UNIVERSITY OF ICELAND

Machine Learning Basics

Chapter 1: Introduction

Based on "Dive into Deep Learning" by Zhang et al.

Instructor: Guðmundur Einarsson

University of Iceland

Slides adapted from Hafsteinn Einarsson, 2025 fall course

What is Machine Learning?

The science of pattern recognition from data

Traditional Programming

How we've built software for decades

- Rigid set of rules
- Precise specifications
- Deterministic behavior

Example: E-Commerce Platform

Building with traditional rules:

1. User interface in browser/mobile app
2. Database for user state and transactions
3. Business logic mapping circumstances to actions

Programming Business Logic

```
def add_to_cart(user_id, product_id):
    # Explicit rule for every action
    if cart_exists(user_id):
        cart.add_item(product_id)
    else:
        create_cart(user_id)
        cart.add_item(product_id)
```

Handle every corner case explicitly

When Traditional Programming Works

- Well-defined rules
- Predictable scenarios
- Limited edge cases
- No adaptation needed

Perfect for deterministic tasks!

Test Your Understanding

When should you use traditional programming instead of machine learning?

- When you have well-defined rules and predictable scenarios
- When dealing with image recognition tasks
- When you need to recognize speech or audio patterns
- When the rules change frequently over time

When Traditional Programming Fails

Some problems resist explicit rules

Even the smartest programmers struggle with these...

Challenge Problems

- **Weather Prediction:** Tomorrow's weather from satellite images
- **Question Answering:** Understanding free-form text questions
- **Person Detection:** Identifying people in images
- **Recommendations:** Suggesting products users haven't seen

Why These Problems Are Hard

- **Dynamic patterns:** Rules change over time
- **Extreme complexity:** Millions of interactions
- **Unknown principles:** We don't know the rules
- **Subconscious processing:** Beyond conscious understanding

> These require learning, not programming!

This particular comic aged like milk

Sources: xkcd #1425: Tasks (https://xkcd.com/1425/)

Test Your Understanding

Which characteristics make a problem suitable for machine learning?

- [ ] Patterns in the data change over time
- [ ] The underlying rules are unknown or too complex to write
- [ ] You have access to large amounts of example data
- [ ] The problem requires simple arithmetic calculations

The Machine Learning Approach

- Computer learns patterns from data
- No explicit rules needed
- Some approaches can even adapt to new situations
    - What is a new situation?

Machine learning is the study of algorithms that can learn from experience, also known as...

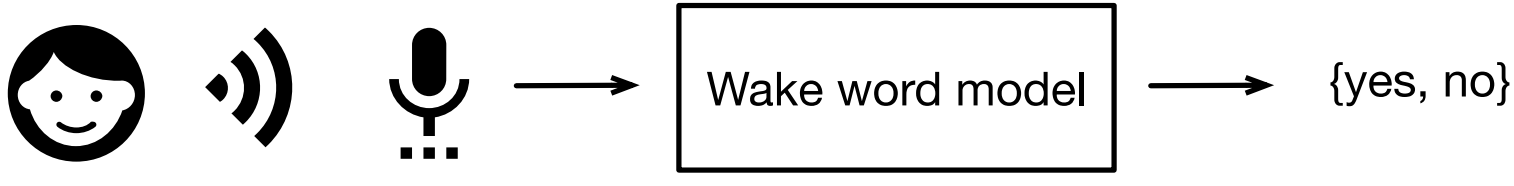
Programming with Data

> "Machine Learning is programming with data instead of explicit instructions"

The paradigm shift that changed computing

Sources: Dive into Deep Learning - Introduction (https://d2l.ai/chapter_introduction/index.html)

Example: Wake Word Detection



How does "Hey Siri" or "OK Google" work?

Why Is Wake Word Detection Hard?

Consider the technical challenge:

- **44,000 samples per second**
- Each sample = sound wave amplitude
- Must map raw audio → wake word detection

> What rule could reliably detect "Alexa" in all these samples?

We don't know how to write such rules from scratch!

How Wake Word Detection Works

1. Continuously listen to audio
2. Extract audio features
3. Check if pattern matches wake word
4. Activate if confidence is high

Traditional Approach Attempt

```
def detect_wake_word(audio):
    if audio_matches_pattern("ah-lek-sah"):
        return True
    elif audio_matches_pattern("uh-lek-suh"):
        return True
    # ... hundreds more rules?
    return False
```

Too many variations to code manually!

The ML Alternative

```
model = train_wake_word_detector(
    positive_examples=["alexa_1.wav", "alexa_2.wav", ...],
    negative_examples=["other_1.wav", "other_2.wav", ...]
)

def detect_wake_word(audio):
    return model.predict(audio) > threshold
```

Learn patterns from examples instead!

Why This Approach is Powerful

- Handles accent variations
- Works in noisy environments
- Improves with more data
- No need to code every scenario
- Can generalize to new speakers

Test Your Understanding

Why is machine learning better than traditional programming for wake word detection?

- ML can handle countless variations in pronunciation, accent, and background noise
- ML provides deterministic, predictable outputs
- ML uses less computational resources
- ML code is simpler to write and understand

When to Use Machine Learning

- Pattern recognition tasks
- Rules are too complex to write
- Need to adapt to changing data
- Have sufficient training data

> ML transforms intractable coding problems into tractable data problems

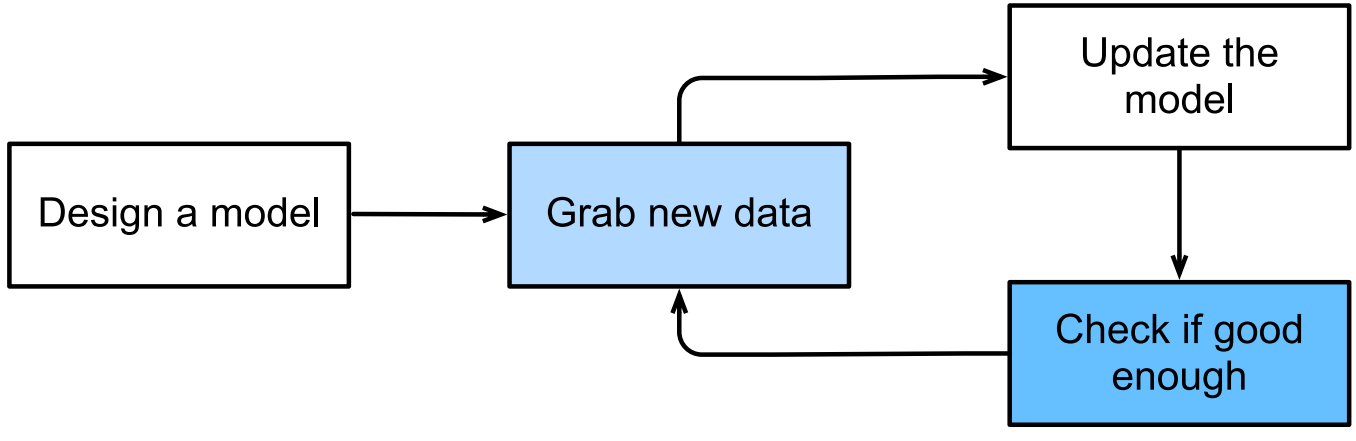
Key Components of ML

The Four Pillars

1. Data
2. Models
3. Objective Functions
4. Optimization Algorithms

Sources: Dive into Deep Learning - Key Components (https://d2l.ai/chapter_introduction/index.html#key-components)

How They Work Together



An iterative process of improvement

Wake Word Example Revisited

Let's see how our four pillars work for "Alexa" detection

Problem: Audio snippet → "Yes/No" for wake word

Sources: Dive into Deep Learning - Training Process (https://d2l.ai/chapter_introduction/index.html#key-components)

Step 1: Define the Problem

- **Input:** Audio snippet (e.g., 1 second)
- **Output:** Binary decision (wake word or not)
- **Model family:** Neural network for audio

Must precisely define inputs and outputs first!

Step 2: Choose Model Family

The model has adjustable "knobs" (parameters)

- One setting → detects "Alexa"
- Different setting → could detect "Apricot"
- Same architecture, different parameters

> Model family must be rich enough for the task

Step 3: The Training Process

1. Start with random parameters (useless model)
2. Grab labeled audio examples
3. Adjust parameters to improve predictions
4. Repeat until performance is satisfactory

This is "learning" - finding the right parameter settings

Programming with Data

Traditional

Write explicit rules for wake word detection

Machine Learning

Write program that learns from wake word examples

> "Programming with data" instead of programming with rules

Different Tasks, Different Models

Similar tasks can share model families:

- "Alexa" detection ≈ "Apricot" detection
- Same architecture, different training

But fundamentally different tasks need different models:

- Image captioning needs vision + language models
- Translation needs sequence-to-sequence models

Test Your Understanding

How do the four pillars of machine learning work together in the wake word detection example?

- [ ] Data: Thousands of audio recordings of people saying the wake word
- [ ] Objective: Minimize false positives while maintaining high detection rate
- [ ] Model: A neural network that processes audio and outputs a detection score
- [ ] Optimization: Manually adjusting each parameter based on test results

Data

The Foundation of Machine Learning

The Data Revolution in Deep Learning

- More data = more powerful models
- Shift from small to big data transformed deep learning
- Many modern models require large datasets to work
- In small data regime, traditional methods often suffice

> The success of deep learning is largely due to data availability

Data Quality: Garbage In, Garbage Out

- Having lots of data isn't enough - we need the **right** data
- Mistakes in data → mistakes in predictions
- Non-predictive features → learning failure

Critical Applications: Predictive policing, resume screening, lending decisions

The Danger of Biased Data

Representation Failures

Example: Skin cancer detection trained only on light skin

Historical Bias Amplification

Resume screening learning from biased hiring history

> ⚠️ This happens without intent or awareness from data scientists

Why Data Matters

- Models learn from examples
- Data quality determines model quality
- "Garbage in, garbage out"

Quality vs Quantity

- More data usually helps
- But quality is crucial
- Clean, labeled data is expensive
- Balance is key

Data Representation

How we encode information for models

- Images → Pixel values
- Text → Numerical vectors
- Audio → Frequency features

Example: Images as Numbers

A 3×3 grayscale image:

```
[[255, 128,   0],
 [128,  64, 128],
 [  0, 128, 255]]
```

Each number represents pixel brightness

Example: Text as Vectors

Words mapped to numbers:

```
"deep" → [0.2, -0.5, 0.8, ...]
"learning" → [-0.1, 0.7, 0.3, ...]
```

Captures semantic relationships

Test Your Understanding

Which statements about data in machine learning are correct?

