilanjan Ray
University of Alberta

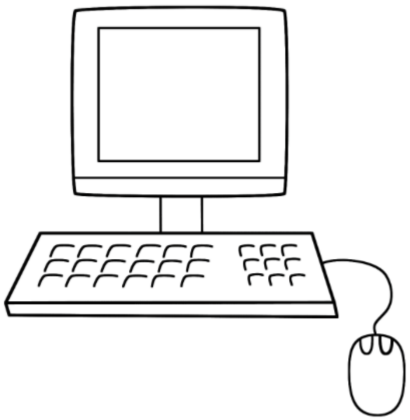
Part A: Visual Recognition and Deep Learning Context

What is Visual Recognition?

- In a nutshell:
- Teaching computers to 'see' like humans.
- Covers tasks: classification, detection, segmentation, recognition.
- Applications: autonomous driving, healthcare imaging, surveillance, robotics.

Visual recognition…

… is teaching computers to see

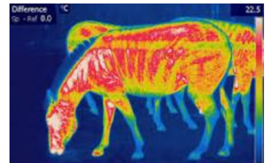


Humans see…

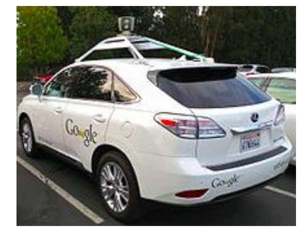
Computers see…

Teaching computers to “see” like humans

“see” not just RGB data

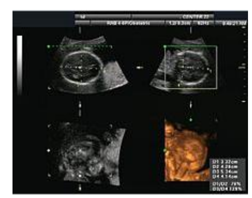


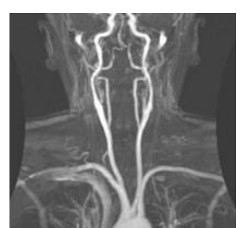
LIDAR
(laser radar)



- IR (infrared) etc
- ToF camera (Time of Flight)
  - ‘range’ camera gives depth
- Medical
  - ultrasonography
  - MRI
- & more

Kinect





Human vs. Computer Vision

- Humans: rich perception (context, depth, motion, prior knowledge).
- Computers: rely on data (RGB, depth, IR, LIDAR).
- Grand Goal: automated scene understanding comparable to humans.

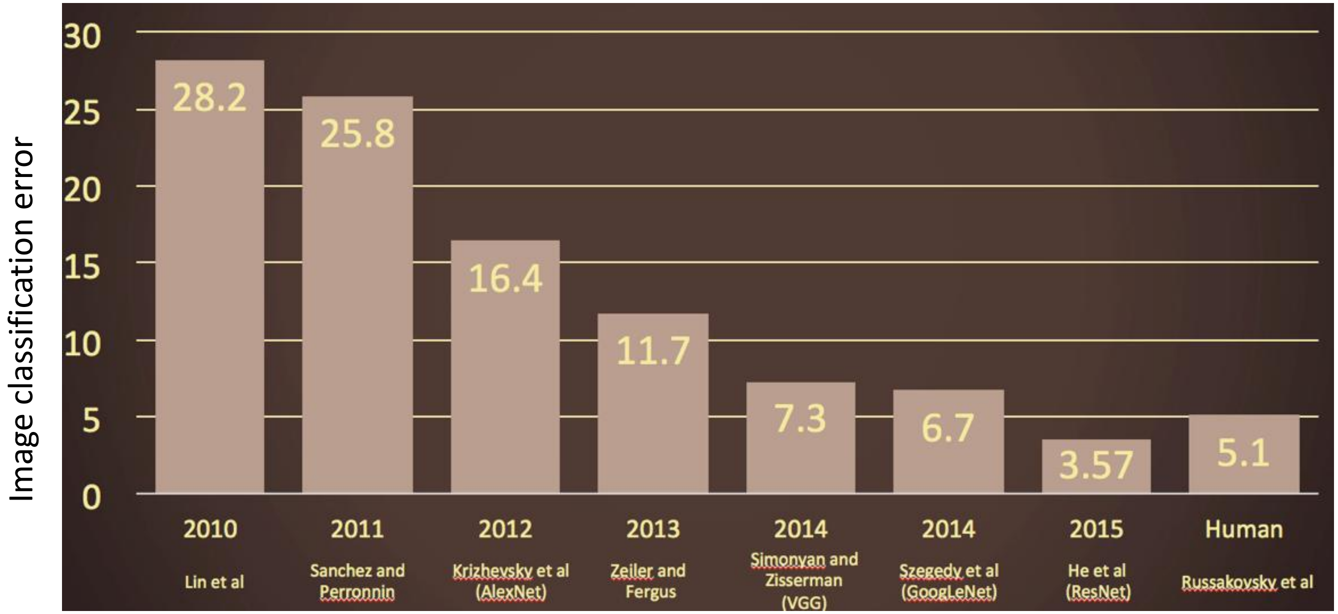
Success Stories of Deep Learning

- ImageNet Challenge (2012): AlexNet breakthrough.
- Medical imaging: diabetic retinopathy, pathology, radiology.
- Diffusion models (Stable Diffusion, DALL-E).
- Segment Anything Model (2023, Meta AI).
- Multimodal AI: CLIP, GPT-4V.

Success stories of deep learning

Image classification results

ImageNet- Large scale visual recognition challenge: 1000 categories, 1,000,000 images



Computer vision has surpassed human level performance on this benchmark!

