Fundamentals of AI and Machine Learning for Healthcare

Study Guide

CONTENTS

Module 1 - Why machine learning in healthcare? 3
Learning Objectives 3
History of machine learning in healthcare 3
The magic of machine learning and the different approaches to solving problems 6
Citations and Additional Readings 9
Module 2 - Concepts and Principles of machine learning in healthcare part 1 10
Learning Objectives 10
Machine Learning Terms, Definitions, and Jargon 10
How Machines Learn 14
Supervised Machine Learning 18
Traditional machine learning 18
Citations and Additional Readings 20
Module 3 - Concepts and Principles of machine learning in healthcare Part 2 21
Learning Objectives 21
Deep learning and neural networks 21
Important concepts in deep learning 25
Types of neural networks and applications 29
Overview of common neural networks 33
Citations and Additional Readings 36
Module 4 - Evaluation and Metrics for machine learning in healthcare 36
Learning Objectives 36
Critical evaluation of models and strategies for healthcare applications 37
Important metrics for clinical machine learning 45
Citations and Additional Readings 50
Module 5 - Strategies and Challenges in Machine Learning in Healthcare 51
Learning Objectives 51
Challenges and strategies for clinical machine learning 51

Copyright © Stanford University

Interpretability and performance of machine learning models in healthcare 55
Medical data for machine learning 56
Module 6 - Best practices, teams, and launching your machine learning journey..... 62
Learning Objectives..... 62
Designing and Evaluating clinical machine learning applications 63
Human factors in clinical machine learning - from job displacement to automation bias 70
Citations and Additional Readings 73
Things to remember..... 73

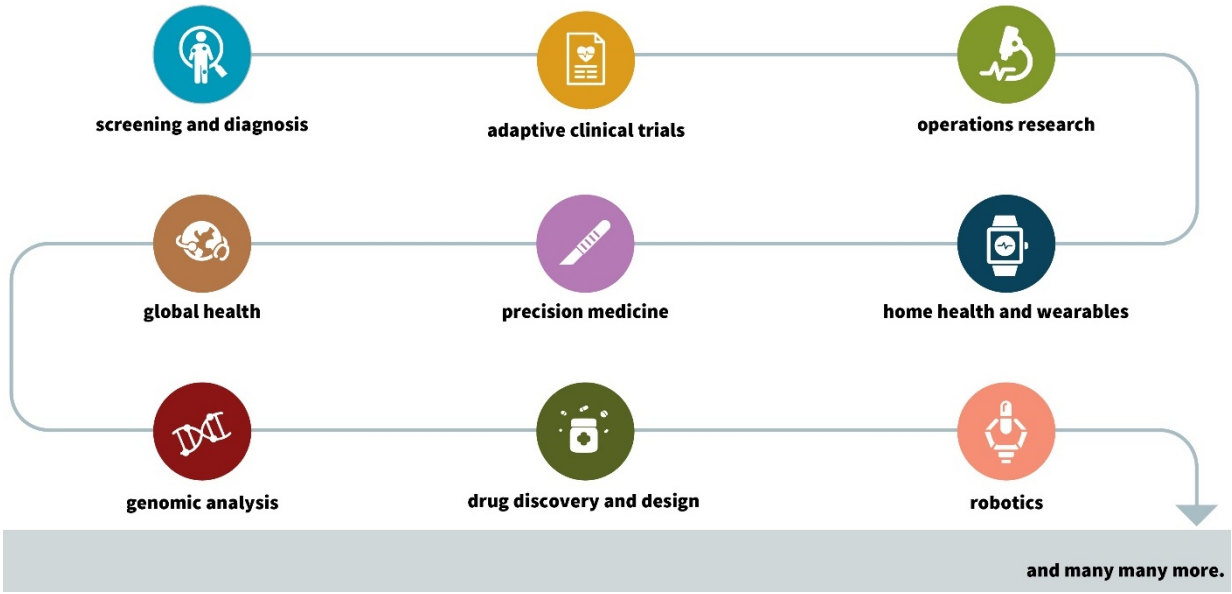
Copyright © Stanford University

MODULE 1 - WHY MACHINE LEARNING IN HEALTHCARE?

LEARNING OBJECTIVES

- Recognize the importance of learning the fundamentals of clinical machine learning for and/all stakeholders in the healthcare ecosystem
- Overview of the origins of machine learning in healthcare
- Understand context and principles of common terms and definitions in machine learning
- Define important relationships between the fields of machine learning, biostatistics, and traditional computer programming
- Begin to recognize limitations to machine learning approach in healthcare
- Introduction to first principles for designing machine learning applications for healthcare

HISTORY OF MACHINE LEARNING IN HEALTHCARE



Examples of areas in which AI could have a large impact: Automated screening and diagnosis, adaptive clinical trials, operations research, global health, precision medicine, home health and wearables, genomic analysis, drug discovery and design, robotics.

Copyright © Stanford University

Groups that will require a basic competency in both healthcare and machine learning concepts and principles: AI developers, tech companies, policy-makers and regulators, health care system leadership, pharmaceutical and device industry, frontline clinicians, ethicists, patients, patient caregivers.