- [ ] Clean, labeled data is typically expensive to obtain
- [ ] Text vectors capture semantic relationships between words
- [ ] Data quality is more important than data quantity
- [ ] All data must be converted to numerical form before training

The Mushroom Example

Death cap - highly poisonous!

Sources: Wikipedia - Amanita phalloides (https://en.wikipedia.org/wiki/Amanita_phalloides)

Why Mushroom Classification?

- Life-or-death importance
- Subtle visual differences
- Expert knowledge required
- Interesting ML use case
- Errors can be very costly

The Cost of Errors

Mushroom Classification Example

Classifier: "20% chance this is poisonous"

Eat It ✓

- 80% chance: Delicious dinner (+10)
- 20% chance: Death (-∞)

Expected value: -∞

Discard It ✗

- 100% chance: No dinner (-1)
- 0% chance: Death (0)

Expected value: -1

Key Insight: In critical applications, even small error probabilities can be unacceptable when the cost of false negatives is catastrophic.

Traditional Approach

Field guides use complex decision trees:

- "If cap is white AND gills are free..."
- "If ring is present AND volva exists..."
- "If spore print is white AND..."

Problem: Too many rules, too many exceptions!

Feature Engineering

What features help identify poisonous mushrooms?

- Cap: color, shape, surface texture
- Gills: attachment, spacing, color
- Stem: color, ring, bulb shape
- Habitat: location, season, substrate

Data Collection Challenges

- Dangerous to collect poisonous samples
- Seasonal availability
- Geographic variation
- Need expert verification
- Imbalanced data (few deadly species)

ML Solution Benefits

- Learns subtle patterns humans miss
- Handles complex feature interactions
    - What is a feature interaction?
- Improves with more data
- Can provide confidence scores

> ML can save lives by making expert knowledge accessible, but the benefit needs to outweigh the cost of mistakes (e.g. false negatives when someone is poisoned)

Test Your Understanding

What makes mushroom classification a good machine learning problem?

- [ ] Subtle visual differences determine toxicity
- [ ] Expert knowledge is required for accurate identification
- [ ] Data collection presents unique challenges like seasonal availability
- [ ] The decision rules are simple and well-defined

Models

The Learning Structures

What is a Model?

- A mathematical function
- Transforms inputs to outputs
- Has adjustable parameters

Models as Function Approximators

Goal: Learn function f where

$$y = f(x)$$

Given input x, predict output y

Notation: True vs Predicted Values

y

True Value

- Actual label in data
- Ground truth
- What we want to predict

$\hat{y}$

Predicted Value

(pronounced "y-hat")

- Model's prediction
- Output of f(x)
- Our estimate of y

Loss measures the difference: $L(y, \hat{y})$

Parameters and Weights

- Parameters define the model's behavior
- Adjusted during training
- Capture learned patterns

Simple Linear Model

```
def linear_model(x, w, b):
    return w * x + b
```

- w: weight (slope)
- b: bias (intercept)
- Both are learned from data

From Simple to Deep Models

Simple Models

Perfect for appropriately simple problems

Linear regression, decision trees, basic classifiers

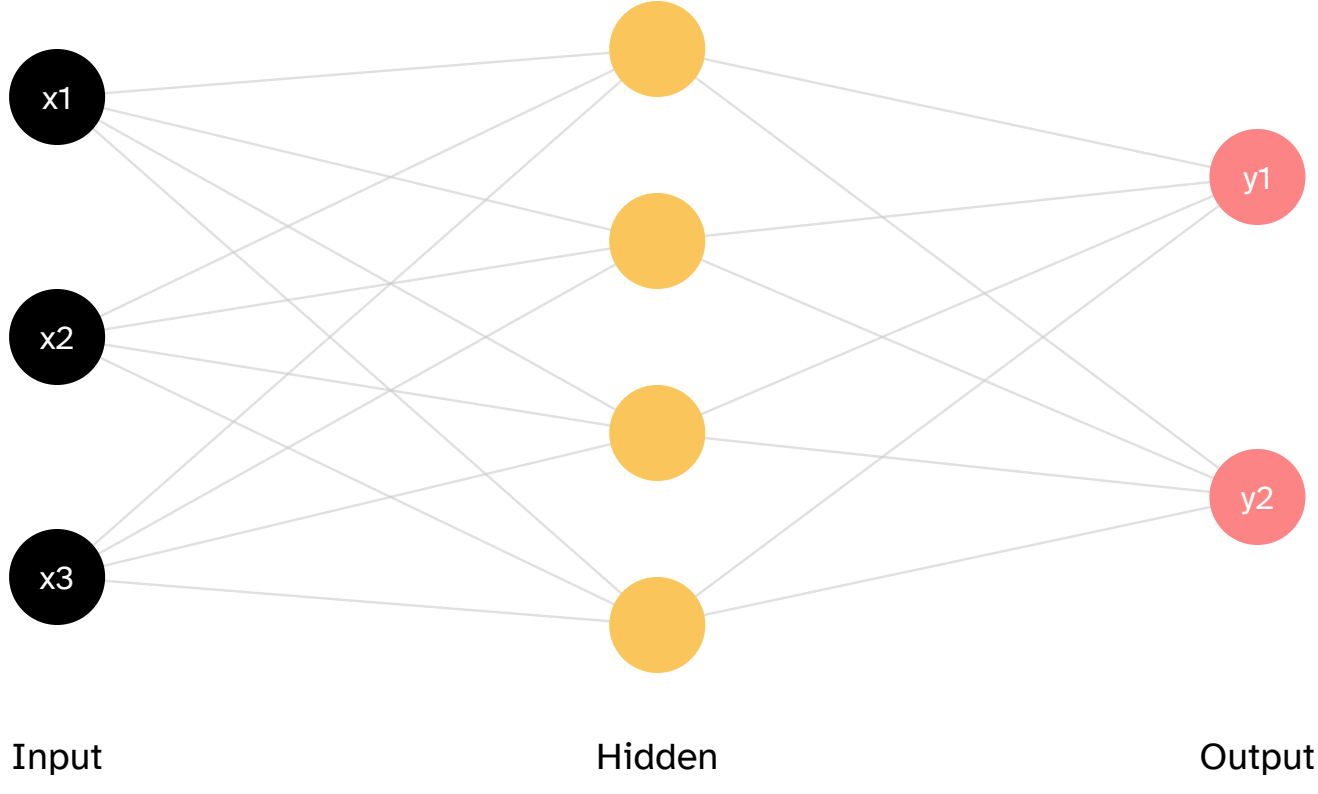
Deep Models

Many successive transformations chained together

The "deep" in deep learning = multiple layers

> Complex problems stretch the limits of classical methods

Neural Network Preview



Multiple layers of transformations

Model Capacity

- Ability to fit complex patterns
- More parameters = higher capacity
- Balance with overfitting risk

Choosing the Right Model

- Problem complexity
- Available data
- Computational resources
- Interpretability needs

Models: Key Insights

- Functions with learnable parameters
- Transform inputs to predictions
- Capacity must match problem

Test Your Understanding

Which statements about machine learning models are correct?

- [ ] A model is a mathematical function that transforms inputs to outputs
- [ ] Parameters are adjusted during training to capture patterns in data
- [ ] More parameters always lead to better model performance
- [ ] Model capacity should match the complexity of the problem

Objective Functions

Measuring Success

Why We Need Objective Functions

Machine learning = learning from experience

But what constitutes "improvement"?

- Different people might disagree on what's better
- We need formal, mathematical measures
- Enter: **Objective Functions**

> Convention: Lower is better (hence "loss functions")

What Are We Optimizing?

- Need to measure model performance
- Quantify "how wrong" predictions are
- Guide parameter updates

Loss Functions Explained

Loss = measure of prediction error

Lower loss = better predictions

Goal: minimize loss

Common Loss Functions

Regression: Squared Error

Square of (prediction - true value)

Easy to optimize, differentiable

Classification: Error Rate

Fraction of incorrect predictions

Hard to optimize directly → use surrogates

Mean Squared Error

For regression problems:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Penalizes large errors more

