ECG782: MULTIDIMENSIONAL DIGITAL SIGNAL PROCESSING

THE MACHINE LEARNING LANDSCAPE

Géron Chapter 1

http://www.ee.unlv.edu/~b1morris/ecg782

OUTLINE

- What is Machine Learning (ML)
- Why Use ML
- Example Applications
- Types of ML Systems
- Main Challenges of ML
- Testing and Validating

WHAT IS ML?

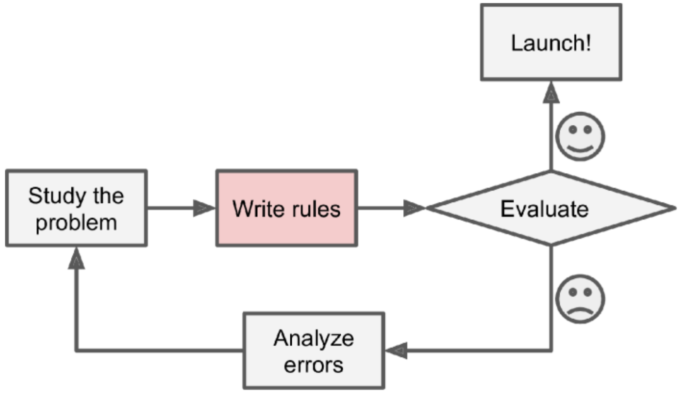
- [Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed. – Arthur Samuel, 1959

- A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. – Tom Mitchell, 1997

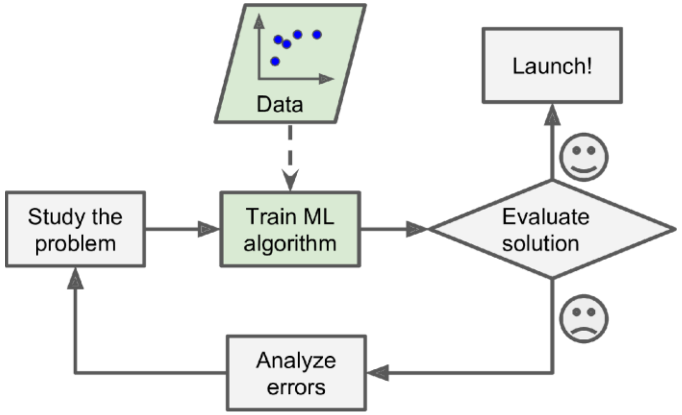
- Machine Learning is the science (and art) of programming computers so they can learn from data
  - Not enough to have lots of data, must be able to use data to solve a task

WHY USE ML?

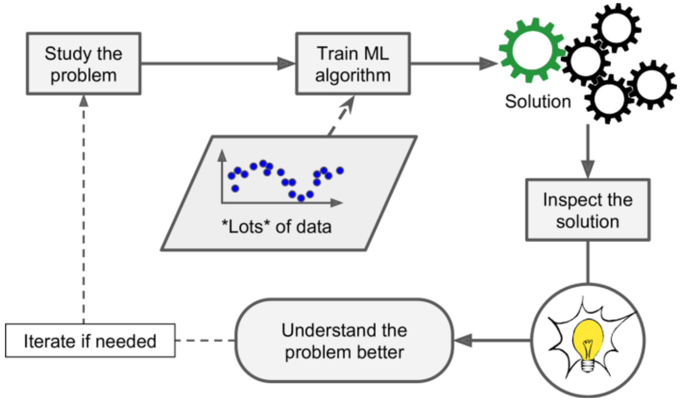
- Traditional approach has complex rules and difficult to maintain



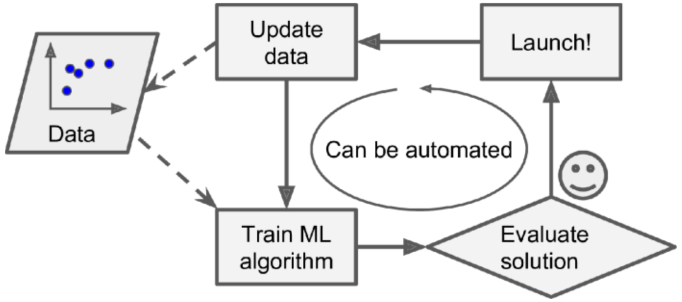
- ML learns from data making code shorter, easier to maintain, and more accurate