In the late 1970s, Stanford became one of the first institutions to launch a program focused on applications of artificial intelligence research to biological and medical problems. It was called **SUMEX-AIM** (Stanford University Medical EXperimental computer for Artificial Intelligence in Medicine).

Projects that came out of the SUMEX-AIM project:

- AI applications to solve difficult diagnostic problems for infectious disease diagnosis
- Cancer drugs
- Diagnosis of diabetic retinopathy images
- AI Handbook Project

Progress plateaued because the ingredients required for high-performance AI algorithms did not yet exist.

The two recent ingredients that have led to newfound success in AI in medicine:

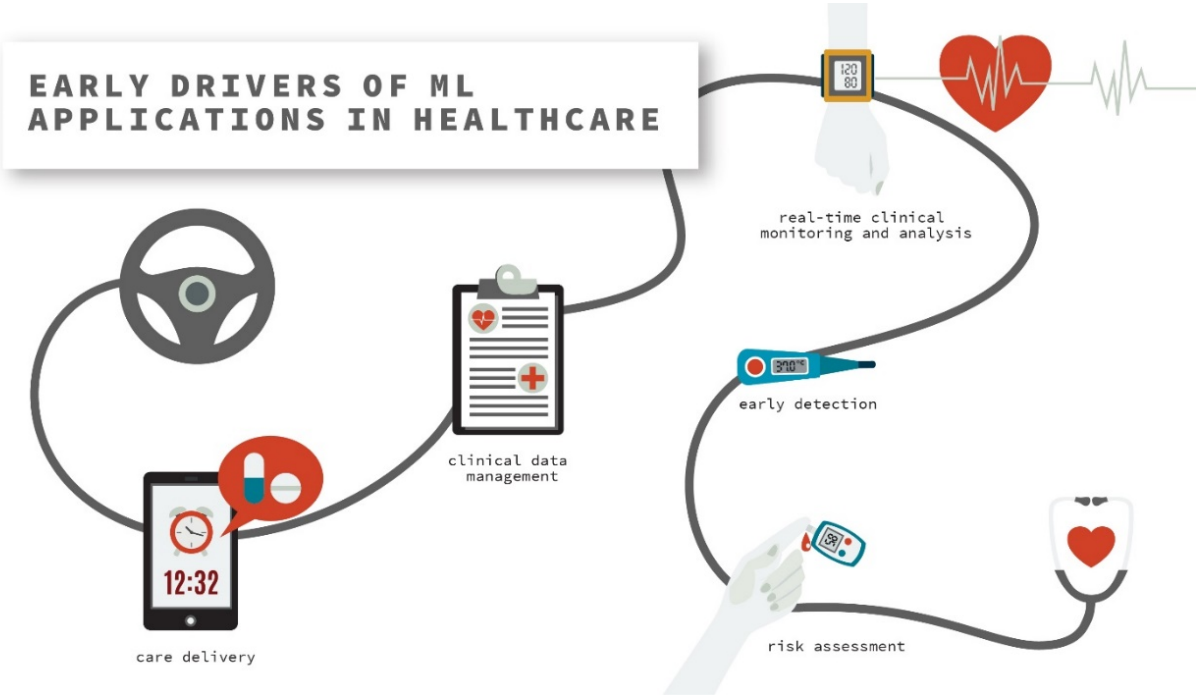
- The availability of and access to large volumes of digital healthcare data
- The development of graphical processing units (GPUs), which enable massive parallel computation

Copyright © Stanford University

We interact with AI algorithms daily in email spam-filters, retail and e-commerce, government, finance, transportation, manufacturing, autonomous driving.

Unresolved concerns about AI in healthcare:

- Workforce displacement
- Skill atrophy
- Algorithmic and user bias
- Patient privacy
- Medical-legal responsibility
- Oversight and regulation

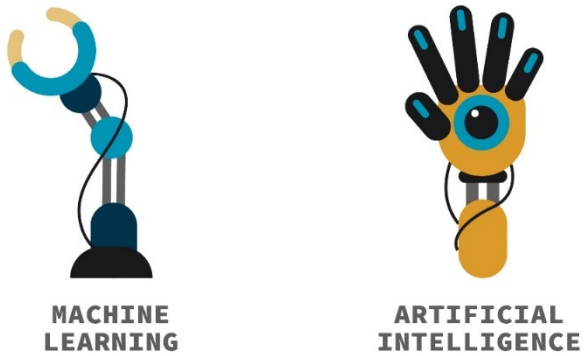


Early use cases of machine learning in healthcare:

- Enhancing and optimizing care delivery
- Improving information management
- Cognitive support for clinical care and prediction
- Early detection
- Risk assessment for individuals

Copyright © Stanford University

THE MAGIC OF MACHINE LEARNING AND THE DIFFERENT APPROACHES TO SOLVING PROBLEMS



The terms "AI" and "machine learning" are often used interchangeably.