Cross-Entropy Loss

For classification problems:

$$L = -\sum_i y_i \log(\hat{y}_i)$$

Measures prediction confidence

Can be derived based on the maximum likelihood approach

Loss During Training vs Testing

Loss = function of model parameters

Training Loss

Like practice exam scores

Used to update parameters

Test Loss

Like final exam scores

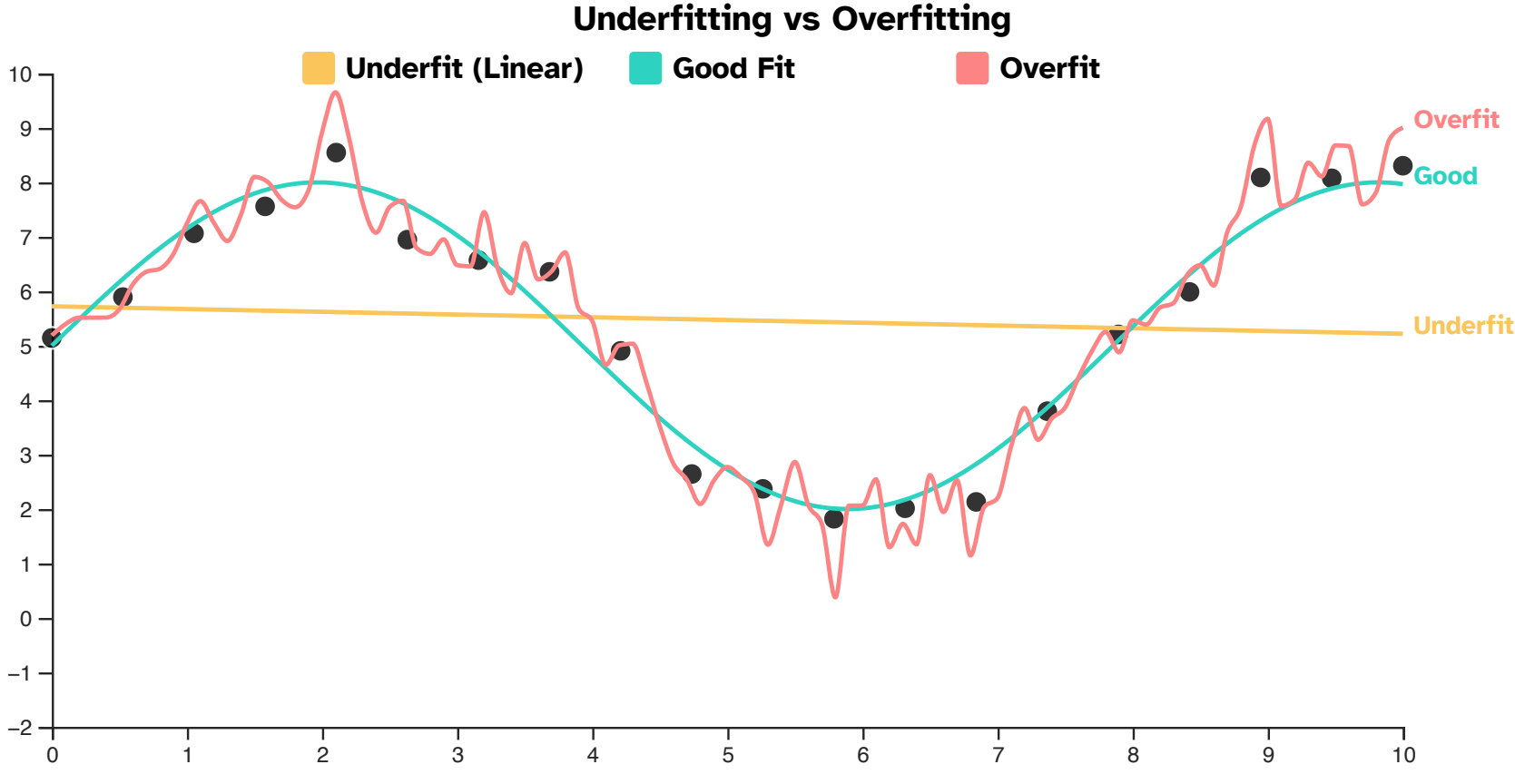
Measures real performance

⚠️ Good training score ≠ Good test score (overfitting)

Training vs Validation Loss

- Training loss: on training data
- Validation loss: on held-out data
- Gap indicates overfitting

Overfitting Preview



When overfitting, the model memorizes instead of generalizing

Objective Functions: Summary

- Quantify model performance
- Different losses for different tasks
- Balance training and generalization

Test Your Understanding

Which statements about objective functions and loss are correct?

- [ ] Lower loss always means better model performance
- [ ] MSE is used for regression while cross-entropy is used for classification
- [ ] Objective functions guide how model parameters are updated
- [ ] The gap between training and validation loss indicates overfitting

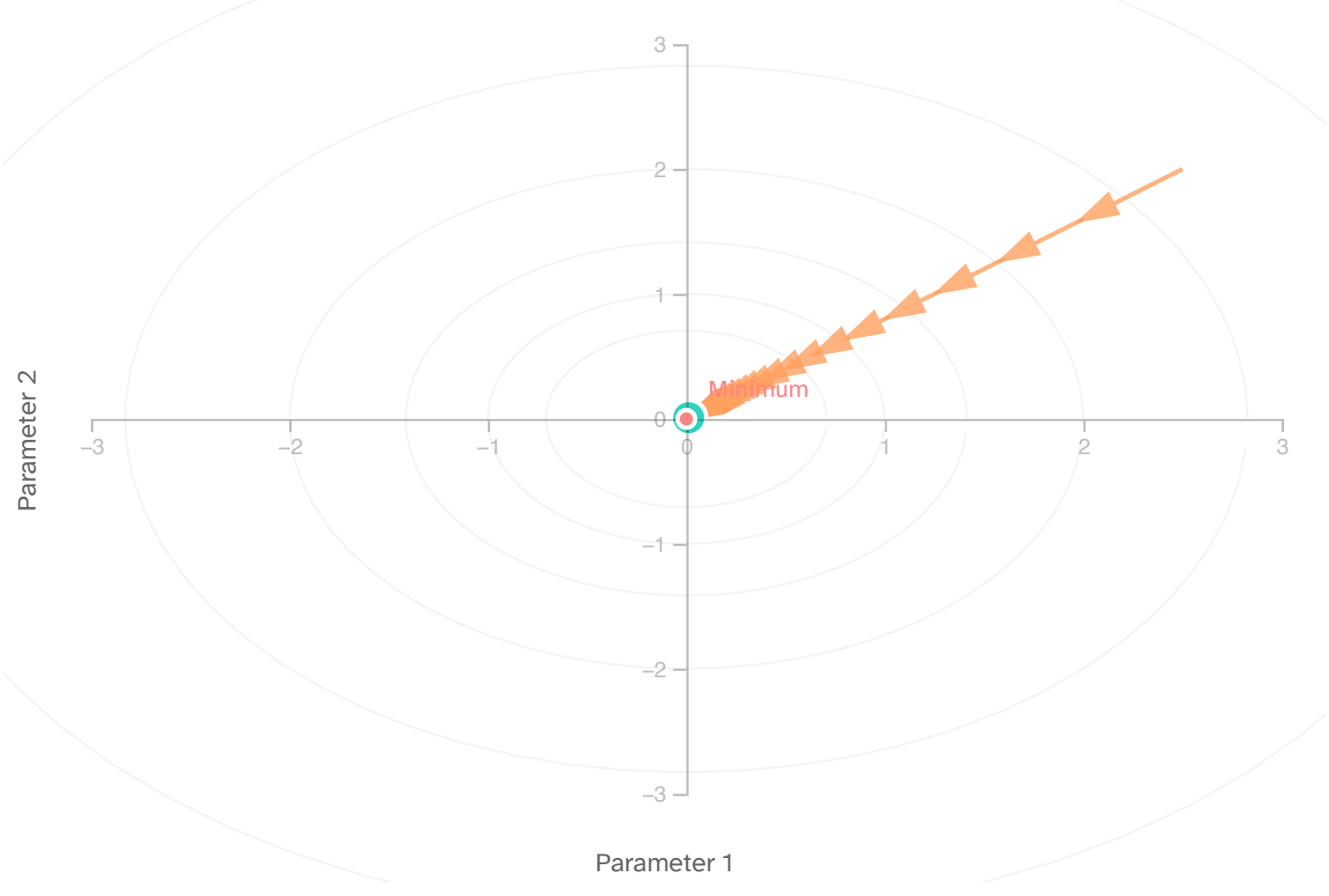
Optimization Algorithms

Finding the Best Parameters

Finding the Best Parameters

- Start with random parameters
- Measure performance (loss)
- Adjust to improve
- Repeat until convergence

The Optimization Landscape

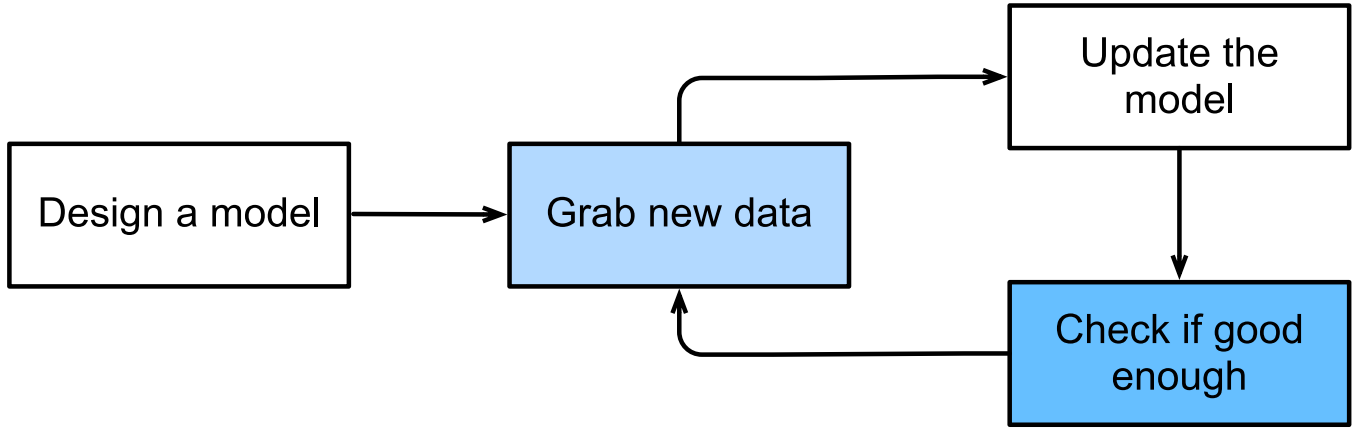


Finding the lowest point

Gradient Descent Basics

- Compute gradient (slope)
- Step in opposite direction
- Like walking downhill in fog

The ML Training Loop



Learning Rate Importance

- Too small: slow convergence
- Too large: might overshoot
- Critical hyperparameter

Convergence

- Loss stops decreasing
- Parameters stabilize
- Model has "learned"

Optimization: Key Points

- Iterative improvement process
- Gradient descent is fundamental
- Many variations exist

Test Your Understanding

Which statements about optimization in machine learning are correct?