Picture courtesy: http://cs231n.stanford.edu/index.html

Large scale video classification

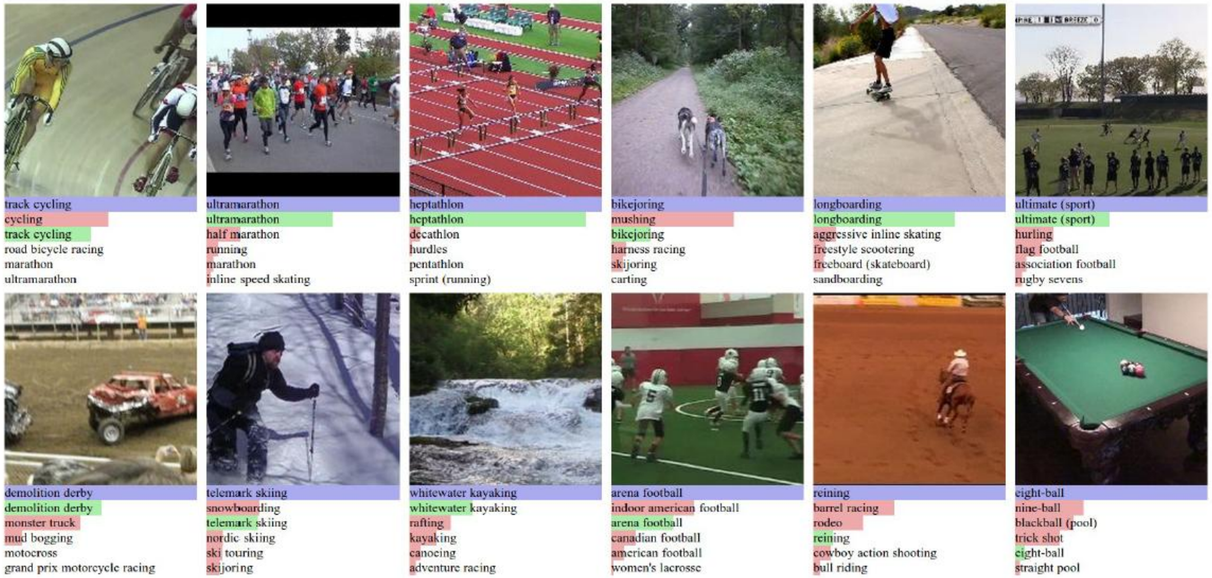


Figure 4: Predictions on Sports-1M test data. Blue (first row) indicates ground truth label and the bars below show model predictions sorted in decreasing confidence. Green and red distinguish correct and incorrect predictions, respectively.

http://cs.stanford.edu/people/karpathy/deepvideo/

Style Transfer

(a) Content (b) Style (c) Content + Style

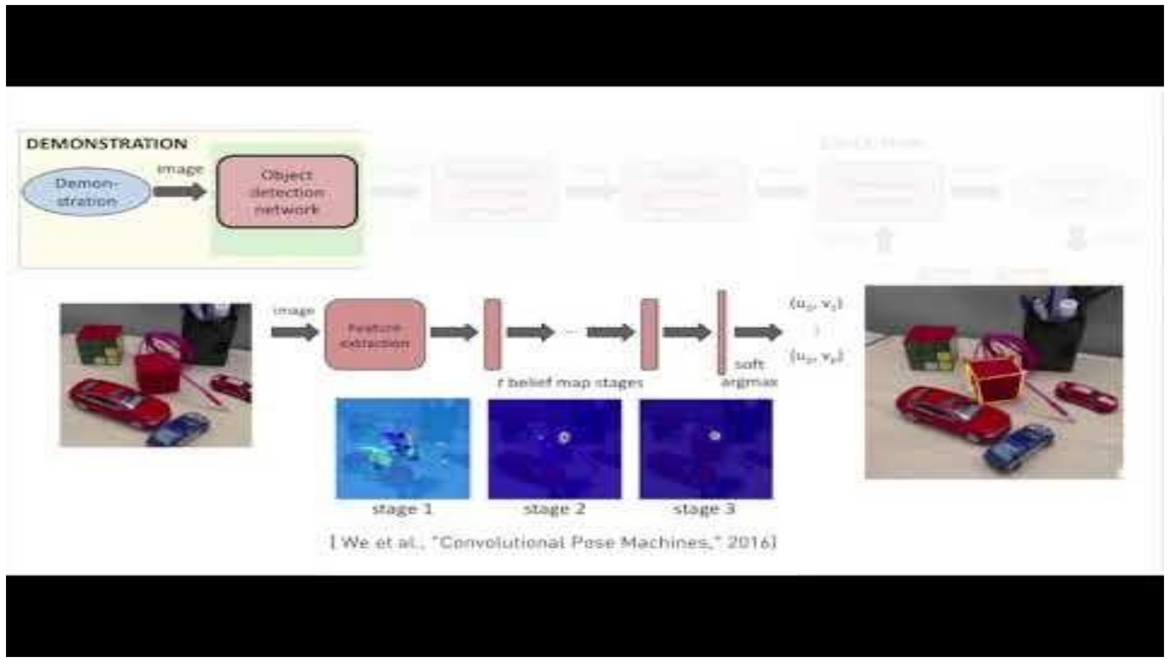
Figure 1. Example of using the Neural Style Transfer algorithm of Gatys *et al*. to transfer the style of Chinese painting (b) onto The Great Wall photograph (a). The painting that served as style is named "Dwelling in the Fuchun Mountains" by Gongwang Huang.