- Can inform humans of what was learned for new insight into a problem



- Can automatically be updated to changes



EXAMPLE APPLICATIONS

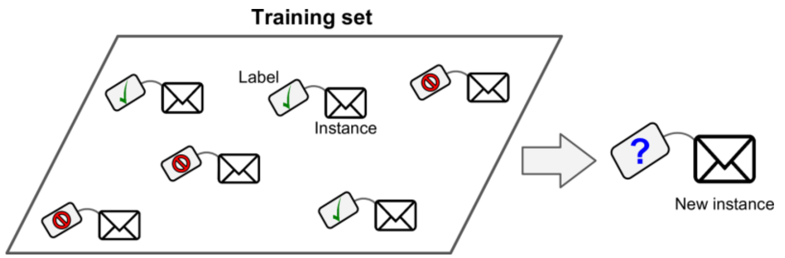
- Analyzing production line images to classify or detecting tumors in brain scans
  - Chapter 14 – convolutional neural networks (CNNs)
- Visual representation of complex, high-dimensional data
  - Chapter 8 – data visualization and data reduction
- Intelligent bot for a game
  - Chapter 18 – reinforcement learning (RL)
- Forecasting future company revenue
  - Chapter 4, 5, 7, 10, 15, 16 – regression using classical (linear/polynomial SVM, Random Forest or deep learning methods)
- News article classification, flagging offensive comments, long document summarization, chatbot creation
  - Chapter 16 – natural language processing (NLP)
- Making an app react to voice commands
  - Chapter 15/16 – recurrent neural networks (RNNs), CNNs, Transformers
- Detecting credit card fraud, segmenting customers for marketing strategy
  - Chapter 9 – anomaly detection, clustering
- Product recommendations based on past purchases
  - Chapter 10 – ANN

TYPES OF ML SYSTEMS

- Broad categories:
  - Training with human supervision (supervised, unsupervised, semi-supervised, and reinforcement learning)
  - Learning incrementally or on the fly (online vs batch learning)
  - Comparison with known data points or by finding patterns in training data to build predictive models (instance-based vs model-based learning)
- Criteria are not exclusive and can be combined

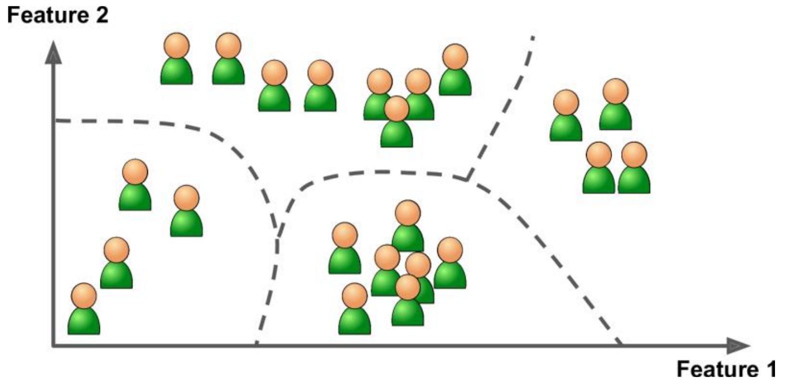
SUPERVISED LEARNING

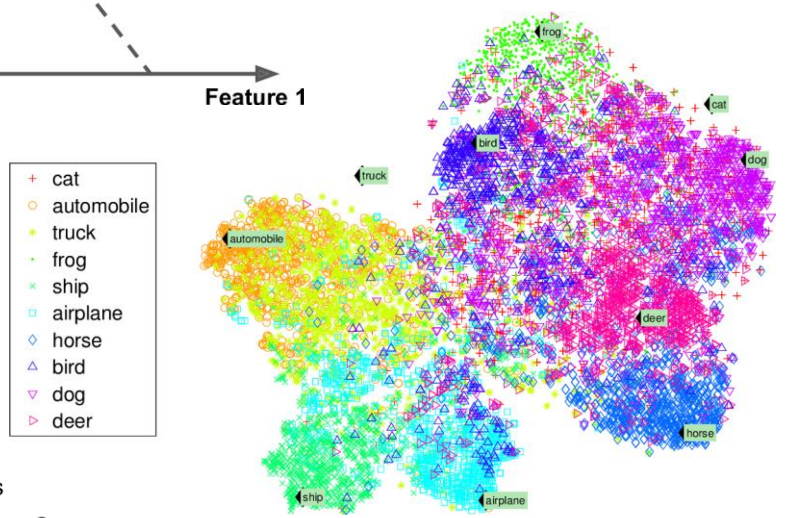
- Training with data and labels (desired solutions)
- Typical tasks
  - Classification: determine data class
    - Spam filter: data=emails, label={spam, not-spam}
  - Regression: predict target numeric value
    - Price of car: data=features (mileage, age, brand, etc.) label=price
- Important algorithms
  - k-NN, linear and logistic regression, SVM, decision trees and random forests, neural networks