- [ ] The learning rate controls how big each optimization step is
- [ ] Gradient descent always finds the global minimum
- [ ] Convergence means the loss has reached exactly zero
- [ ] Optimization is an iterative process that improves parameters gradually

Types of Machine Learning

Different Learning Paradigms

Three Main Paradigms

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

Overview Comparison

Supervised	Labeled	Predict labels
Unsupervised	Unlabeled	Find structure
Reinforcement	Rewards	Maximize reward

Test Your Understanding

Match the learning paradigm with its characteristics:

- [ ] All three paradigms require the same type of training data
- [ ] Unsupervised learning is used when we have clear target outputs
- [ ] Reinforcement learning learns from rewards and penalties
- [ ] Supervised learning requires labeled training data

Supervised Learning

Learning from Examples

What is Supervised Learning?

- Given: input-output pairs
- Learn: mapping function
- Goal: predict new outputs

Input-Output Pairs

Training data format:

```
(image of cat) → "cat"
(image of dog) → "dog"
(house features) → $500,000
(email text) → "spam"
```

Test Your Understanding

Which are valid examples of supervised learning tasks?

- [ ] Predicting house prices based on features like size and location
- [ ] Grouping customers by purchasing behavior without predefined categories
- [ ] Discovering hidden topics in a collection of documents
- [ ] Classifying emails as spam or not spam

Regression

Predicting Continuous Values

What is Regression?

- Output is a continuous number
- Examples: prices, temperatures, scores
- Goal: minimize prediction error

> Regression = predicting "how much" or "how many"

Real-World Regression Tasks

🏠 Real Estate

- House prices
- Rental rates
- Property valuations

📈 Finance

- Stock prices
- Risk scores
- Loan amounts

🌡️ Science

- Temperature prediction
- Chemical concentrations
- Growth rates

⚡ Engineering

- Power consumption
- Load forecasting
- Performance metrics

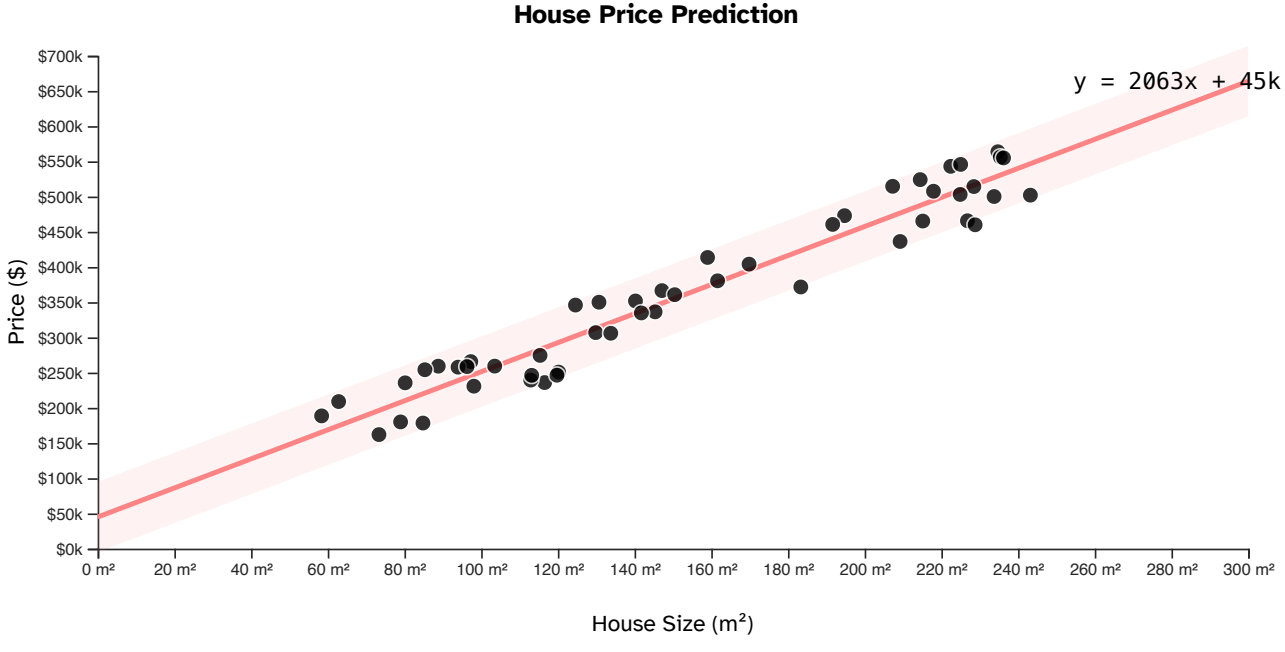
Linear Regression: The Foundation

$$\hat{y} = wx + b$$

- w: weight (slope) - relationship strength
  - w is often referred to as the parameters of the linear model
- b: bias (intercept) - baseline value
  - How can b be a part of x and w?
- ŷ: predicted value

Multiple features: $\hat{y} = w_1 x_1 + w_2 x_2 + \ldots + b$

Example: House Price Prediction



Interactive applet in web based slides

Each point is a house, line shows learned relationship

Measuring Regression Performance

Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Squares the errors (penalizes large errors more)
  - Why can it be bad to penalize large errors more? What is the impact of outliers?
- Always positive
- Lower is better

> Other metrics: MAE (Mean Absolute Error), R^2 score

Beyond Linear: Polynomial Regression

When relationships are non-linear:

$$\hat{y} = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \ldots$$

- Can fit curves, not just lines
- More flexible but risk of overfitting
- Need to choose degree carefully

Test Your Understanding

Which statements about regression are correct?

- [ ] Linear regression can only fit straight lines
- [ ] Regression predicts continuous numerical values
- [ ] MSE penalizes large errors more than small ones
- [ ] Higher degree polynomials always give better predictions

Classification

Predicting Categories

What is Classification?

- Output is a discrete category/class
- Examples: spam/not spam, digit recognition
- Can be binary or multi-class

> Classification = predicting "which type" or "what category"

Binary Classification

Two possible outcomes:

✓ Positive Class

- Email is spam
- Transaction is fraud
- Patient has disease
- Image contains cat

✗ Negative Class

- Email is not spam
- Transaction is legitimate
- Patient is healthy
- Image doesn't contain cat

Output: probability between 0 and 1

Multi-Class Classification

More than two categories:

Examples:

- Digit recognition (0-9)
- Language detection (100+ languages)
- Product categories (electronics, clothing, food...)
- Disease diagnosis (multiple conditions)

From Scores to Probabilities

The Softmax Function

Converts raw scores to probabilities:

$$P(y = k) = \frac{e^{z_k}}{\sum_j e^{z_j}}$$

Example:

```
Scores: [2.0, 1.0, 0.1]
Probabilities: [0.66, 0.24, 0.10]
Prediction: Class 0 (highest probability)
```

All probabilities sum to 1.0

Decision Boundaries

How classifiers separate classes:

Interactive applet in web based slides

Linear boundaries for simple problems, non-linear for complex ones

Measuring Classification Performance

Accuracy

% of correct predictions

Good for balanced datasets

Precision

% of positive predictions that are correct

Important when false positives are costly

Recall

% of actual positives correctly identified

Important when false negatives are costly

F1 Score

Harmonic mean of precision and recall

Balanced metric

Confusion Matrix

Visualizing classification errors:

True Positive	False Positive
False Negative	True Negative

Shows exactly where the model makes mistakes

Test Your Understanding

Which statements about classification are correct?

- [ ] Classification predicts discrete categories, not continuous values
- [ ] Binary classification can use a threshold on probability to make decisions
- [ ] Accuracy is always the best metric for classification
- [ ] Softmax ensures all class probabilities sum to 1

Tagging

Multi-Label Problems

Multi-Label vs Multi-Class

Multi-Class

One label per item

- Animal: cat OR dog OR bird
- Sentiment: positive OR negative
- Digit: 0 OR 1 OR … OR 9

Multi-Label

Multiple labels per item

- Movie: action AND comedy AND sci-fi
- Article: tech AND business AND AI
- Image: person AND car AND tree

Labels are not mutually exclusive in multi-label problems

Real-World Tagging Applications

📽️ Content Tagging

Netflix movie: [Drama, Thriller, Based on Book, Award-Winning]

🏥 Medical Diagnosis

Patient conditions: [Diabetes, Hypertension, Obesity]

📸 Image Annotation

Photo tags: [Sunset, Beach, People, Vacation, Summer]

📰 Document Classification

Research paper: [Machine Learning, Computer Vision, Neural Networks]

How Multi-Label Works

Independent binary classifiers for each label:

```
Input: Movie description

Output probabilities:
- Action: 0.85 ✓
- Comedy: 0.72 ✓
- Drama:  0.23 ✗
- Horror: 0.05 ✗
- Sci-Fi: 0.91 ✓

Threshold: 0.5
Tags: [Action, Comedy, Sci-Fi]
```

Each label has its own probability threshold

Challenges in Multi-Label Learning

- **Label Correlation:** Some labels often appear together
- **Label Imbalance:** Some tags are rare
- **Threshold Selection:** Different thresholds for different labels
- **Evaluation Complexity:** Partial matches complicate metrics

> Example: "Action" and "Adventure" often co-occur in movies

Test Your Understanding

Which statements about multi-label classification (tagging) are correct?

- [ ] An item can have multiple labels simultaneously
- [ ] Each label can have its own probability threshold
- [ ] Labels in multi-label problems are mutually exclusive
- [ ] Multi-label is the same as multi-class classification

Search and Ranking

Ordering Results by Relevance

The Ranking Problem

Not just finding results, but ordering them:

Query: "machine learning"

Challenge: Millions of results - which first?

- Most relevant at the top
- Personalized to user
- Consider multiple factors

> Users rarely look past the first page!