- The term "**machine learning**" is often used by scientists or data-science practitioners
- The term "**AI**" is often used for marketing purposes or for communicating to the public

In most cases, "**machine learning**" is likely more appropriate.

The modern terms of machine learning and artificial intelligence were coined in the 1950s and '60s: The theory that machines could be made to simulate learning or any other feature of intelligence.

Artificial intelligence refers broadly to the development of machine capabilities.

Machine learning: A family of statistical and mathematical modeling techniques that uses a variety of approaches to automatically learn and improve the prediction of a target state, without explicit programming

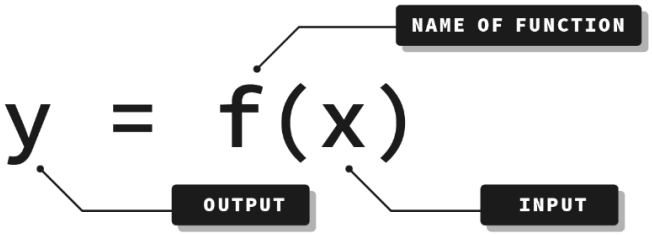
	Statistics	Machine Learning
Background	Statistics and data science	Computer science and engineering
Approach	Hypothesis-drive model development	Creating system that learn from data
Goal	Inferences; Relationships between variables	Optimization; Prediction accuracy
Assumptions	Some knowledge about population usually required	None

Copyright © Stanford University

Data complexity	Usually applied to low-dimensional data	Usually applied high dimensional data; ML learns from data
Definition of Success	Discoveries that can be applied for new predictions	Resulting model produces accurate predictions without predefined characteristics

Computer Programming vs. Machine Learning

- All computer interactions consist of an input, a function, and an output



- Computer programming
  - The computer programmer knows what the input and output looks like
  - The computer programmer writes a function that processes an input and produces an output
  - The programming and potential decisions are part of a manual effort to deliberately encode the steps or knowledge needed to provide automated output
  - Often called "rules-based systems"
- Machine learning
  - The function that maps inputs to outputs can sometimes too complex to code manually
  - In machine learning, the computer learns the function that maps inputs to outputs
  - Instead of relying on a computer programmer to come up with the rules of the function, we instead leverage existing input-output pairs to enable function learning. This is called "training" the statistical model

Copyright © Stanford University

Project Success:

- With biostatistics, success is chiefly learning new insights about the problem based on statistical assumptions, and building models based on those insights
- With Machine Learning, success is chiefly defined as creating the most accurate and reproducible model for the given task

Methodology in Machine Learning vs. Traditional computer programming:

- Machine Learning: Function learning based on the data
- Traditional computer programming: Function writing using manually coded rules

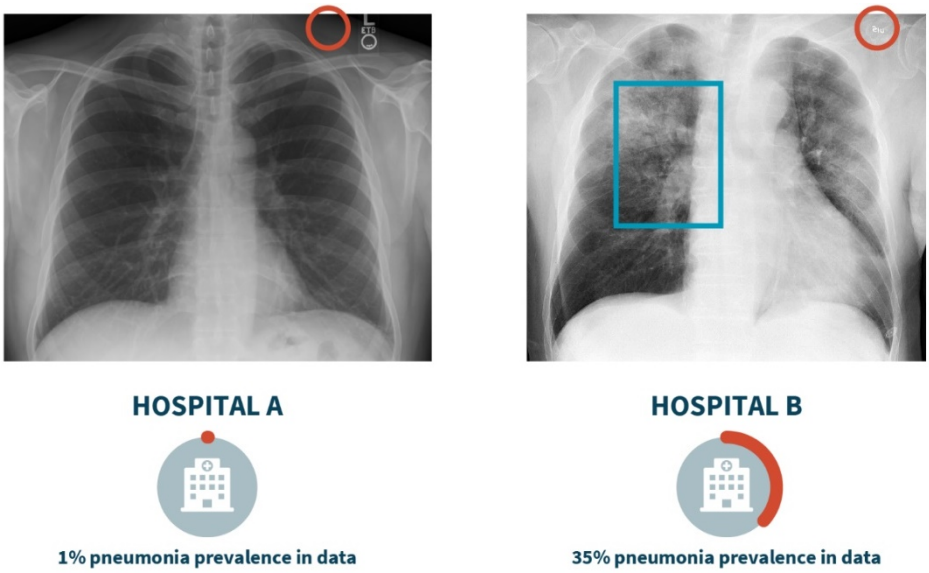
Machine learning relies heavily on pattern recognition and the theory that computers can learn useful relationships in data towards an output without being explicitly programmed.

"Learning" in machine learning is the reference of the desire to create a model that can learn like a human, through experience, and achieve an objective with little to no external (human) assistance.

Machine learning is often anthropomorphized– however, the algorithms behind the scenes use mathematical formulations to represent models and strive to learn parameters in these formulations, by tracking them back from a dataset of observations.