UNSUPERVISED LEARNING

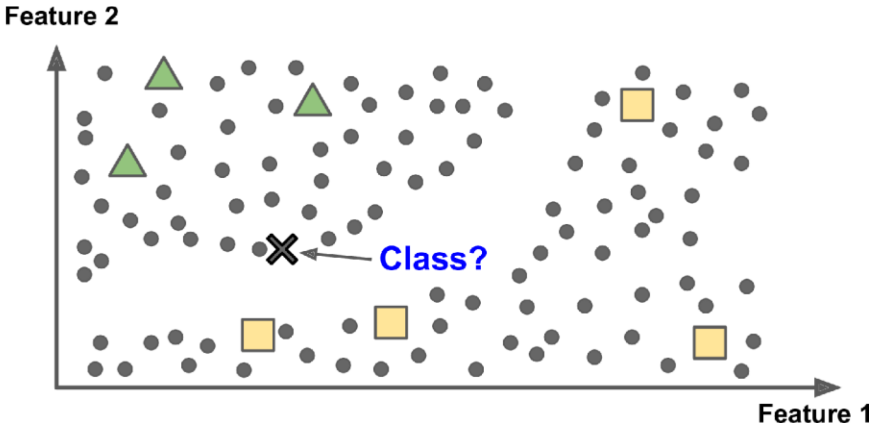
- Training with unlabeled data (no teacher)
- Important algorithms
  - Clustering – discovering groups
    - K-means, DBSCAN
  - Visualization and dimensionality reduction – reduce feature dimension and maintain structure
    - (Kernel) principle component analysis (PCA), LLE, t-SNE
  - Anomaly/novelty detection – find unusual test data
    - One-class SVM, isolation forest
  - Association rule learning – find relations between features
    - Apriori, Eclat





SEMI-SUPERVISED LEARNING

- Training with partially labeled data
  - Lots of unlabeled and few labeled instances

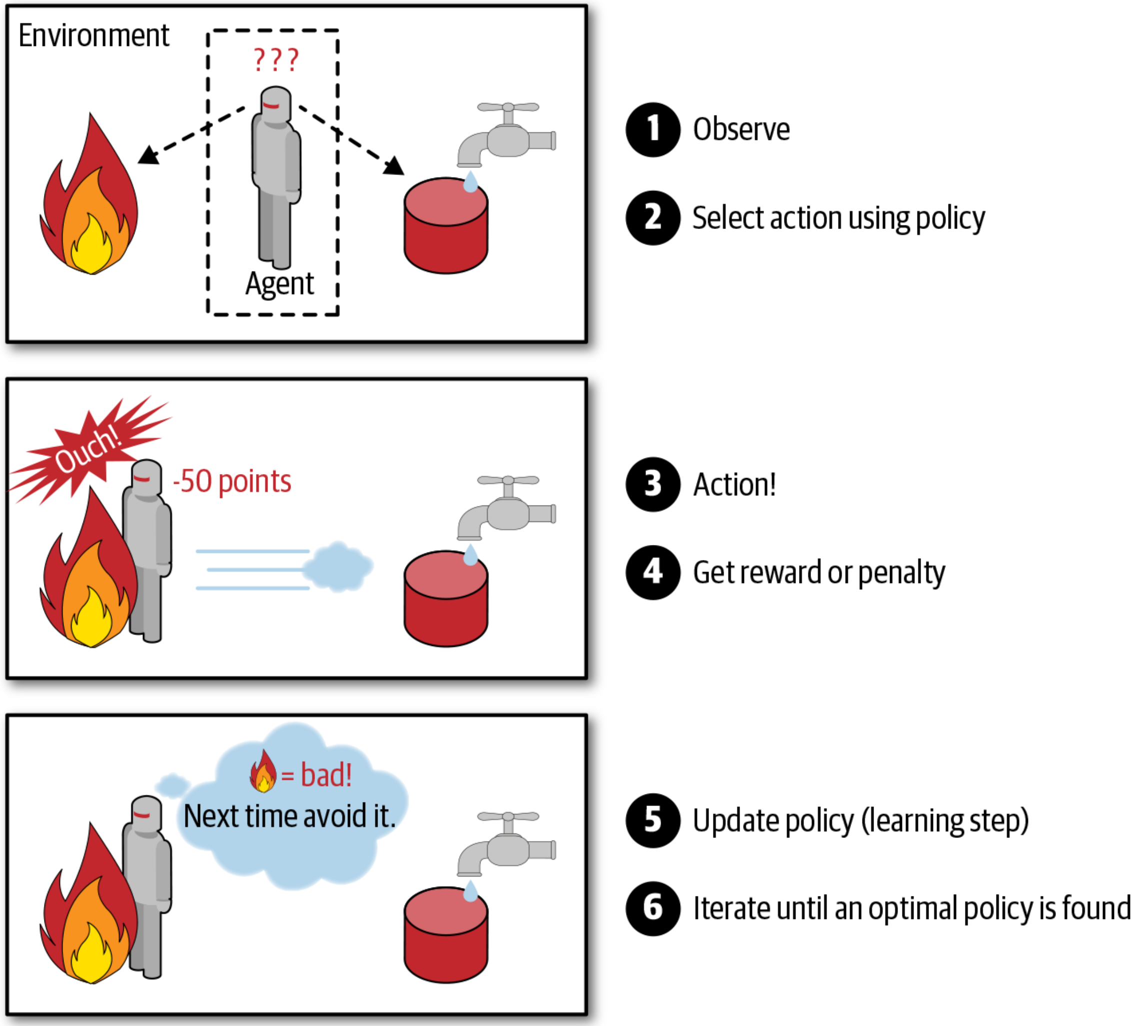


- Most are combination of unsupervised and supervised algorithms
  - Deep belief networks (DBNs) are based on stacked unsupervised restricted Boltzmann machines (RBMs) that are fine-tuned using supervised learning techniques