Learning to Rank

ML learns what makes results relevant:

Input Features

- Query-document similarity
- Document quality/authority
- User location/history
- Freshness of content

Training Signal

- Click-through rates
- Dwell time on page
- Bounce rates
- Manual relevance labels

Applications Beyond Web Search

🛍️ E-commerce Product Search

Rank products by relevance, price, reviews, availability

📧 Email Priority Inbox

Rank emails by importance to user

📱 App Store Rankings

Rank apps by relevance, quality, popularity

💼 Job Matching

Rank candidates or positions by fit

Test Your Understanding

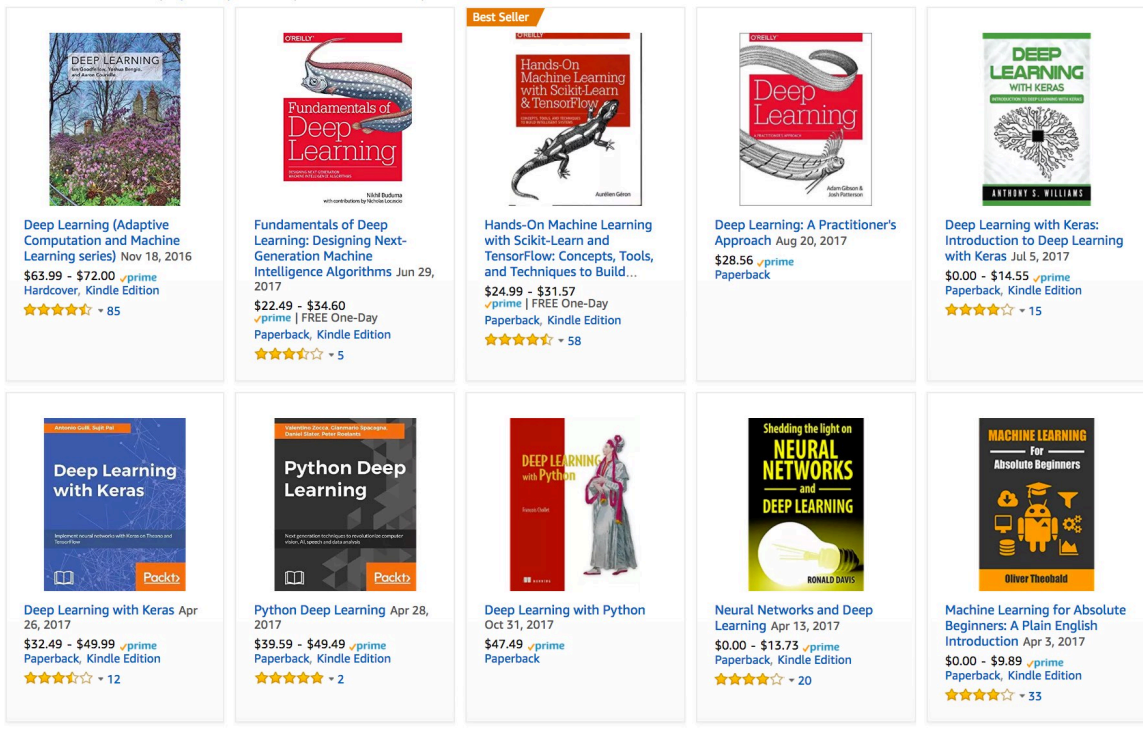
Which statements about search and ranking are correct?

- [ ] Click-through rate is a useful training signal for learning to rank
- [ ] Learning to rank can use multiple features beyond text matching
- [ ] All users should see the same ranking for the same query
- [ ] Ranking is about ordering results by relevance, not just finding them

Recommender Systems

Personalized Suggestions

Why Recommendations Matter



- 35% of Amazon purchases from recommendations
- 75% of Netflix viewing from recommendations
- Crucial for user engagement

Sources: Collaborative Filtering for Implicit Feedback Datasets (http://yifanhu.net/PUB/cf.pdf)

Two Main Approaches

Collaborative Filtering

"Users like you also liked..."

- Based on user behavior
- Find similar users
- Recommend their favorites
- No content analysis needed

Content-Based

"Because you liked X..."

- Analyze item features
- Find similar items
- Match user preferences
- Works for new items

Modern systems combine both approaches

The Cold Start Problem

How to recommend when you have no data?

- **New Users:** No history to learn from
- **New Items:** No ratings or interactions

Solutions:

- Ask for preferences during onboarding
- Use demographic information
- Show popular items initially
- Use content features for new items

Matrix Factorization

Core technique for collaborative filtering:

User-Item matrix → User factors × Item factors

Learns latent features automatically

The Danger of Recommendation Rabbit Holes

Filter Bubbles

Users only see content similar to past behavior

Limits exposure to diverse perspectives

Echo Chambers

Reinforces existing beliefs and biases

Can amplify misinformation

Addiction Patterns

Optimizing for engagement can exploit psychology

Particularly harmful for vulnerable users

> ⚠️ Recommender systems shape what billions of people see online

Test Your Understanding

Which statements about recommender systems are correct?

- [ ] Matrix factorization discovers latent features from user-item interactions
- [ ] Content-based recommendations work well for new items
- [ ] Collaborative filtering needs user behavior data to work
- [ ] The cold start problem only affects new users

Sequence Learning

Temporal Dependencies

When Order Matters

Sequential data is everywhere:

- **Language:** Word order determines meaning
- **Time Series:** Stock prices, weather patterns
- **Audio/Video:** Temporal progression
- **User Behavior:** Click sequences, navigation paths

> "Time is an arrow" ≠ "Arrow an is time"

Challenges of Sequential Data

- **Variable Length:** Sequences can be any length
- **Long-Range Dependencies:** Early events affect later ones
- **Temporal Patterns:** Trends, seasonality, cycles
- **Context Window:** How much history to consider?

Types of Sequence Problems

One-to-Many

Image → Caption (words)

Many-to-One

Sentence → Sentiment

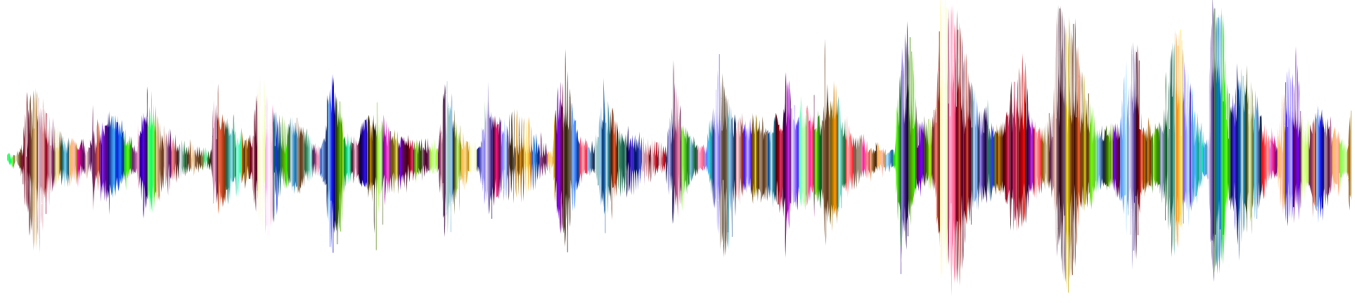
Many-to-Many (Synchronized)

Video → Frame labels

Many-to-Many (Sequence-to-Sequence)

English text → French text

Example: Speech Recognition



Audio waveform → Text transcription

- Variable length input and output
- Temporal alignment challenges
- Context helps disambiguate

Recurrent Neural Networks Preview

Networks with memory:

```
For each time step t:
    hidden_state[t] = f(input[t], hidden_state[t-1])
    output[t] = g(hidden_state[t])
```

- Maintains state across time steps
- Can handle variable length sequences
- Foundation for LSTMs and GRUs

> Later evolved into Transformers (ChatGPT, etc.)

Test Your Understanding

Which statements about sequence learning are correct?

- [ ] Machine translation is a sequence-to-sequence problem
- [ ] Recurrent networks maintain a hidden state across time steps
- [ ] Order of elements is crucial in sequential data
- [ ] All sequences must have the same length for ML models