Machine learning has its weaknesses:

- Bias
    - Tank Example: Learning the background and the weather, not the object



    - A machine learning model trained to identify pneumonia in Chest X-rays may rely on artifacts in the images in order to determine which hospital the X-rays came from. If one hospital sees much higher prevalence than another, the model may "cheat" and predict pneumonia cases with reasonable accuracy, despite not learning anything about pneumonia at all
- Data Formatting
    - Hospitals have been collecting data for years; however, it may not be usable
    - Medical data is often generated in a discontinuous timeline
    - Medical shelf-life

It is critically important to begin with an informed question. This may be from the medical literature or pressing clinical question, but you have to start with a question that include the detailed analysis of the output of your future model and the available actions. Especially in medicine, machine learning is best understood as a means to an end that has consequences.

CITATIONS AND ADDITIONAL READINGS

Liu, Y., Chen, P. H. C., Krause, J., & Peng, L. (2019). How to read articles that use machine learning: users' guides to the medical literature. *Jama*, *322*(18), 1806-1816. https://jamanetwork.com/journals/jama/fullarticle/2754798

Matheny, M. E., Whicher, D., & Israni, S. T. (2020). Artificial Intelligence in health Care: A report from the National Academy of Medicine. *Jama*, *323*(6), 509-510. https://jamanetwork.com/journals/jama/fullarticle/2757958

Office, U. (2020, January 21). Artificial Intelligence in Health Care: Benefits and Challenges of Machine Learning in Drug Development [Reissued with revisions on Jan. 31, 2020.]. https://www.gao.gov/products/GAO-20-215SP

Schwartz, W. B. (1970). Medicine and the computer: the promise and problems of change. In *Use and Impact of Computers in Clinical Medicine* (pp. 321-335). Springer, New York, NY. https://www.nejm.org/doi/full/10.1056/NEJM197012032832305

Copyright © Stanford University

MODULE 2 - CONCEPTS AND PRINCIPLES OF MACHINE LEARNING IN HEALTHCARE PART 1

LEARNING OBJECTIVES

- Distinguish the machine learning subfield from other areas of artificial intelligence and computer science.
- Describe the model fitting procedure in the supervised learning setting and distinguish supervised learning from unsupervised learning in healthcare applications.
- Understand the difference between structured and unstructured data, as well as some of the commonly used methods to represent unstructured data.
- Become familiar with common machine learning approaches like regression, support vector machines, and decision trees and how they might apply to clinical problems.

MACHINE LEARNING TERMS, DEFINITIONS, AND JARGON

Definitions of Machine Learning

- Formal: A family of statistical and mathematical modeling techniques that uses a variety of approaches to automatically learn and improve the prediction of a target objective, without explicit programming
- Informal: Systems that improve their performance in a given task, through exposure to experience, or data

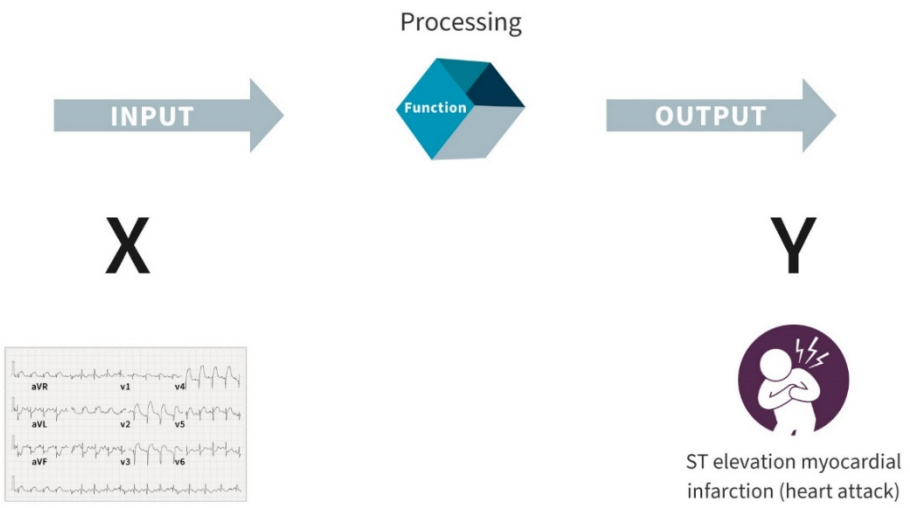
Three machine learning paradigms

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Machine learning problems fall along a spectrum of supervision between these terms.

Explaining computer programming:

Copyright © Stanford University



- Boils down to three components: (1) the input, (2) some processing, and (3) The output
- Example:
    - Equation y = x^2. The input is x and an output y
    - Abnormality detection. The input could be an ECG and the output could be a medical diagnosis like ST elevation myocardial infarction

For both these examples, in between the input and the output, there is something that **processes the input to produce the output.**

- In example 1, it is squaring of the input x to arrive at the output y
- In example 2, there is a visual analysis being performed on the ECG leading to the output

"Processing", we are referring to is the thing that transforms the input into the output, is called a **function**.

In a traditional computer programming approach, we deliberately write rules to process the inputs so that they produce the desired outputs. Traditional computer programming is also referred to as a "rules-based" approach.