https://arxiv.org/abs/1705.04058

Deep reinforcement learning

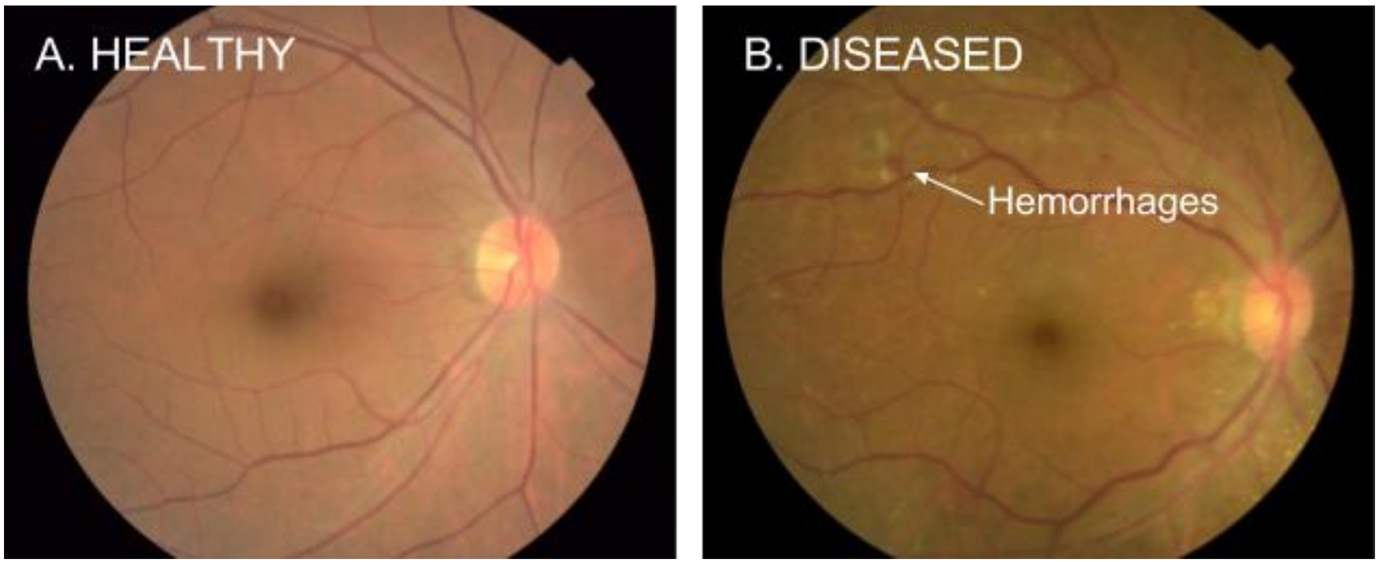
Picture source: https://deepmind.com/blog/deep-reinforcement-learning/

Impressive robotics with deep learning



https://www.youtube.com/watch?v=B7ZT5oSnRys

Diabetic retinopathy using deep learning



https://www.nature.com/articles/s41467-023-44676-z

Photorealistic image generation

From NVIDIA research: https://arxiv.org/pdf/1710.10196v1.pdf

Diffusion-based image and video generation

https://stability.ai/

Segment anything model

https://segment-anything.com/

Deep learning and natural language processing

- Impressive developments are happening in the NLP space
- Word embedding
- Language translation
- Language modeling
- ChatGPT!!

http://ruder.io/nlp-imagenet/

What created this revolution?

- Lots and lots of annotated data (such as ImageNet)
- Compute power (parallel processing with GPUs)
- Strong neural network architectures
- Good old back-prop algorithm + only a few new tweaks! And
- Open-source software platforms: TensorFlow, PyTorch,…

Challenges of Deep Learning

- Requires massive labeled datasets.
- Bias in training data → unfair outcomes.
- Adversarial vulnerability (images misclassified with tiny perturbations).
- Poor interpretability → 'black box' issue.
- High compute and energy costs.

WE DON’T HAVE AUTOMATED SCENE UNDERSTANDING YET

Today at NYT

- Gary Marcus, The Fever Dream of Imminent ‘Superintelligence’ Is Finally Breaking: https://www.nytimes.com/2025/09/03/opinion/ai-gpt5-rethinking.html?unlocked_article_code=1.jE8.MKWI.YCS0TSrbPReK&smid=url-share

Video: Yann LeCun on AI (Historical Context)

- Watch here: https://www.youtube.com/watch?v=4__gg83s_Do
- Pioneer of convolutional neural networks (CNNs).
- Perspective on AI and AGI.

Video: MIT Economist on AI and Jobs (Societal/Economic Impact)

- Watch here: https://www.youtube.com/watch?v=-zF1mkBpyf4
- Economic disruption/hype from AI adoption

Video: Andrej Karpathy on Deep Learning (Modern Breakthroughs)

- Watch here: https://www.youtube.com/watch?v=LCEmiRjPEtQ
- Former Tesla/Stanford researcher.
- Discusses modern breakthroughs and future directions.

Part B: Machine Learning Foundations

What is Machine Learning?

- Arthur Samuel (1959): 'Field of study that gives computers the ability to learn without explicit programming.'
- Tom Mitchell (1997): Learning improves performance on a task with experience.
- Core idea: Learn functions/mapping using data.

Why do we need ML?

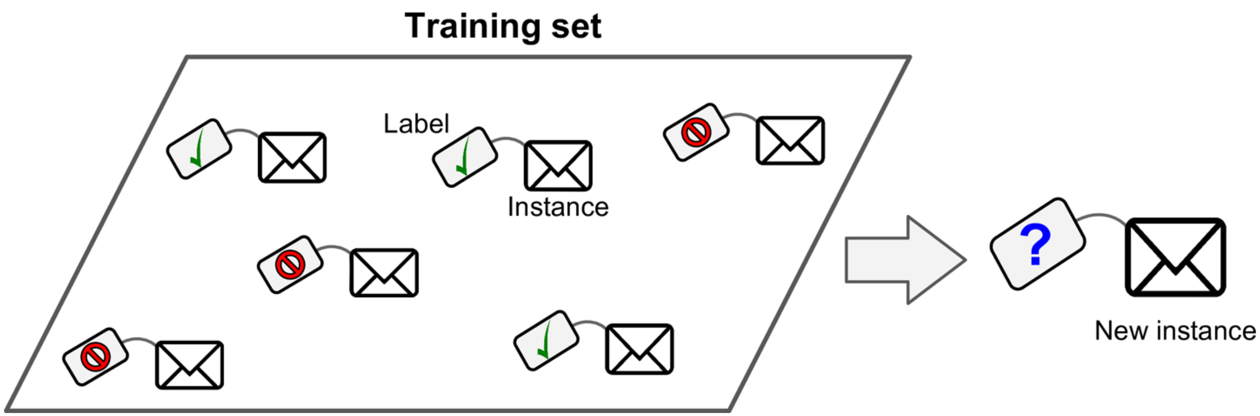
- Hard-coded rules fail in complex domains.
  - Quite relevant for visual recognition: You simply cannot describe a cat or dog or an object with full-proof, explicit description.
- ML adapts to new data automatically.
- Applications: spam filters, recommendation systems, speech recognition, visual recognition,…