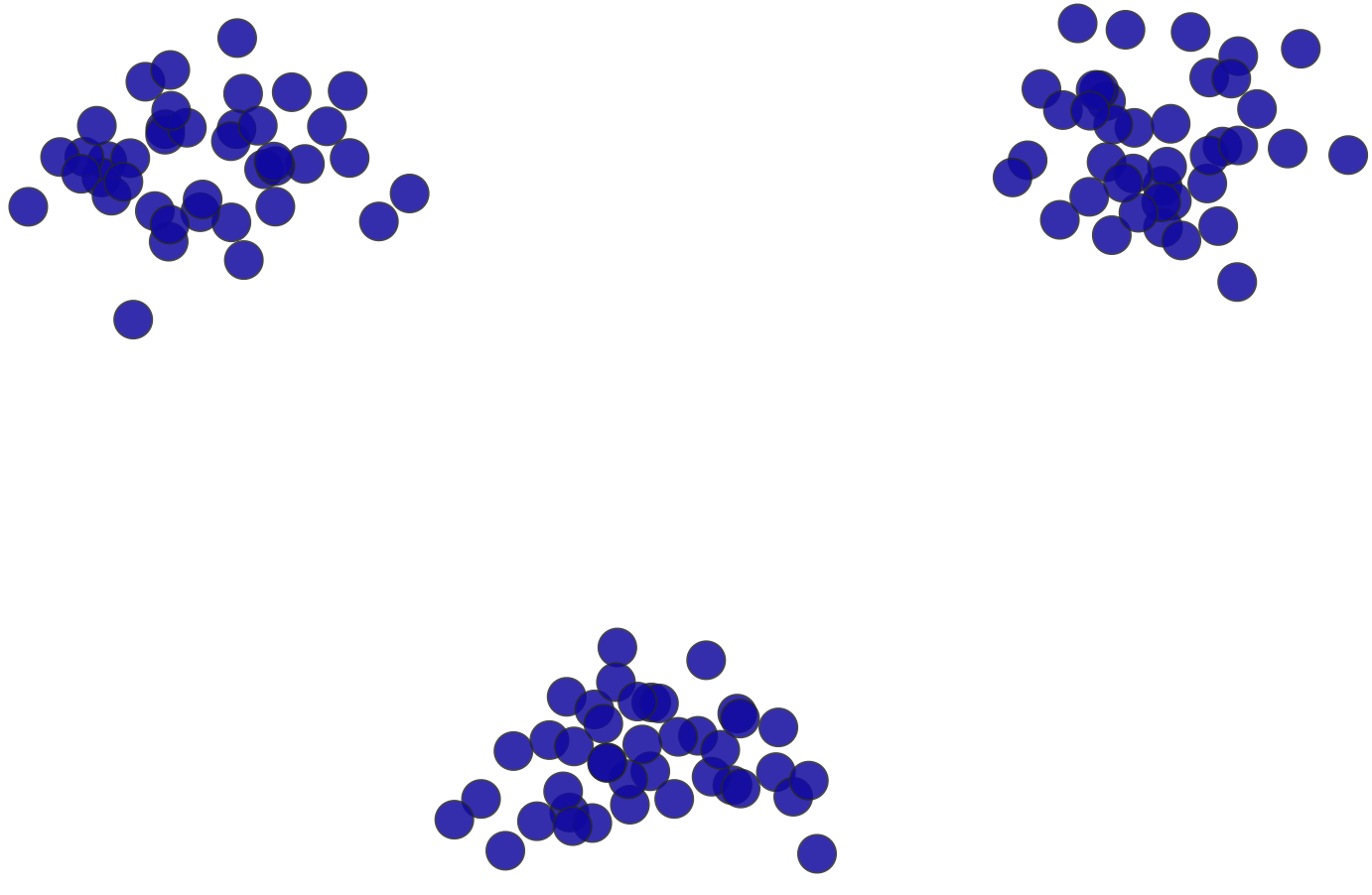
Unsupervised Learning

Finding Hidden Patterns

Learning Without Labels

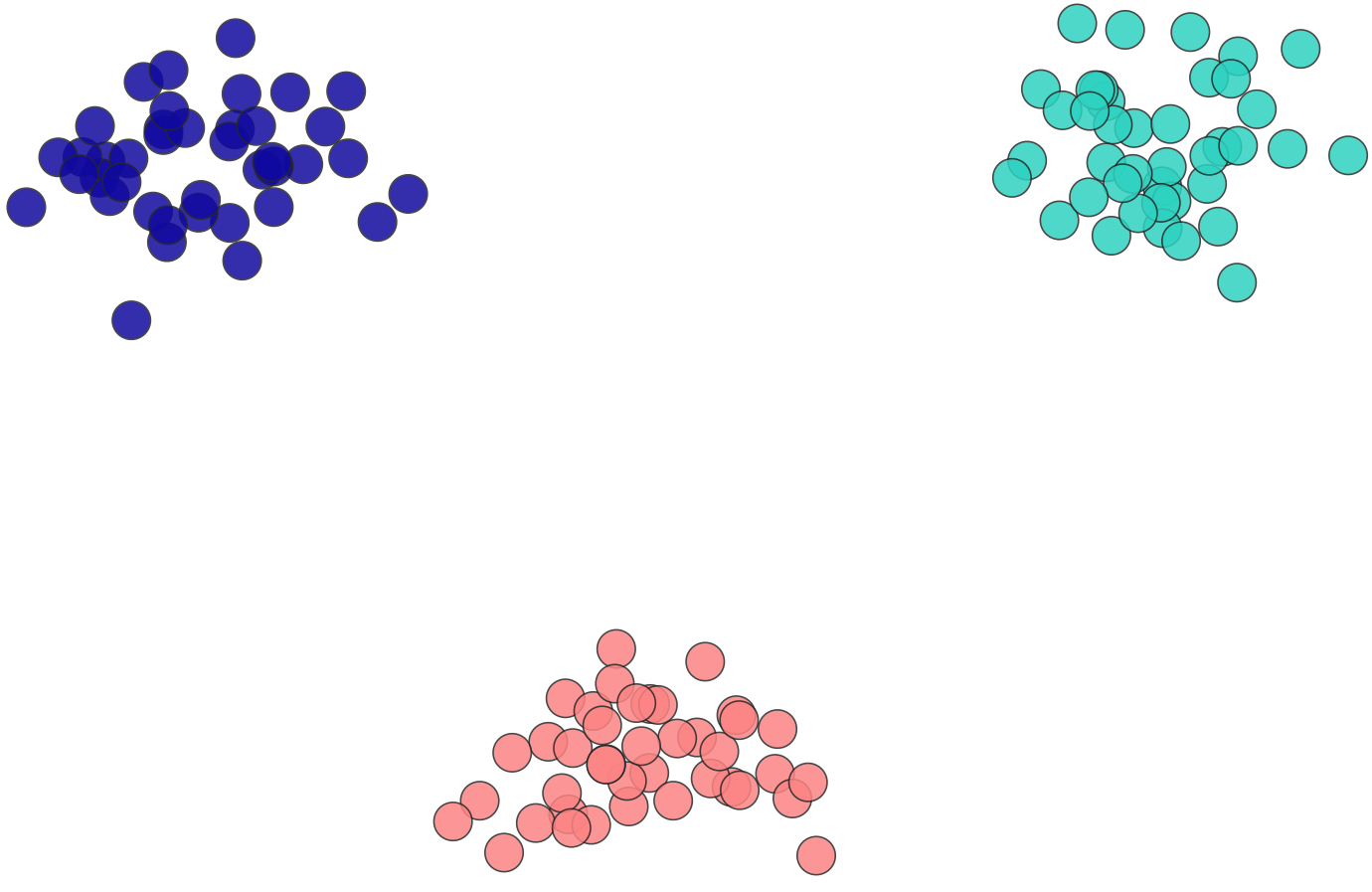
- No correct answers provided
- Find structure in data
- Often used for exploration

Clustering Example



Can you spot the groups?

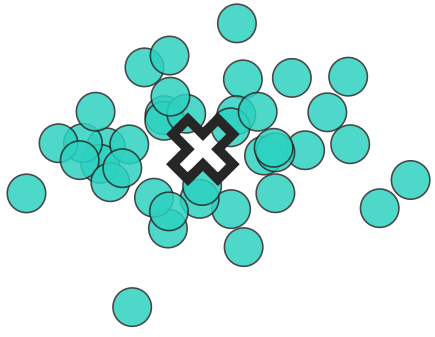
The Hidden Clusters

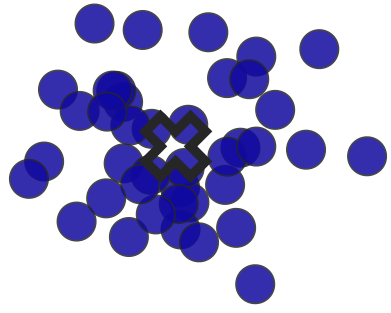


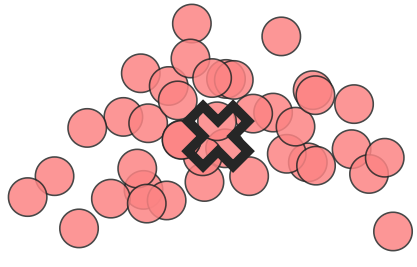
Groups emerge naturally — these points were sampled from three different distributions

Try It: K-Means Clustering

Interactive applet in web based slides







The algorithm finds cluster assignments without ever seeing the true labels

Dimensionality Reduction

- Compress high-dimensional data
- Preserve important information
- Enable visualization
- What are examples of dimensionality reduction methods?

Masked Language Modeling

Foundation of modern NLP (BERT, GPT)

```
The cat sat on [MASK] mat → the
```

- Hide parts of the input text
- Model learns to predict missing words
- No manual labels needed!

> Powers embedding models such as BERT and other variations power GPT models

Masked Image Modeling

Self-supervised learning for computer vision

Original → Masked → Reconstructed

- Mask random patches of images
- Model learns visual representations
- Used in MAE, SimMIM, and others

Achieves state-of-the-art with less labeled data

Sources: Masked Autoencoders Are Scalable Vision Learners (MAE) - He et al., 2022 (https://arxiv.org/abs/2111.06377) · SimMIM: A Simple Framework for Masked Image Modeling - Xie et al., 2022 (https://arxiv.org/abs/2111.09886) · BEiT: BERT Pre-Training of Image Transformers - Bao et al., 2022 (https://arxiv.org/abs/2106.08254)

Self-Supervised Learning

- Create labels from data itself
- Example: predict next word
- Foundation of modern NLP

Unsupervised Applications

- Customer segmentation
- Anomaly detection
- Data compression
- Feature learning

Test Your Understanding

Which statements about unsupervised learning are correct?

- [ ] Clustering algorithms group similar data points without predefined categories
- [ ] Self-supervised learning creates its own labels from the data
- [ ] Unsupervised learning requires more labeled data than supervised learning
- [ ] Dimensionality reduction helps visualize high-dimensional data

Reinforcement Learning

Learning from Interaction

Beyond Supervised Learning

Not all learning needs labeled data

- **Unsupervised Learning:** Find patterns without labels
- **Self-Supervised:** Create labels from data itself
- **Reinforcement:** Learn from interaction

Unsupervised Learning

"Do some data science with it!"

- Clustering: Group similar items
- Dimensionality Reduction: Find key features
- Density Estimation: Learn data distribution
- Anomaly Detection: Find outliers

No labels needed - let the data speak for itself!