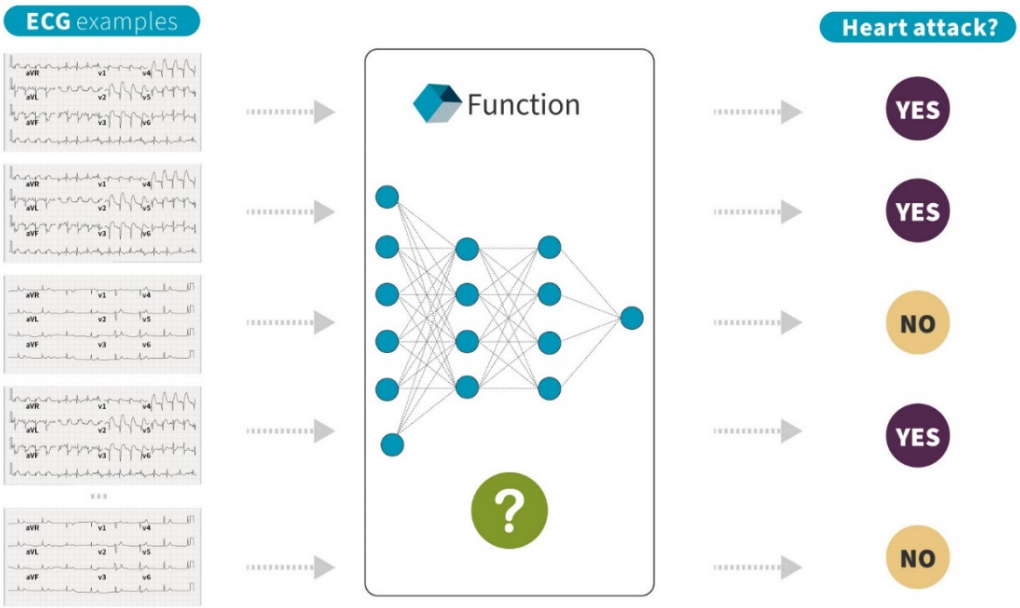
In other words, computer programmers write functions with specific rules - the program written is the function, or the processing, that the computer performs to achieve an output.

Explaining Machine Learning, in particular **supervised learning**:

- The program written in this type of approach searches for (or in other terms, learns or finds) a function that can accurately map our data of inputs to outputs. We then use this function to process new inputs, and produce new outputs

Copyright © Stanford University

If a cardiologist has already looked at the ECG (the input) and recorded a diagnosis, (the output) then we likely have two parts to the equation. All we need to do is figure out the function that solves the input-output.



Supervised learning is the process through which a program takes input-output pairs and 'learns' the function that links them together

Note that we call this 'supervised' learning because we provide input and output pairs. We are 'supervising' the model by providing it with the right answers.

- The **model**– the entity that undergoes supervised learning
  - It 'represents' or 'models' the relationships between the inputs and the outputs
  - Learning this relationship means learning a function which, in this case, means adjusting a set of numbers known as parameters
  - A model is defined entirely by its parameters and the operations between them
  - Sometimes called a model a **function approximator**– it approximates the function between the inputs and the outputs

Once the program learns a function that works well, we can use it in place of software that would have been written by traditional computer programming. We can take new inputs, put them through our learned function, and produce new outputs. This is the ultimate goal of supervised learning.

Copyright © Stanford University

In supervised learning, as in traditional computer programming, a **program still has to be written**. However, the **purpose** of the program, to search for or learn an accurate mapping function instead of pre-specifying it, is fundamentally different.

Basic Terminology:

- Example: Single input-output pairs
- Features: Input. The part of an example that is fed into the model
- Labels: Output. The part of an example that is compared with the prediction of the model
- Dataset: A collection of examples
- Prediction: The output by a model that has learned from many examples of inputs-output examples and can now take a new input and give a new output

Dataset Terminology:

- **Training set**: A set of examples that the model is given in order to learn the function that links the inputs to the outputs
- **Validation set**: A set of examples that we hold out and do not expose the model to during training, and instead use it periodically to assess, or "validate", the generalization performance of our model, as we develop the model. We also use it to make meta-level design choices about **hyperparameters,** aspects of the program that trains the model
- **Test set**: A set of examples that we hold out until the very end of the model development process, to double-check the model's generalization performance on examples that are 'completely' unseen during any aspect of model development

Training loop: A repeated training procedure that allows the model several chances of learning good, generalizable functions from the training set

Training loop structure:

1. Start the program. The program sets up the training environment with a selection of hyperparameters, and initializes the model with a random function
2. Expose the model to examples from the training set, to learn a function from inputs to outputs
3. Evaluate how the function does on the validation set. If the model gets better performance than it ever has before, we save this version of the model
4. Repeat steps 2 and 3 until the performance on the validation set no longer goes up

Typically, we repeat for various hyperparameter settings. This is known as **hyperparameter tuning**. Different hyperparameters can produce different models.

Copyright © Stanford University

Once we are satisfied with the model's performance on the validation set, we can run this final model on the **test set**.