- Example: Google Photos
  - Given large personal photo library
  - Automatically cluster photos into groups of people
  - Supervision when specify name of group
    - Need to merge and split groups to fine-tune

REINFORCEMENT LEARNING

- Agent-based learning paradigm
  - Agent – learning system that can observe environment, select and perform actions, and get rewards
  - Rewards/penalties – "value" associated with actions
  - Policy – strategy, action to choose which is learned to maximize reward over time
- Popular for robotics and game playing
  - E.g. DeepMind's Q-Learning Atari Breakout or AlphaGo

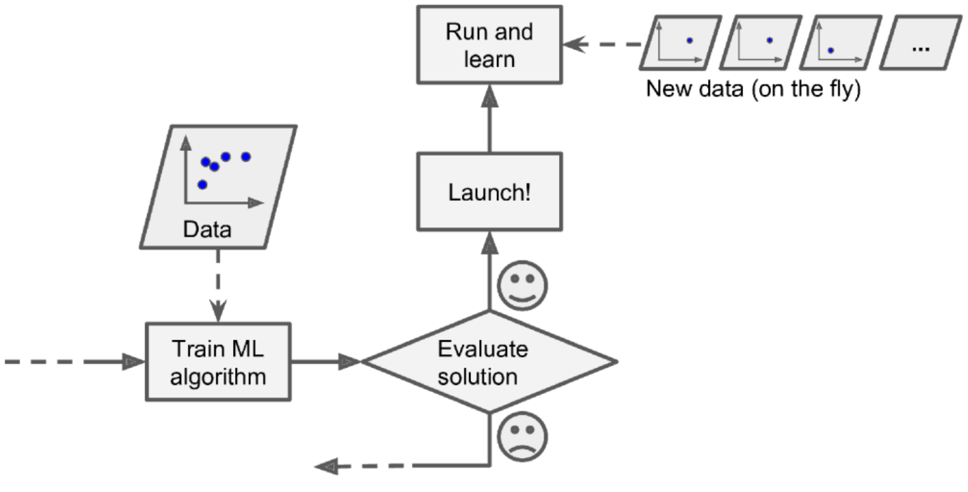


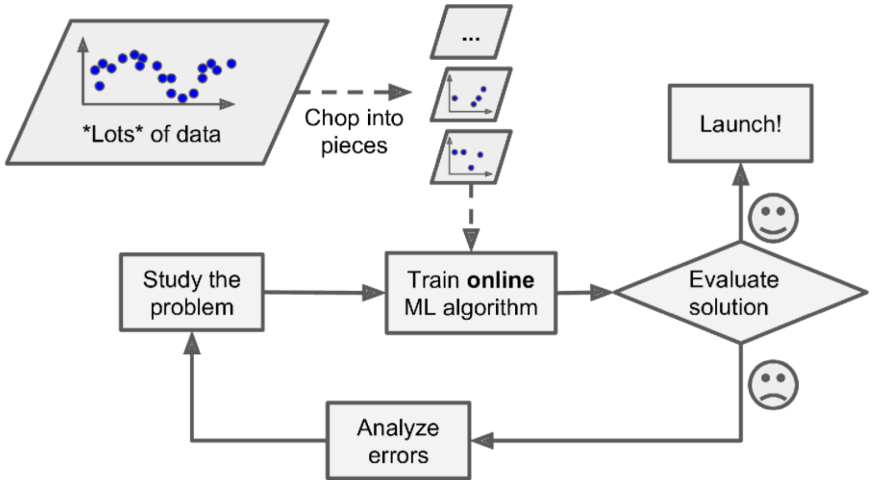
BATCH LEARNING

- Learning uses all available data
- Offline learning – train then launched into production with no more training
  - Often because of heavy time and resource requirements
- Can update model fairly easily by incorporating new data (say every 24 hours)
- Does not work in many situations
  - Rapidly changing data – need to adapt more quickly
  - Big data and computational restrictions – too costly to train, too large to batch, or not enough resources (mobile phone or mars rover)