Sources: Dive into Deep Learning - Unsupervised Learning (https://d2l.ai/chapter_introduction/index.html#unsupervised-and-self-supervised-learning)

Self-Supervised Learning

Creating supervision from the data itself

Text: Mask words and predict them

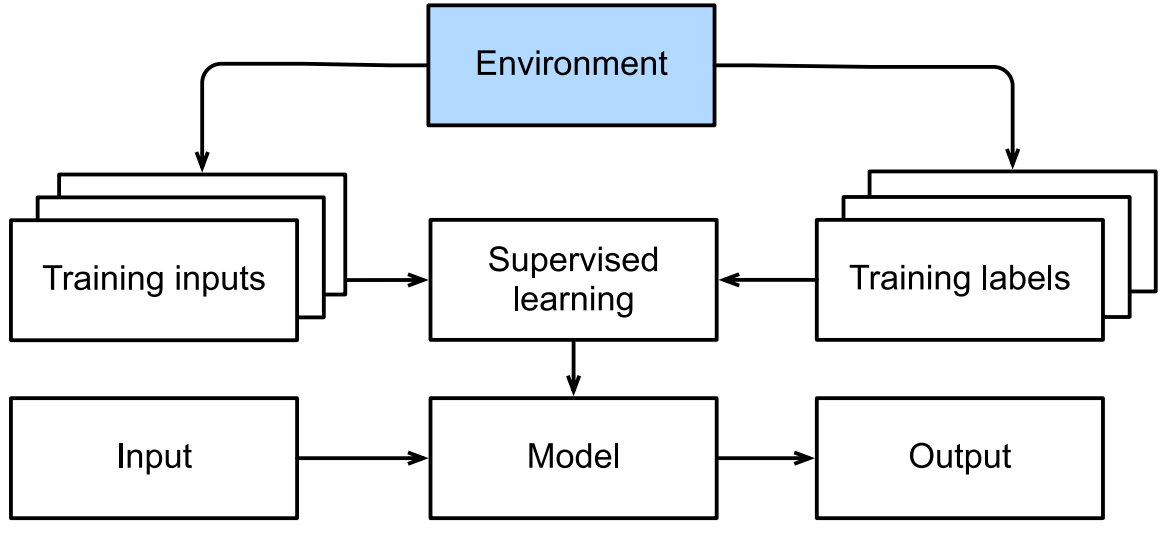
```
"The [MASK] jumped over the fence" → "dog"
```

Images: Predict relative positions or masked patches

Learn representations without manual labeling!

Sources: BERT: Pre-training of Deep Bidirectional Transformers - Devlin et al., 2018 (https://arxiv.org/abs/1810.04805) · Unsupervised Visual Representation Learning - Doersch et al., 2015 (https://arxiv.org/abs/1505.05192)

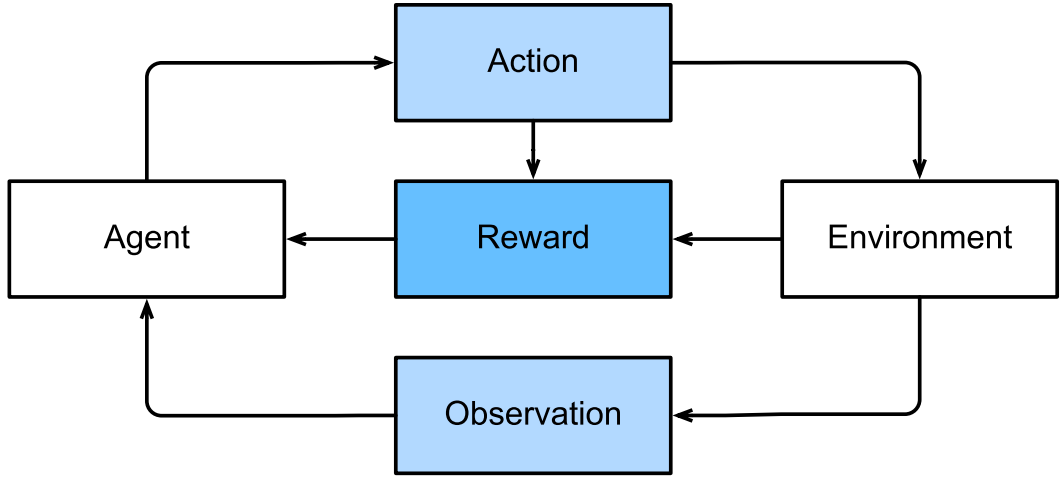
Offline vs Online Learning



Traditional ML: Collect data → Train → Deploy

But what if our model affects the environment?

Reinforcement Learning



- Agent takes **actions**
- Environment provides **rewards**
- Learn through **trial and error**

> Actions impact future observations!

Sources: Reinforcement Learning: An Introduction - Sutton & Barto (http://incompleteideas.net/book/the-book.html)

Unsupervised and Self-Supervised Learning

Learning without explicit labels

Unlike supervised learning with its "dictatorial boss" telling you exactly what to do...

Unsupervised learning is like having a boss who says:
"*Here's data - do some data science with it!*"

Sources: Dive into Deep Learning - Chapter 1.3.2 (https://d2l.ai/chapter_introduction/index.html#unsupervised-and-self-supervised-learning)

Key Unsupervised Learning Tasks

- **Clustering**
  Group similar data points together
- **Dimensionality Reduction**
  Find key parameters (PCA, subspace estimation)
- **Representation Learning**
  Map objects to meaningful vectors
- **Causality Discovery**
  Find root causes in data

Sources: Principal Component Analysis (https://en.wikipedia.org/wiki/Principal_component_analysis) · Clustering Algorithms (https://scikit-learn.org/stable/modules/clustering.html)

Self-Supervised Learning

Creating supervision from the data itself

Text

Fill in the [MASK]

"The cat sat on the [MASK]"
→ predict: "mat"

Images

Predict relative positions

Two image patches
→ predict: above/below/left/right

💡 No manual labeling required - the data provides its own supervision!

Sources: BERT: Pre-training of Deep Bidirectional Transformers - Devlin et al., 2018 (https://arxiv.org/abs/1810.04805) · Unsupervised Visual Representation Learning by Context Prediction - Doersch et al., 2015 (https://arxiv.org/abs/1505.05192)

Deep Generative Models

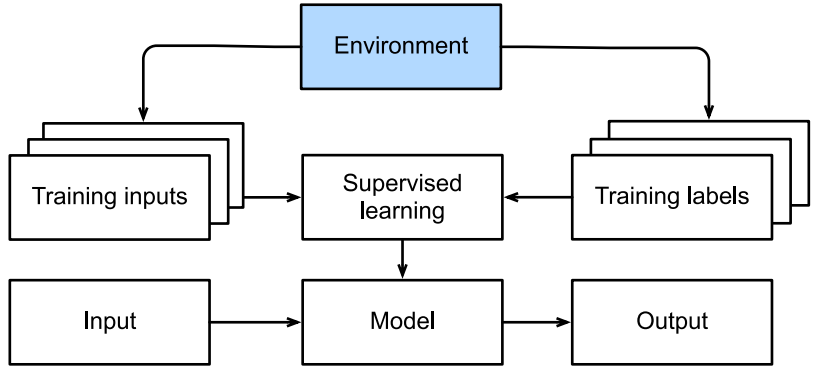
Learning to generate new data

- Variational Autoencoders (VAEs) - 2014
- Generative Adversarial Networks (GANs) - 2014
- Normalizing Flows - 2014-2017
- Diffusion Models - 2020-2021

These models learn data distributions and can generate new, realistic samples

Sources: Auto-Encoding Variational Bayes - Kingma & Welling, 2014 (https://arxiv.org/abs/1312.6114) · Generative Adversarial Nets - Goodfellow et al., 2014 (https://arxiv.org/abs/1406.2661) · Denoising Diffusion Probabilistic Models - Ho et al., 2020 (https://arxiv.org/abs/2006.11239)

Interacting with an Environment



Supervised Learning: Collect data → Train offline → Deploy

But real agents need to:

- Take actions that affect the environment
- Learn from consequences of actions
- Adapt to changing conditions

This leads us to Reinforcement Learning...

Test Your Understanding

Which statements about unsupervised and self-supervised learning are correct?

- [ ] Self-supervised learning creates supervision signals from the data itself
- [ ] Unsupervised learning requires labeled training data
- [ ] PCA is a supervised learning technique
- [ ] Deep generative models can create new data samples
- [ ] Clustering is an example of unsupervised learning

Historical Roots

Where It All Began

The Desire to Predict

Humans have sought to analyze data and predict outcomes for centuries

- Bernoulli distribution (1655-1705)
- Gaussian distribution (1777-1855)
- Least mean squares algorithm

> These tools enabled the experimental approach in natural sciences

Sources: Dive into Deep Learning - Roots (https://d2l.ai/chapter_introduction/index.html#roots)

Medieval Statistics: Köbel's Algorithm

Jacob Köbel (1460-1533): Early data collection

1. Measure 16 adult men's feet
2. Sum the measurements
3. Divide by 16 for average
4. Improvement: Remove outliers (shortest/longest)

> One of the earliest examples of a trimmed mean!

The Rise of Modern Statistics

Ronald Fisher (1890-1962)

- Linear discriminant analysis
- Fisher information matrix
- Applications in genetics
- The famous **Iris dataset** (1936)

⚠️ Note: Fisher was also a proponent of eugenics, reminding us that data science can be misused

Sources: Ronald Fisher - Wikipedia (https://en.wikipedia.org/wiki/Ronald_Fisher) · The Iris Dataset (https://archive.ics.uci.edu/ml/datasets/iris)

Information Theory & Computation

Claude Shannon (1916-2001)

Information theory

- Entropy
- Channel capacity
- Data compression

Alan Turing (1912-1954)

"Can machines think?"

- Turing test
- Computability
- Machine intelligence

Sources: Claude Shannon - Information Theory (https://en.wikipedia.org/wiki/Claude_Shannon) · Computing Machinery and Intelligence - Turing, 1950 (https://doi.org/10.1093/mind/LIX.236.433)

Biological Inspiration

Donald Hebb (1904-1985)

"*Neurons that fire together, wire together*"

- Hebbian learning rule
- Positive reinforcement of connections
- Foundation for gradient descent
- Inspired Rosenblatt's perceptron

The biological metaphor gave "neural networks" their name

Sources: The Organization of Behavior - Hebb, 1949 (https://doi.org/10.1002/1097-4679(195007)6:3%3C307::AID-JCLP2270060338%3E3.0.CO;2-K) · The Perceptron - Rosenblatt, 1958 (https://doi.org/10.1037/h0042519)

Core Neural Network Principles

Key ideas that persist today:

- **Layers:** Alternating linear and nonlinear processing
- **Backpropagation:** Using chain rule to adjust all parameters
- **Networks:** Interconnected computational units

But from 1995-2005, neural networks fell out of favor...