Feature Types:

- **Structured data**: A patient's lab values, diagnosis codes, etc.– also commonly used with the more traditional statistical models. Structured data is commonly input into the model as a list (or vector) of numbers.
- **Unstructured data:** Images or natural language (text reports)
    - **Images** are typically represented as grids of numbers, where each number represents intensity at a given pixel location. In grayscale images, there is only one grid. In color images, there are three grids overlaid on top of each other; the Red, Green, and Blue grids.
    - **Texts** are typically represented with what are called **embeddings**. Word embeddings are geometric, numerical vector representations of words.

HOW MACHINES LEARN

Label types:

- Labels can be real numerical values. If a model predicts real numerical values, it is solving a **regression problem**
- Labels can be categories, or classes. In this case, labels are just numbers that act as category IDs. If a model predicts categories, it is solving a **classification problem**

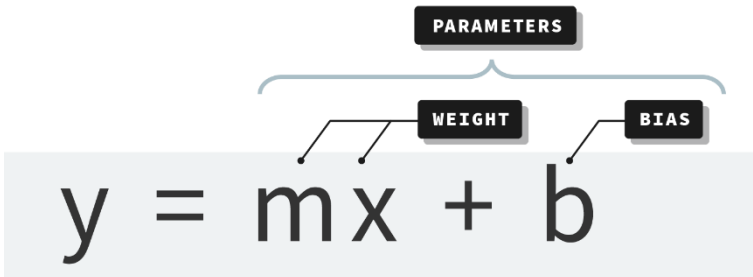
Model training

- Mathematically, training minimizes the difference between the output of the model's function and the true label, for every sample in the training set
- **Loss:** The difference between the function output and the true label. Typically, we average or sum the loss for every data point that we have
    - If the model is poorly trained, then will have high loss
    - If it is well trained, then the difference between the true label and our function will be small on average, and thus resulting loss is small as well

The model updates its function to map inputs to labels, as accurately as possible. This is known as **fitting** or **training** the model.

Copyright © Stanford University

- The model updates its function by adjusting its parameters. Parameters are numerical values that are used to perform operations on the input data.



- Example: Linear regression. The parameters are the numerical coefficients **m** and **b**, as in the equation **y=mx + b**
    - Parameters that multiply features as **weights**
    - Parameters that are added to the features as **biases**
    - It is also common practice to call all parameters– both weights and biases– "weights"
- Bias (the parameter) vs. Bias (the phenomenon)
    - Bias, the parameter: a number added to features
    - Bias, the phenomenon: a concept that relates to model performance and algorithmic fairness

Classification: predicting categorical labels

Examples:

INPUT FEATURES	LABELS
Image pixels	What is present in the image: lung nodule, skin cancer, knee arthritis, etc.
Structured data (e.g. lab values, diagnosis codes, age)	Heart attack, sepsis, mortality, etc.
Unstructured text in nursing notes or pathology reports	Final diagnosis: stroke, appendicitis, etc.

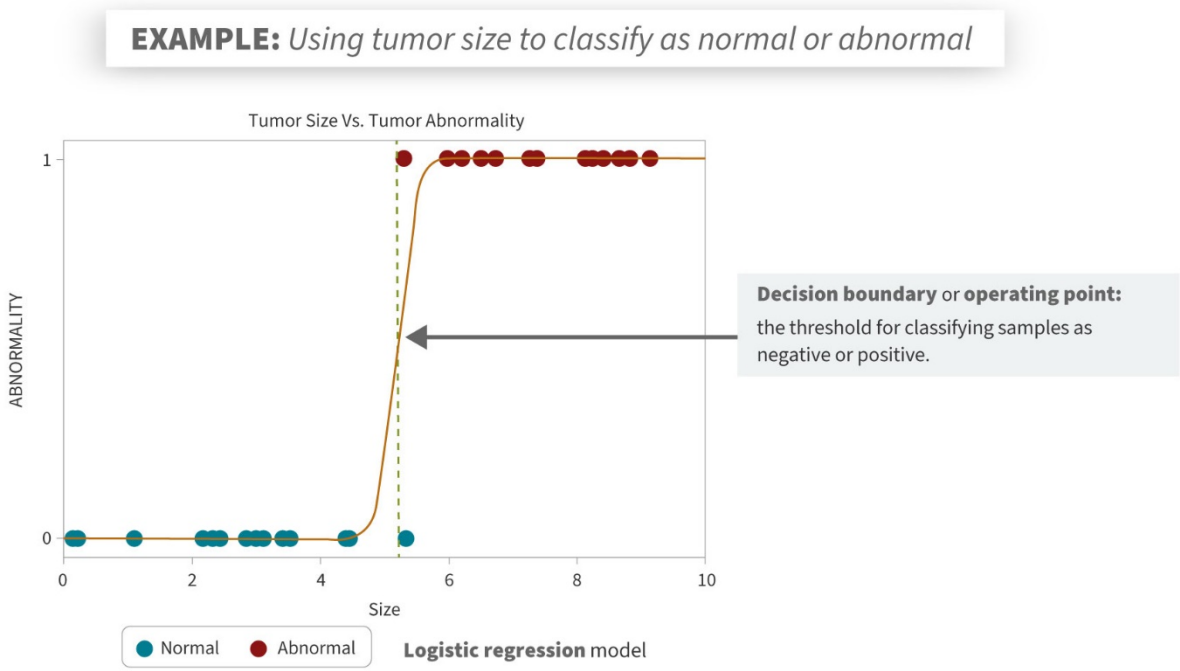
Difference between classification and regression:

Copyright © Stanford University

- Labels are categorical
- Classification models output probabilities. A model prediction in a classification task is a probability that a given set of features belongs in each category
- Probabilities are produced using the sigmoid function
- Logistic regression is a model type commonly used for classification

Decision boundary (or operating point)

- The probability number that we use as our cutoff between categories
- Commonly the decision boundary is the 50-50 mark



- Depending on the use case the operating point can be moved
    - Example: screening test in healthcare, perhaps false positives are acceptable and false negatives are not. The operating point can be adjusted so that more examples are classified as positive.

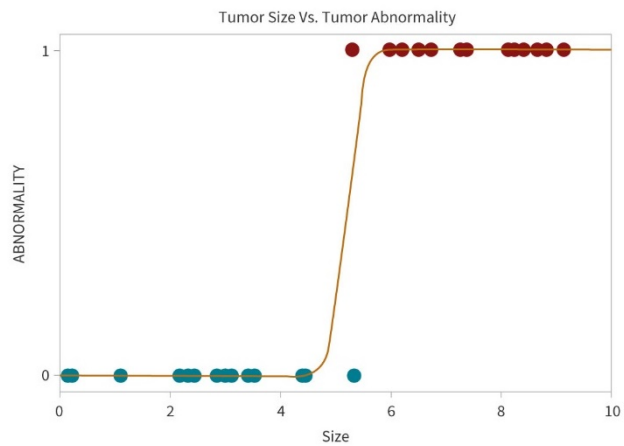
The multiple-feature setting is similar to the one-feature setting.

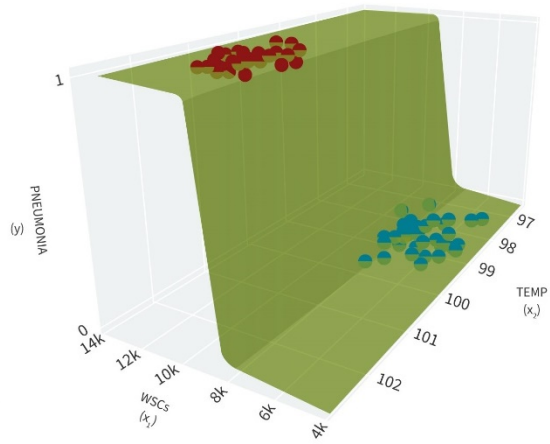
Consider the classification setting:

- In both the one-feature and two-feature cases, we can visualize the label as a separate dimension whose values separate the categories

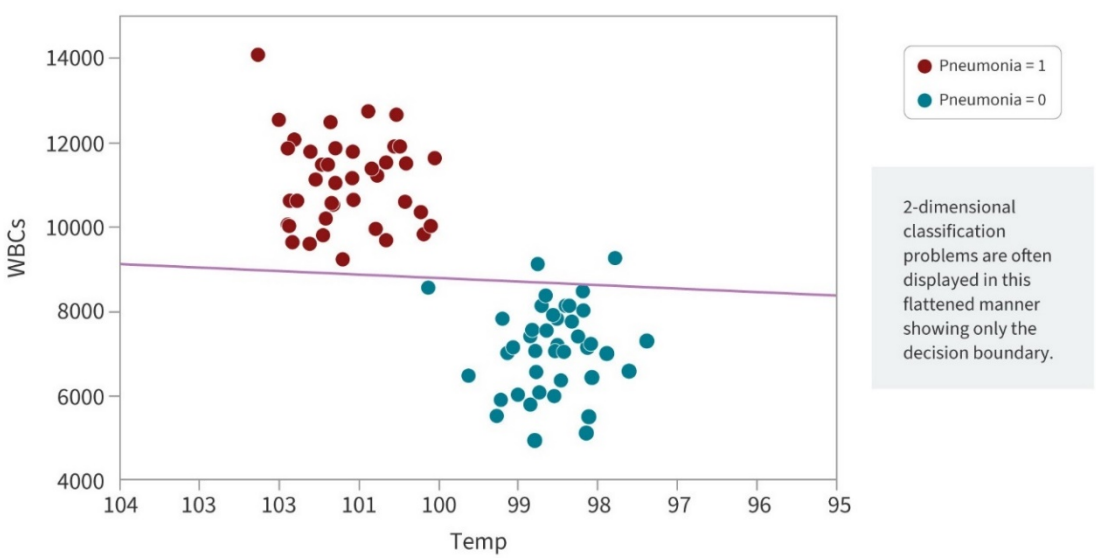
Copyright © Stanford University

- The model still has to learn a function with a decision boundary that separates data points with different labels (also called "y's")
- However, in the two-feature case, we need to adjust parameter values corresponding to both features, x1 and x2, to find a good model fit. Recall that the parameter values are the model weights, which are multiplied with the features





- Drawing the decision boundary is similar as well. Recall that a common decision boundary is where the function outputs 0.5, as in there is a 50-50 chance that the output for a sample sitting on the decision boundary is 1 vs 0.
    - Since y in this case can only take two values, we can flatten this entire figure to two dimensions and mark the y's only through color, in other words a point whose y is 1 is red, and a point whose y is 0 is blue. We can also demarcate the decision boundary by drawing a line everywhere our function equals 0.5.