Types of ML

- Supervised: learns from labeled data.
- Unsupervised: finds structure in unlabeled data.
- Semi-supervised: mix of labeled + unlabeled.
- Reinforcement learning: learns by trial and error with rewards.
- Batch vs Online learning.

Supervised Learning

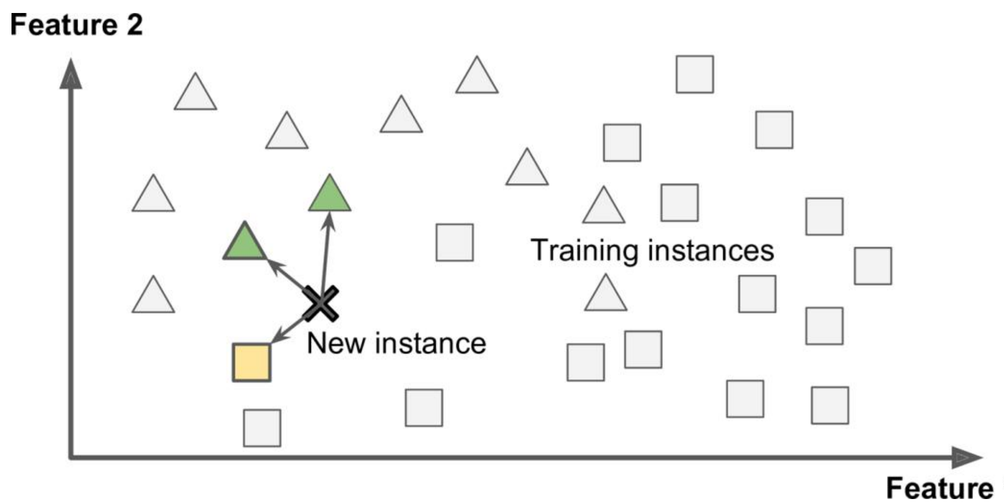
Classification:
The spam filter is a good example of this: it is trained with many example emails along with their *class* (spam or ham), and it must learn how to classify new emails.



Regression: Another typical task is to predict a *target* numeric value, such as the price of a car, given a set of *features* (mileage, age, brand, etc.) called *predictors*. To train the system, you need to give it many examples of cars, including both their predictors and their labels (i.e., their prices).

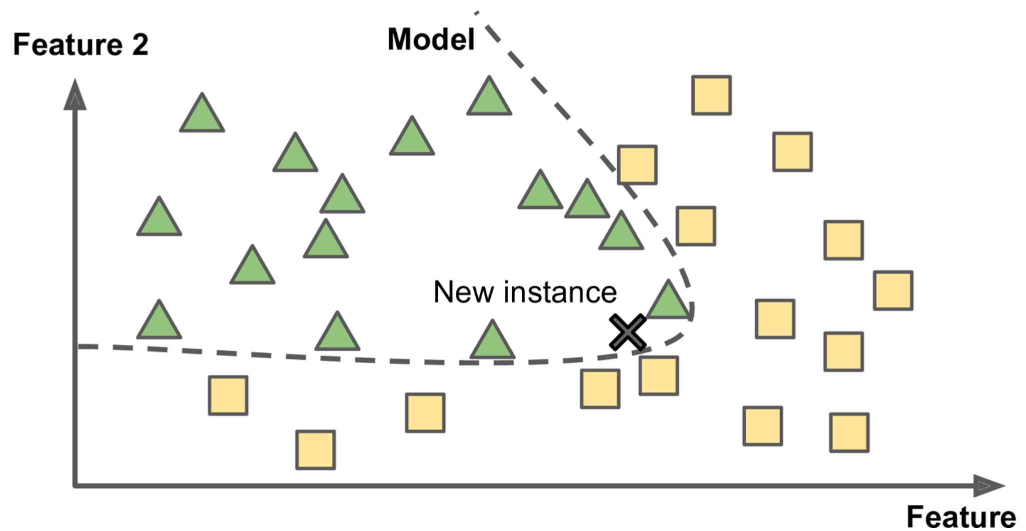
Instance-based supervised learning

- Remember all training examples
- When a test email comes, compare it with its “neighbors” from the training examples and classify accordingly
- Requires a measure of similarity
- Example: k-nearest neighbor (knn) method



Model-based supervised machine learning

- From all the training examples, build a model for the learner
- When a test example comes, apply the model
- Don’t need to remember all training examples, after training
- Examples: neural net, support vector machine, linear regression, etc.