ONLINE LEARNING

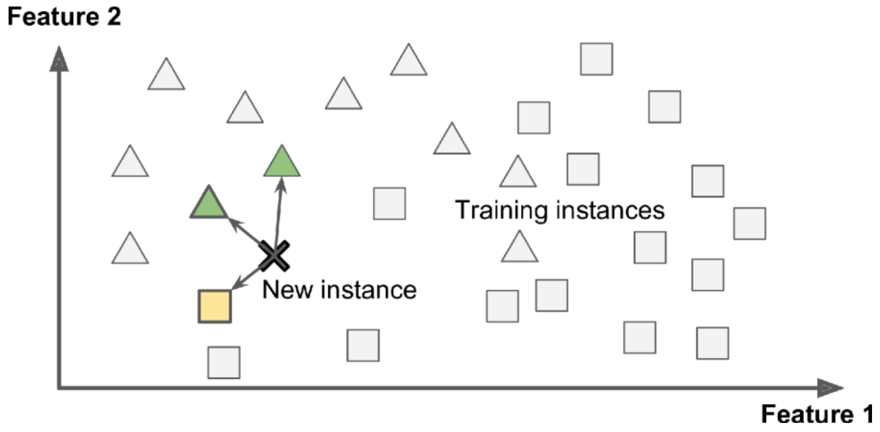
- Incremental training by feeding data instances sequentially
  - Use of mini-batch – small groupings of data
- Well-suited for streaming data or limited computing resources
  - Can react/adapt quickly to changes autonomously
  - Can discard samples after incorporating into model
  - Out-of-core learning for large datasets that do not fit in memory
- Learning-rate – how fast to adapt to changing data
  - High: quickly adapt, but forget old
  - Low: less sensitive to noise/outliers but slower to update (inertia)
- Major challenge: graceful degradation over time
  - How to handle bad data that comes in?





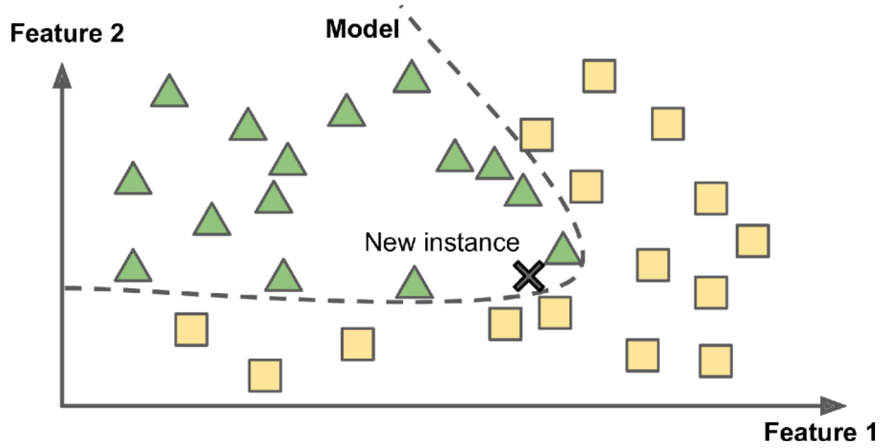
INSTANCE-BASED LEARNING

- System learns examples by-heart, then generalizes to new cases using a similarity measure
  - Simple learning method (e.g k-NN)
  - Needs to store instances (database)
  - Define meaningful similarity measure