Sources: Early Neural Network Models (https://en.wikipedia.org/wiki/Artificial_neural_network#History)

The AI Winter (1995-2005)

Why neural networks struggled:

- **Computational expense:** Training was too slow
- **Small datasets:** MNIST's 60,000 digits was "huge"
- **Better alternatives:**
  - Kernel methods (SVMs)
  - Decision trees
  - Graphical models

> These methods gave predictable results with theoretical guarantees

Test Your Understanding

Which historical contributions are fundamental to modern machine learning?

- [ ] The Hebbian learning rule (1949) inspired gradient descent
- [ ] The trimmed mean of Köbel (1535) for robust statistics
- [ ] The least squares method of Gauss (1805) for optimization
- [ ] The Iris dataset of Fisher (1936) proved ML was possible

Neural Network History

- 1943: McCulloch-Pitts neuron
- 1958: Perceptron
- 1986: Backpropagation
- 2012: Deep learning revolution

Sources: McCulloch & Pitts (1943) - A Logical Calculus of Ideas (https://doi.org/10.1007/BF02478259) · Rosenblatt (1958) - The Perceptron (https://doi.org/10.1037/h0042519) · Rumelhart et al. (1986) - Learning representations by back-propagating errors (https://doi.org/10.1038/323533a0) · Krizhevsky et al. (2012) - ImageNet Classification with Deep CNNs (https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html)

The Deep Learning Era

- Enabled by GPUs
- Big data availability
- Algorithmic improvements
- Transforming every field

Test Your Understanding

Which statements about the history of machine learning are correct?

- [ ] Backpropagation (1986) enabled training of multi-layer neural networks
- [ ] The deep learning revolution began around 2012 with GPU acceleration
- [ ] Linear regression, developed in 1805, forms a foundation for modern ML
- [ ] The perceptron was the first successful deep neural network

The Road to Deep Learning

The confluence of big data, cheap storage, and GPU computing enabled the deep learning revolution

Data vs Compute Evolution

1970	100 (Iris)	1 KB	100K (Intel 8080)
1980	1K (Boston Housing)	100 KB	1M (Intel 80186)
1990	10K (MNIST)	10 MB	10M (Intel 80486)
2000	10M (Web pages)	100 MB	1G (Intel Core)
2010	10G (Advertising)	1 GB	1T (NVIDIA C2050)
2020	1T (Social networks)	100 GB	1P (NVIDIA DGX-2)

Key Algorithmic Breakthroughs

- **Dropout (2014)**: Regularization by noise
- **Attention (2014)**: Learnable memory pointers
- **Transformers (2017)**: Attention-only architecture
- **Scaling Laws (2020)**: Predictable improvements with size

Sources: Srivastava et al. (2014) - Dropout (https://jmlr.org/papers/v15/srivastava14a.html) · Bahdanau et al. (2014) - Attention Mechanisms (https://arxiv.org/abs/1409.0473) · Vaswani et al. (2017) - Attention Is All You Need (https://arxiv.org/abs/1706.03762)

Language Model Revolution

- **GPT-3 (2020)**: 175B parameters, few-shot learning
- **ChatGPT (2022)**: RLHF
- **GPT-4 (2023)**: Multimodal capabilities
- **Open Models**: LLaMA, Mistral democratizing AI

Sources: Brown et al. (2020) - GPT-3 (https://arxiv.org/abs/2005.14165) | OpenAI (2023) - GPT-4 (https://arxiv.org/abs/2303.08774) - Touvron et al. (2023) - LLaMA (https://arxiv.org/abs/2302.13971)

Generative Model Revolution

- **GANs (2014)**: Adversarial training
- **Diffusion Models (2020)**: Denoising approach
- **DALL-E 2 (2022)**: Text-to-image generation
- **Stable Diffusion**: Open-source creative AI

Sources: Goodfellow et al. (2014) - GANs (https://arxiv.org/abs/1406.2661), Ho et al. (2020) - Denoising Diffusion (https://arxiv.org/abs/2006.11239), Ramesh et al. (2022) - DALL-E 2 (https://arxiv.org/abs/2204.06125)

Distributed Training Breakthroughs

- Training on 1000+ GPUs simultaneously
- Batch sizes: 32 → 64,000 images
- ResNet-50 training: Days → 7 minutes
- Enabling massive model scale

Sources: You et al. (2017) - Large Batch Training (https://arxiv.org/abs/1708.03888), Jia et al. (2018) - ImageNet in 4 Minutes (https://arxiv.org/abs/1807.11205)

Deep Learning Frameworks Evolution

- **Gen 1**: Caffe, Torch, Theano
- **Gen 2**: TensorFlow, Keras, CNTK
- **Gen 3**: PyTorch, JAX
- From PhD homework → 10 lines of code

Test Your Understanding

What key factors enabled the deep learning revolution?

- [ ] Memory growth kept pace with dataset growth
- [ ] Deep learning frameworks made implementation accessible
- [ ] GPUs originally designed for gaming provided massive parallel computing
- [ ] Attention mechanisms solved the long-sequence problem

AI Success Stories

From hidden applications to headline-grabbing achievements

Traditional ML Applications

- OCR - Source of MNIST dataset
- Check reading & credit scoring
- Fraud detection (PayPal, Stripe, Visa)
- Search, recommendations, ranking

ML has been pervasive, albeit often hidden

Sources: MNIST Dataset History (http://yann.lecun.com/exdb/mnist/)

Intelligent Assistants

- **Siri, Alexa, Google Assistant** (before 2022)
- **ChatGPT, Claude, Gemini** (after 2022)
- Speech recognition: Human parity (2018)
- From light switches to appointments
- Most visible AI in daily life

Sources: Xiong et al. (2018) - Microsoft Speech Recognition (https://arxiv.org/abs/1708.02021)

Computer Vision Breakthroughs

2010	28%	Traditional ML
2012	16%	AlexNet
2015	3.5%	ResNet
2017	2.25%	SENet

Also: Birdsong ID, skin cancer diagnosis

Sources: Lin et al. (2010) - ImageNet Baseline (https://ieeexplore.ieee.org/document/5540121) · Hu et al. (2018) - SENet (https://arxiv.org/abs/1709.01507)

Game-Playing AI Milestones

- **1992**: TD-Gammon (Backgammon)
- **1997**: Deep Blue (Chess)
- **2016**: AlphaGo (Go)
- **2017**: Libratus (Poker)
- **2019**: AlphaStar (StarCraft II)

Sources: Campbell et al. (2002) - Deep Blue (https://doi.org/10.1016/S0004-3702(01)00129-1) · Silver et al. (2016) - AlphaGo (https://doi.org/10.1038/nature16961) · Brown & Sandholm (2017) - Libratus (https://www.ijcai.org/proceedings/2017/772)

Self-Driving Vehicles

- Tesla Autopilot, Waymo, NVIDIA Drive
- Deep learning for perception
- Challenges: Reasoning, rule incorporation
- Partial autonomy achieved, full autonomy pending

Scientific Applications

- **Biology**: AlphaFold
- **Physics**: Particle detection, gravitational waves
- **Astronomy**: Galaxy classification, exoplanet discovery
- **Medicine**: Drug discovery, diagnosis
- **Climate**: Weather prediction, climate modeling

AI Ethics & Concerns

- Not AGI: Task-specific, engineered systems
- Real concerns:
  - Job automation impact
  - Algorithmic bias
  - Privacy & surveillance
  - Decision transparency
- Need careful, ethical deployment

Test Your Understanding

Which statements about AI success stories are true?

- [ ] ImageNet error rates dropped from 28% to 2.25% between 2010-2017
- [ ] Full self-driving has been achieved by multiple companies
- [ ] Speech recognition has achieved human parity for some applications
- [ ] AI systems today have artificial general intelligence

The Essence of Deep Learning

What makes deep learning *deep* and why it revolutionized AI

What Makes It "Deep"?

- Many layers of transformations
- Operations at each layer learned *jointly* from data
- Not just stacked processing steps
- Depth enables hierarchical feature learning

End-to-End Training Revolution

Traditional ML Pipeline

1. Feature Engineering
2. Feature Selection
3. Model Training
4. Separate tuning

Deep Learning Pipeline

1. Raw Data Input
2. Learned Features
3. Joint Optimization
4. End-to-end training

Sources: Canny (1987) - Edge Detection (https://doi.org/10.1109/TPAMI.1986.4767851) · Lowe (2004) - SIFT Features (https://doi.org/10.1023/B:VISI.0000029664.99615.94)

From Manual to Learned Features

- **Before**: Domain experts design features
- **Problem**: Human ingenuity is limited
- **Deep Learning**: Automatic feature discovery
- **Result**: Superior accuracy across domains

Millions of automatic choices beat manual design

Unified Tools Across Domains

- Same architectures work for:
  - Computer Vision
  - Natural Language Processing
  - Speech Recognition
  - Medical Imaging
- Eliminated domain-specific boundaries
- Transfer learning across modalities

Parametric → Nonparametric

- **Scarce data era**: Simplifying assumptions needed
- **Big data era**: Let data speak for itself
- Like physics: Analytical → Numerical simulations
- Trade-off: Accuracy vs Interpretability

The Empirical Revolution

- Accept suboptimal solutions
- Embrace nonconvex optimization
- Try first, prove later
- Rapid experimentation culture

Open Source Culture

- Shared tools across academia & industry
- Released models & datasets
- Collaborative progress
- Lowered barriers to entry

From PhD homework → 10 lines of code

Test Your Understanding

What are the key characteristics that define deep learning?

- [ ] End-to-end training from raw data to outputs
- [ ] Requires theoretical proofs before implementation
- [ ] Unified tools working across different domains
- [ ] Many layers of transformations learned jointly from data
- [ ] Manual feature engineering followed by neural networks

Key Concepts Recap

- ML = Programming with data
- Four components: data, models, objectives, optimization
- Three paradigms: supervised, unsupervised, reinforcement
- Wide range of applications

What's Next?

- Mathematical foundations
- Linear models in depth
- Neural network basics
- Hands-on implementation