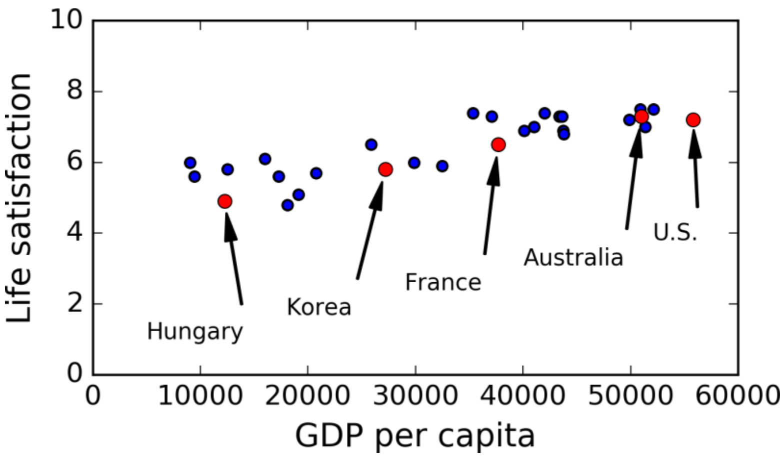


Linear Regression

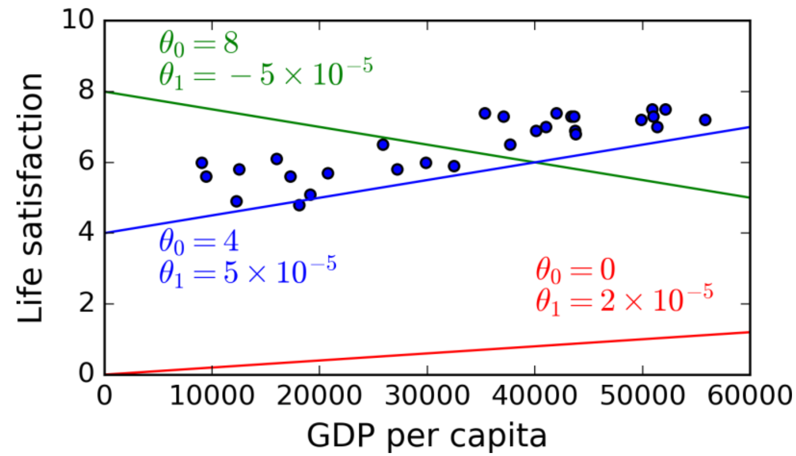
- Model: $y = \theta_0 + \theta_1 x$.
- Fits a straight line to data.
- Used for trend prediction (housing prices, stock prices).

Linear-model based supervised learning

Do we see any trend?



A few possible linear models

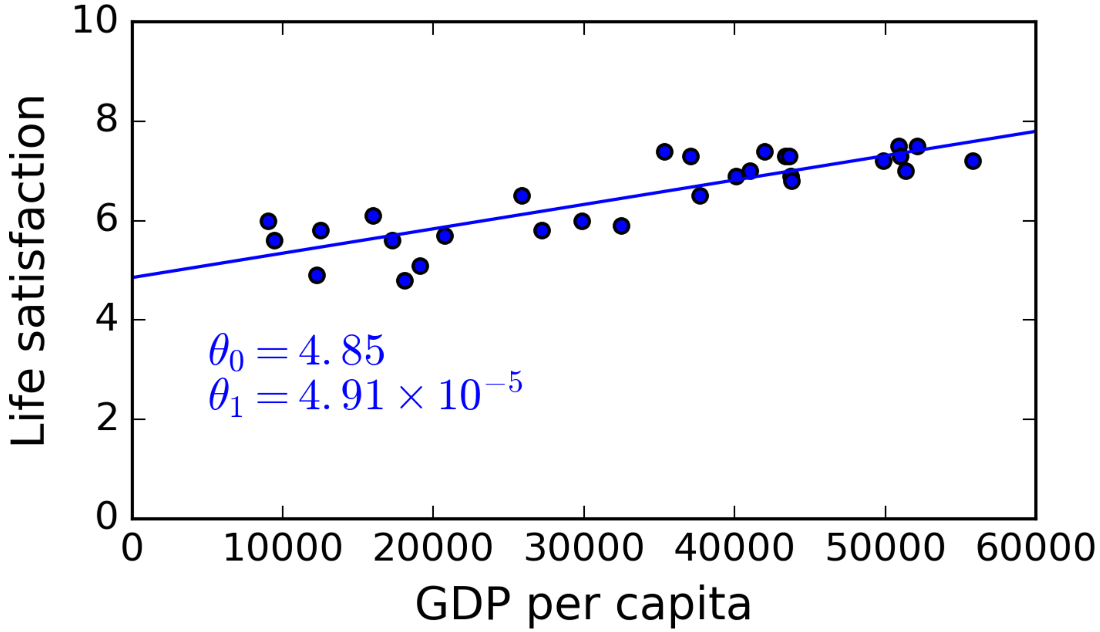


Linear model: $\text{life\_satisfaction} = \theta_0 + \theta_1 \times \text{GDP\_per\_capita}$

Parameters of the model: θ_0, θ_1

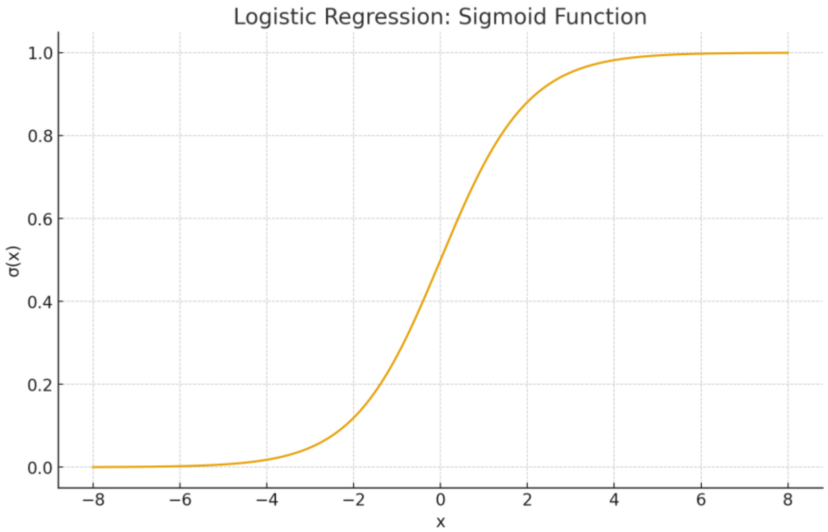
Linear-model based supervised learning

Best fit linear model



Logistic Regression

- Used for binary classification tasks.
- Outputs probability values using sigmoid function.
- Example: predicting if an image is a cat or not.