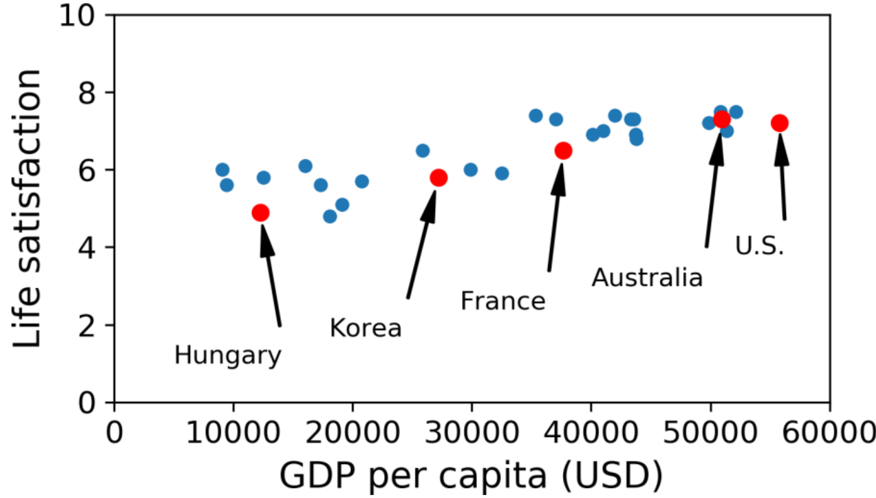
MODEL-BASED LEARNING

- Build model of examples and use model to make predictions

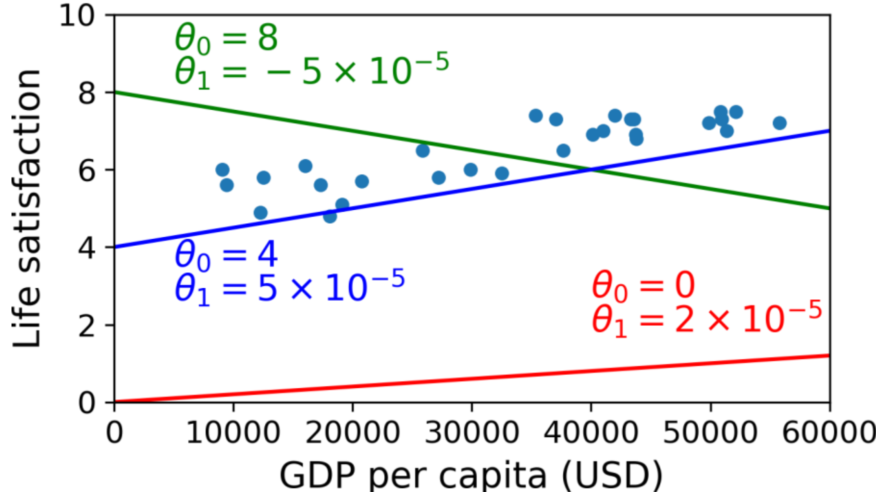


- Need to choose a "model"
- Tune parameters for good fit
  - Define utility/fitness function for goodness or cost function for badness of fit

- Data



- Linear model

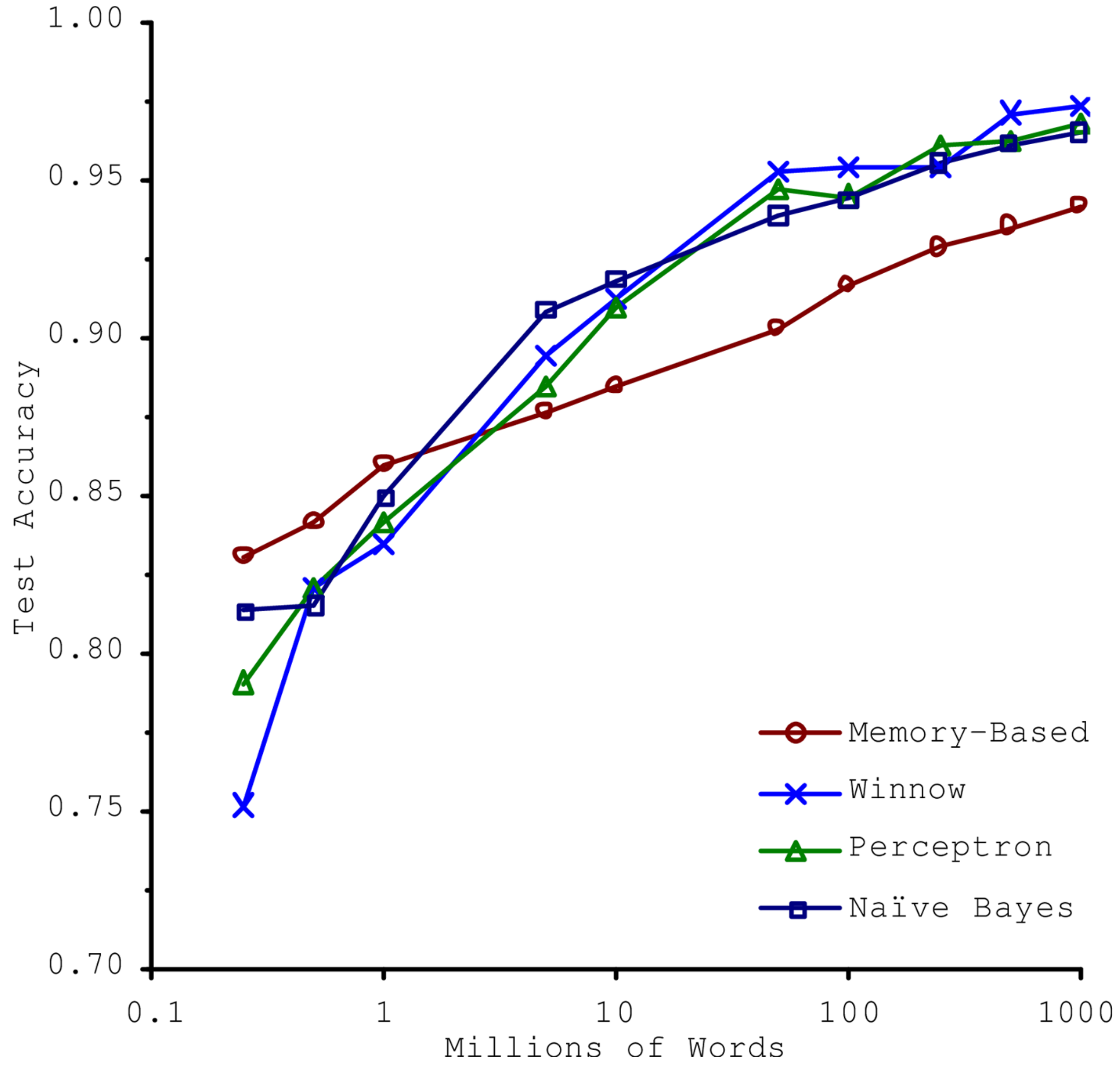


MAIN CHALLENGES OF ML

- “Bad Data”
  - Insufficient quantity of data – not enough
  - Non-representative data – biased data
  - Poor quality data – errors, noise, outliers
  - Irrelevant features – not measuring the right things
- “Bad Algorithms”
  - Overfitting – overreliance on limited training data
  - Underfitting – not enough model capacity