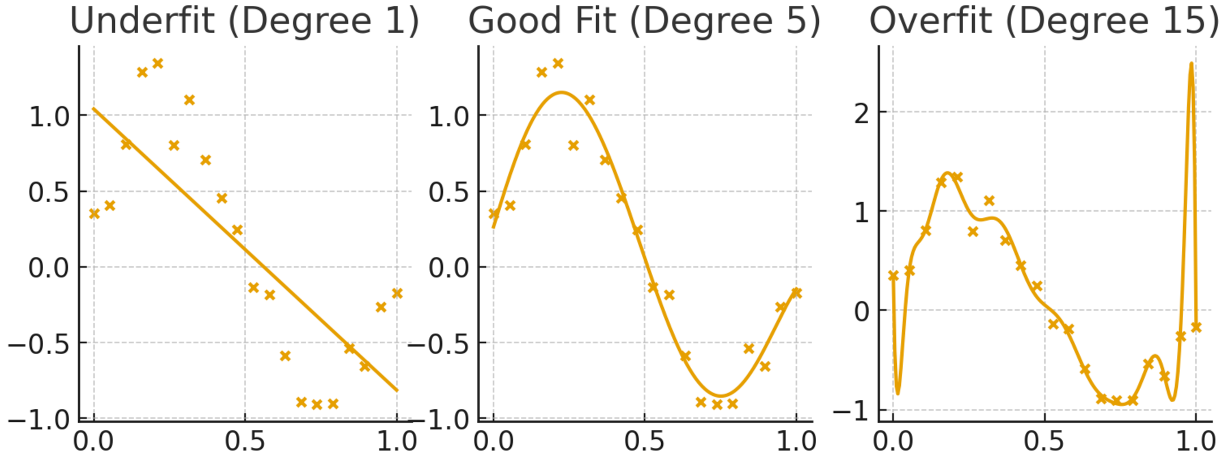


Training & Evaluation Metrics

- Accuracy = correct predictions / total predictions.
- Precision, Recall, F1-score (important in imbalanced data).
- Confusion matrix for visualization.
- ROC curve and AUC as performance measures.

Overfitting vs. Underfitting

- Overfitting: model memorizes training data → poor generalization.
- Underfitting: model too simple → fails to capture patterns.
- Goal: balance bias and variance.

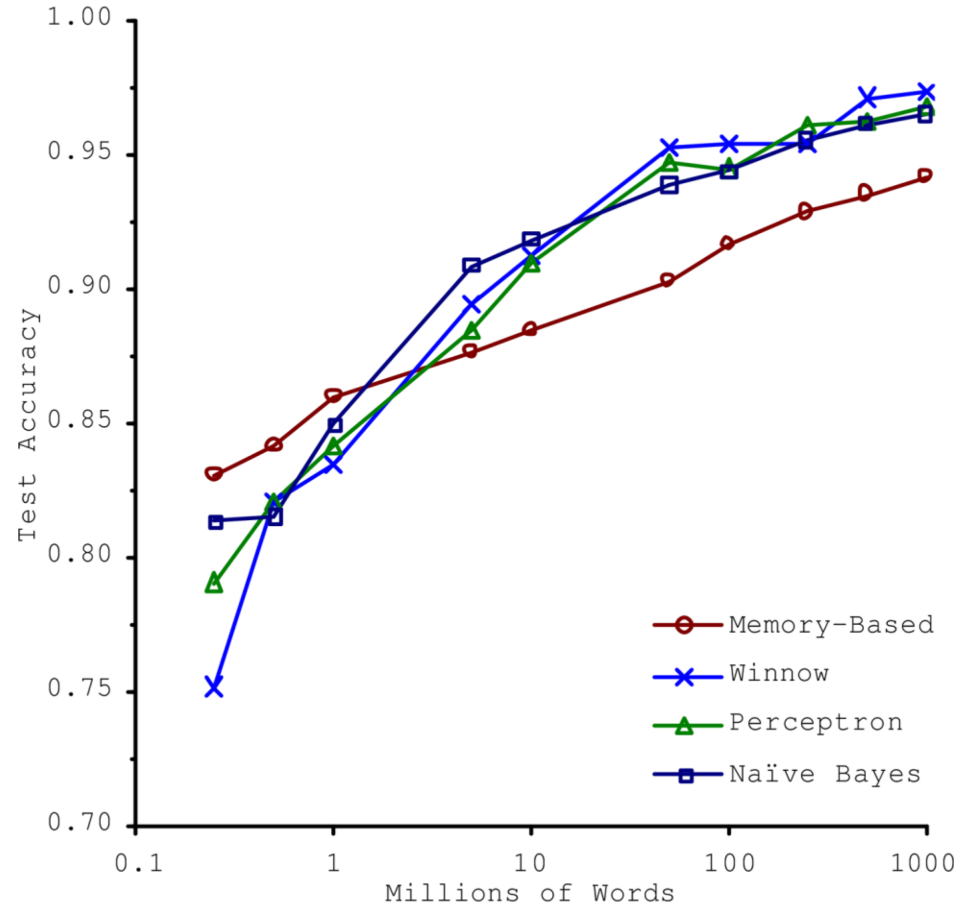


Regularization

- Penalty on large parameter values (L1, L2 regularization).
- Helps prevent overfitting.
- Encourages simpler models.