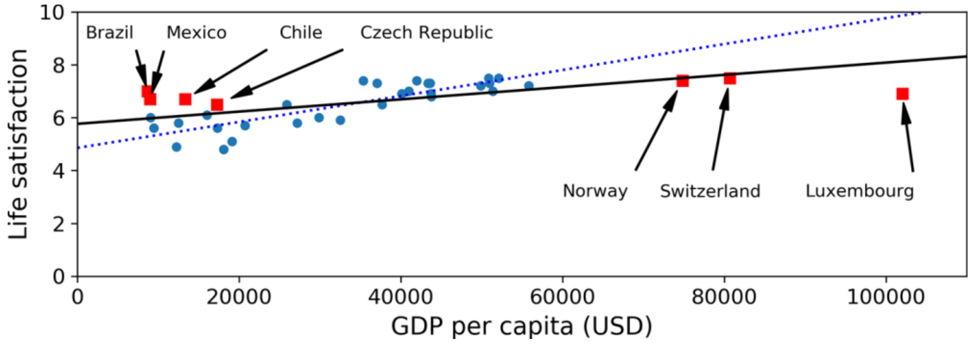
INSUFFICIENT QUANTITY OF TRAINING DATA

- ML still requires a lot of data to work properly
  - 1000s or more (millions for image/speech)
- The Unreasonable Effectiveness of Data
  - Given enough data, very different ML algorithms (including fairly simple) all perform similarly
  - "Reconsider trade-off between spending time and money on algorithm development versus spending it on corpus development"
    - Has led to much of modern ML and computer vision → massive datasets
    - Do we now have enough (too much) data?



NONREPRESENTITIVE TRAINING DATA

- Training data must be representative of test cases to generalize well



  - Dashed blue old model using blue dots
  - Solid line trained using also red squares
  - Poor performance with old model
    - Especially with poor and rich countries

- Sampling noise – nonrepresentative sample data due to chance
- Sampling bias – training samples have systematic issue in collection which produces non-uniformity (or mis-matching of underlying distribution)
  - E.g. facial recognition systems performing poorly on darker skin tones

POOR-QUALITY DATA

- Data full of errors, outliers, and noise (e.g., due to poor-quality measurements
  - Will make it harder to detect underlying patterns and less likely to perform well
- Data scientist spend significant time to cleaning up data
  - Clear outliers – discard or manually fix errors
  - Missing a few features – decide to ignore attribute, instances with "holes", fill in missing value, or train multiple models (with/without missing features)

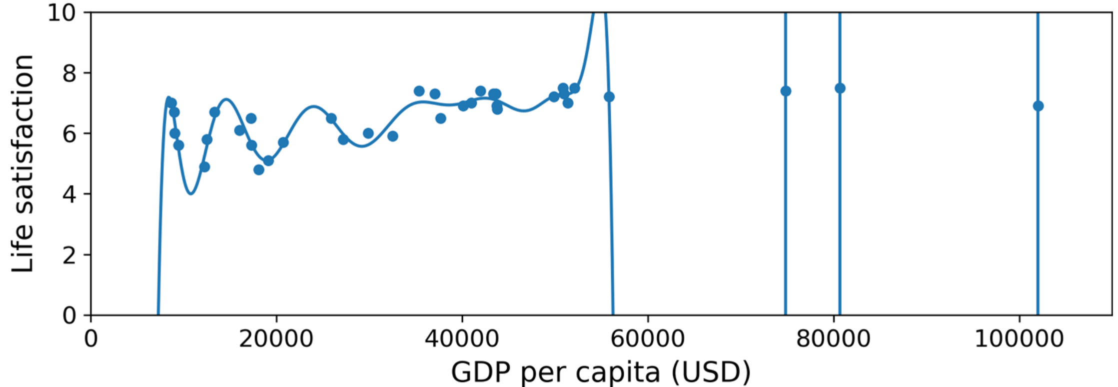
IRRELEVANT FEATURES

- Garbage in, garbage out
  - Can only learn if features are relevant, not too much irrelevant info
- Feature engineering – process of determining a good set of features to train on
  - Feature selection – select most useful features among all available/existing features
  - Feature extraction – combining existing features to produce more useful ones
  - Creating new features by gathering new data
- Classical ML uses “hand-crafted” features while deep learning has data-driven features