Challenge 1

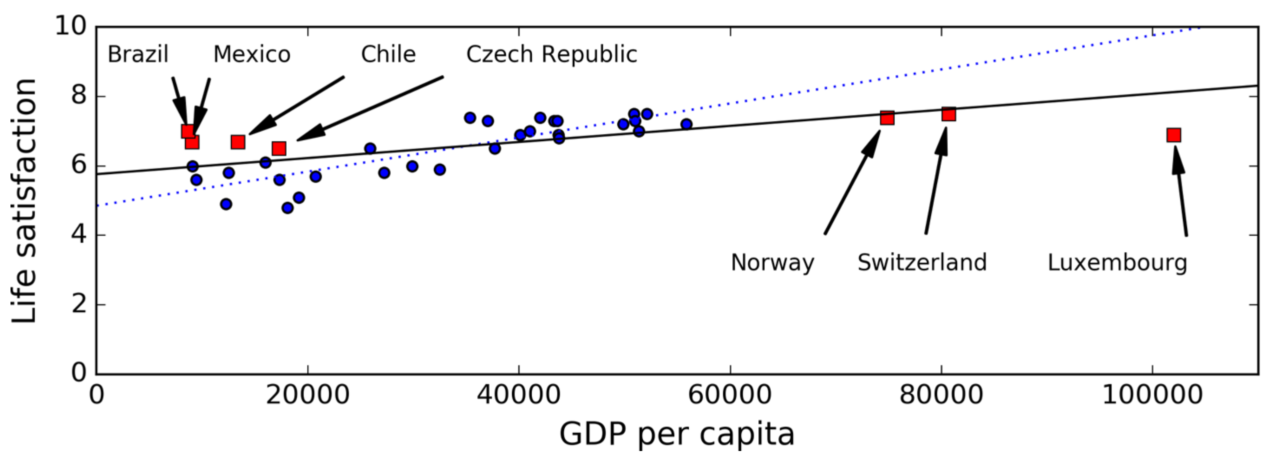
- Insufficiency of annotated training data



The importance of data versus algorithms: by Peter Norvig

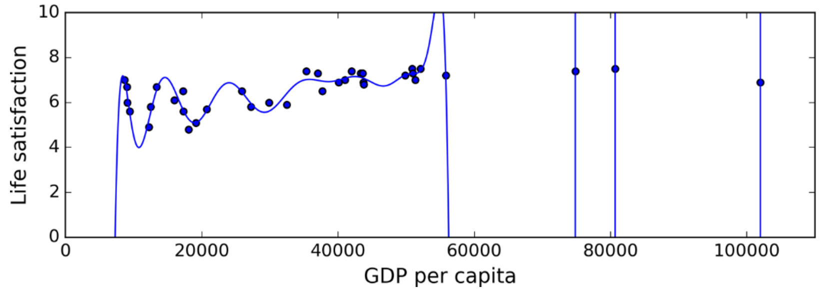
Challenge 2

- Non-representative training data

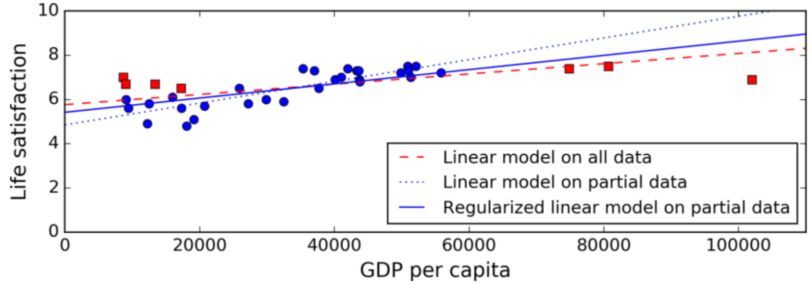


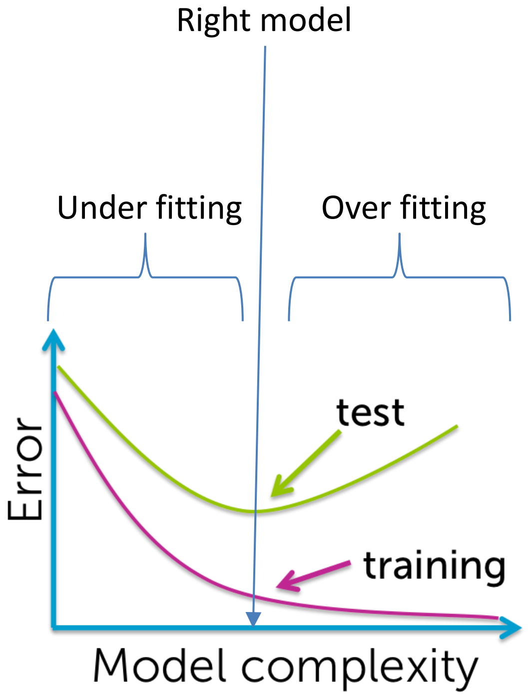
Challenge 3

Fitting a high degree polynomial: typical overfitting



Regularization reduces risk of overfitting





What is regularization?

Part C: Instance-based Learning with k-NN

ML as Function Mapping ($x \rightarrow y$)

- Machine learning learns a function $f(x) \approx y$.
- x = features (inputs), y = annotations (outputs).
- Goal: generalize well to unseen test data.

Supervised machine learning: the tabular view

Independent variable (aka features or predictors)

Output / Prediction / dependent variable

x_1	x_2	x_3	x_4	y
1.2	-3.9	4.0	0	1.6
2.1	2.4	-0.7	-0.2	1.2
…	…	…	…	…
…	…	…	…	…
3.2	…	…	1.9	0.3
1.4	…	…	1.5	?
3.1	…	…	2.1	?