OVERFITTING THE TRAINING DATA

- Overgeneralizing based on limited data
  - Model is too complex relative to the amount of noisiness in the training data → modeling noise
  - Good performance on training but poor generalization (bad performance on test)

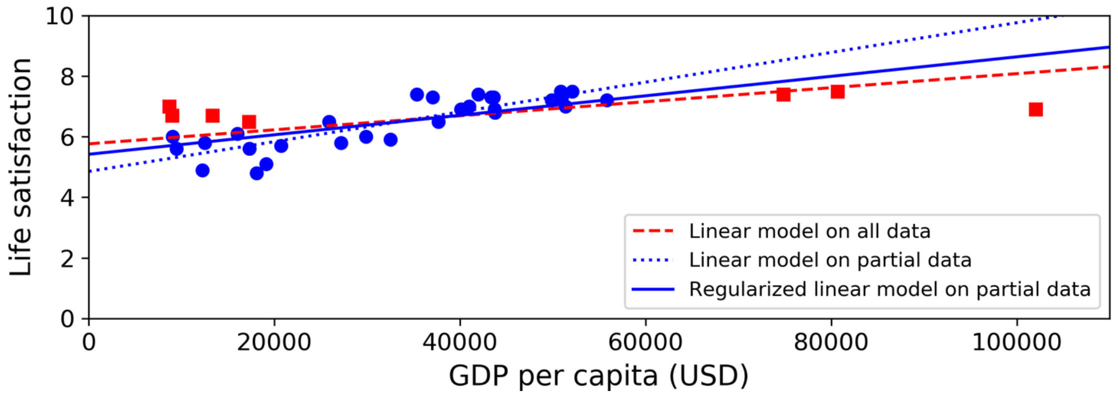
- Options to address problem
  - Simplify model by selecting one with fewer parameters, reducing the number of features, or constraining model
  - Gather more training data
  - Reduce noise in training data (e.g., fix data errors and remove outliers)



High degree polynomial with overfitting

REGULARLIZATION

- Constraining a model to make it simpler and reduce the risk of overfitting
  - E.g. constrain parameters to limit search space
- Hyperparameter – parameter of a learning algorithm (not model) to control regularlization
  - Constant set prior to training
  - Not affected by the learning parameter itself



- Will have to tune (train) hyperparameters for best performance

UNDERFITTING THE TRAINING DATA

- Occurs when your model is too simple to learn the underlying structure of the data
  - Data is more complex that your selected model
  - Predictions will be poor, even on training data
- Options to address the problem
  - Select a more powerful model, with more parameters
  - Use better features (feature engineering)
  - Reduce the constraints on the model (e.g., reduce regularization hyperparameter)

BIG PICTURE

- ML – making machines get better at a task by learning from data rather than explicitly coding rules
- ML comes in many flavors: un/supervised, batch/online, instance/model-based
- ML steps
  - Select modeling approach
  - Feed data to learning algorithm
  - Tune parameters to fit model to training data
- ML systems do not perform well if:
  - Training data is too small
  - Data is too noisy or polluted with irrelevant features
  - Model is not too simple or too complex

TESTING AND VALIDATION

- Most important goal for ML is to generalize well
  - Model should behave as expected to new unseen cases
  - Evaluate and fine-tune models to be sure it works well
- Split training into training and test sets
  - Training data – used to train model
  - Test data – test model on unseen data and measure the error rate (generalization or out-of-sample error) to estimate how well model performs
- Low training and test error is desired
  - Low training error but high test error means the model is overfitting

HYPERPARAMETER TUNING AND MODEL SELECTION

- Must select a model with various # parameters (e.g. linear and polynomial) and add regularization to avoid overfitting
- Can use test set for model generalization but not for regularization parameter tuning (test set tuning)

- Use a validation (val or development or dev set) for holdout validation
  - Subset of training data used specifically for model and hyper parameter tuning
  - Train full model on train+val and get generalization error on test set

- Cross-validation (multiple train/val data splits) can be used for better characterization with smaller datasets by averaging performance across splits
  - Val too small → imprecise model evaluations
  - Val too large → not enough training data

DATA MISMATCH

- Data must be representative
  - Don't want to train on magazine/professional (web) images if the use case are coming from user cell phones
- Makes sure val/test sets match use case
- Train-dev set is split of training data used to determine if model is overfitting or if there is data mismatch
  - Poor val performance → data mismatch
  - Poor train-dev performance → overfit and need to simplify model, add regularization, get more data, or clean data

NO FREE LUNCH THEOREM

- If you make absolutely now assumptions about the data, then there is no reason to prefer one model over any other – David Wolpert 1996
- A priori, there is no model guaranteed to work better
  - Cannot test all possible models
  - Must make reasonable assumptions about the data and evaluate only a few reasonable models
    - Simple tasks – linear models with regularization
    - Complex tasks – neural networks