Training data: complete table

Test data: incomplete table

ML learns to map **x** to y

In other words, ML learns a function, *f* so that
$y = f(\mathbf{x})$

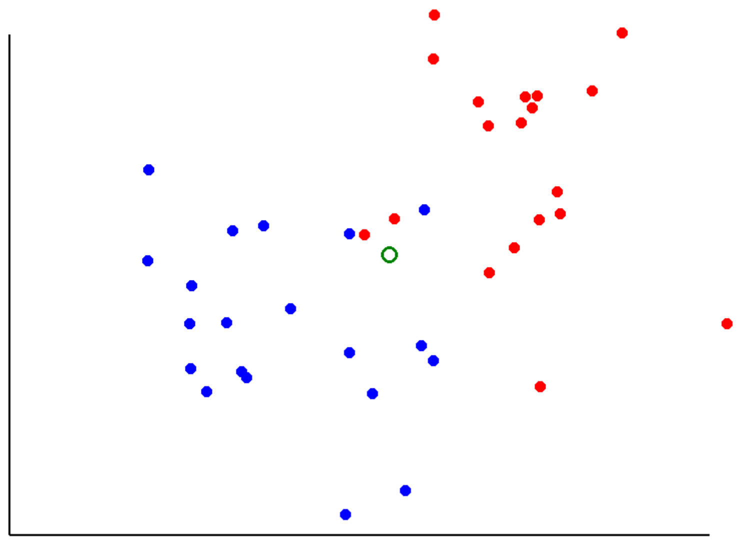
The function *f* is called prediction function

k-NN Algorithm (intuition, steps)

- Store all training examples.
- For a new point, compute distance to all training points.
- Pick the k (could be 1, 2, 3, etc.) closest neighbors.
- Predict class by majority vote (classification) or average (regression).

Toy Example (2D features, distance calculations)

- Example: classify a point (1,1) with k=3.
- Compute distances to training points.
- Select nearest neighbors → assign majority label.



K-nn: A toy numerical example…

x_1	x_2	y
2	-1	0
3	2	1
0	4	0
-2	5	0
2	0	1
1	1	?

Training data, m = 5

Test data point

For this problem, note that the feature vector dimension, d=2

Let's assume k = 3

To find out k=3 nearest neighbors, compute distances:

$D_1([1, 1], [2, -1]) = |1-2|+|1+1| = 3$
$D_2([1, 1], [3, 2]) = |1-3|+|1-2| = 3$
$D_3([1, 1], [0, 4]) = |1-0|+|1-4| = 4$
$D_4([1, 1], [-2, 5]) = |1+2|+|1-5| = 7$
$D_5([1, 1], [2, 0]) = |1-2|+|1-0| = 2$

So, k=3 nearest neighbors are
$N_3([1,1]) = \{1, 2, 5\}$

Prediction for test data point:
$f([1, 1]) = Ave([y(1), y(2), y(5)])$
$= Ave([0, 1, 1]) = 1$

Here, we computed "Ave" by taking mode.

Choosing k (validation sets, bias-variance tradeoff)

- Small k → sensitive to noise (low bias, high variance).
- Large k → smoother decision boundaries (high bias, low variance).
- Use cross-validation to choose optimal k.

Choosing *k* in practice

- Divide training data into two sets: training (90%) and validation (10%).

- For each *k* in a range, find out k-nn prediction accuracy on the validation set.

- Choose the *k* that has yielded the highest accuracy on the validation set.

MNIST digit image classification

Small 28 pixels-by-28 pixels images of handwritten digits

The visual recognition problem definition:
to recognize the digit from an image

We can attempt to solve this using k-nn.

Feature dimension, d = 28 * 28 = 784

Pixel values (feature) Digit

	x_1	x_2	…	x_{784}	y
Training data	0.1	0.3	…	0.0	0
	0.2	0.1	…	0.5	1
	…	…	…	…	…
	…	…	…	…	…
	0.0	0.98	…	0.8	9
Test data	0.5	0.25	…	0.36	?
	0.1	0.95	…	0.1	?

A recommended resource: https://cs231n.github.io/classification/

Efficient Computation of k-NN

- Brute force distance computation is expensive.
- Vectorization with NumPy/PyTorch speeds up calculations.
- KD-trees, ball trees, and approximate nearest neighbor methods scale to large datasets.

Efficient computation of K-nn: Vectorization

- For loops are slow. How do we avoid for loops in K-nn computation?
- Suppose X^{tr} is the training data matrix of shape N-by-d and X^{tst} is the test data matrix of shape M-by-d, d is the dimension of feature vector
- We want to compute the M-by-N distance matrix D:

$$D_{ij} = \sum_{k=1}^{d} \left(X_{ik}^{tr} - X_{jk}^{tst}\right)^2 = \sum_{k=1}^{d} \left(X_{ik}^{tr}\right)^2 + \sum_{k=1}^{d} \left(X_{jk}^{tst}\right)^2 - 2\sum_{k=1}^{d} X_{ik}^{tr} X_{jk}^{tst}$$

- Using Python's broadcast feature, we can compute D as:

$$\begin{aligned} &D \\ &= sum(X^{tr} ** 2, dim = 1, keepdim = True) \\ &+ transpose(sum(X^{tst} ** 2, dim = 1, keepdim = True) - 2 \\ &* matmul(X^{tr}, transpose(X^{tst})) \end{aligned}$$

MNIST Digit Classification with k-NN

- 28x28 grayscale images (784 features).
- Train k-NN on digits 0–9.
- Works well for small datasets but slow for large scale.
- Let’s look at the notebook

Strengths & Limitations of k-NN

- Strengths: simple, interpretable, no training time.
- Limitations: high memory usage, slow prediction, poor in high dimensions.

Transition to Neural Networks

- k-NN shows limits for complex tasks.
- Neural networks learn abstract features directly.
- Next week: perceptron, MLPs, backpropagation.

Wrap-Up (10 min)

Recap: Visual Recognition, ML Foundations, k-NN

- We introduced visual recognition and its challenges.
- Explored ML foundations.
- Learned k-NN as first ML algorithm for vision tasks.

Looking Ahead: Neural Networks and CNNs (Week 2)

- Next: neural networks → perceptron, MLPs.
- Then: convolutional neural networks (CNNs) for images.
- Transformers in vision coming later in the course.