The two-feature setting is a straightforward extension of the 1-feature setting. While it is more difficult to visualize, the same idea holds for **any number of features**. For binary classification, the geometric intuition is the same: **find a function whose decision boundary sits between the two sets of samples**.

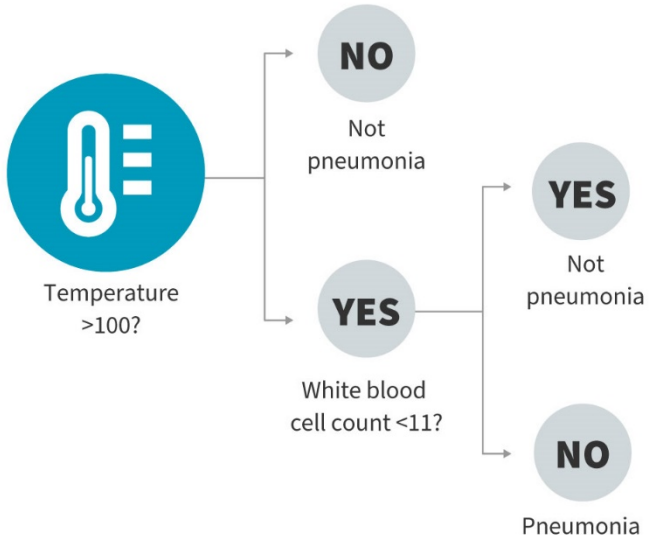
SUPERVISED MACHINE LEARNING

The "No Free Lunch" theorem: No one Machine Learning algorithm is best for all problems

- Regression variant: Polynomial Regression
  - Useful for handling non-linearly separable data where the best fit line is not a straight line
  - It fits a curve instead of a line
- Other common regression variants
  - Examples: Lasso Regression, Ridge Regression, ElasticNet Regression
  - At their core, they are all functions that can act as a classifier and can be adjusted to better fit the relationships between the features to predict the correct label.

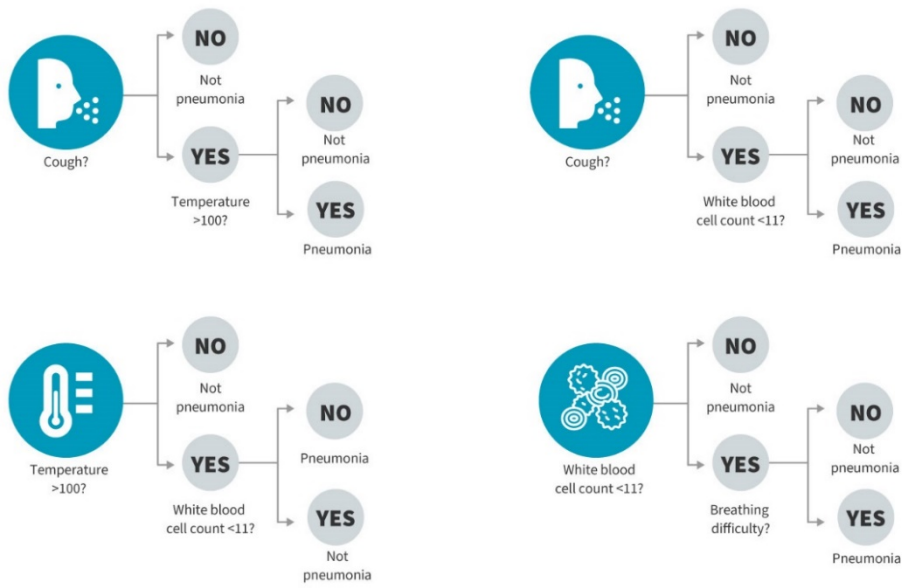
TRADITIONAL MACHINE LEARNING

Decision tree algorithms, specifically classification tree algorithms, generate a tree structure where the branching points are decision points that are based on the relationships between features found in the training dataset.



- Advantages:
  - They can be very fast to train with high dimensional datasets
  - They are simple to understand and interpret– every branch in the tree is a decision point based on some relationship between the features
- Disadvantages:
  - They can sometimes be inaccurate. This effect can be mitigated by using decision tree variants such as Random Forests

Random forests improve decision trees by ensembling, or combining, the predictions of many, many decision trees. Typically, decision trees are trained on all features using all samples in the training dataset. Each of the decision trees in a random forest algorithm only (1) sees a subset of the features made available for each sample and (2) sees a subset of the samples in the training dataset.



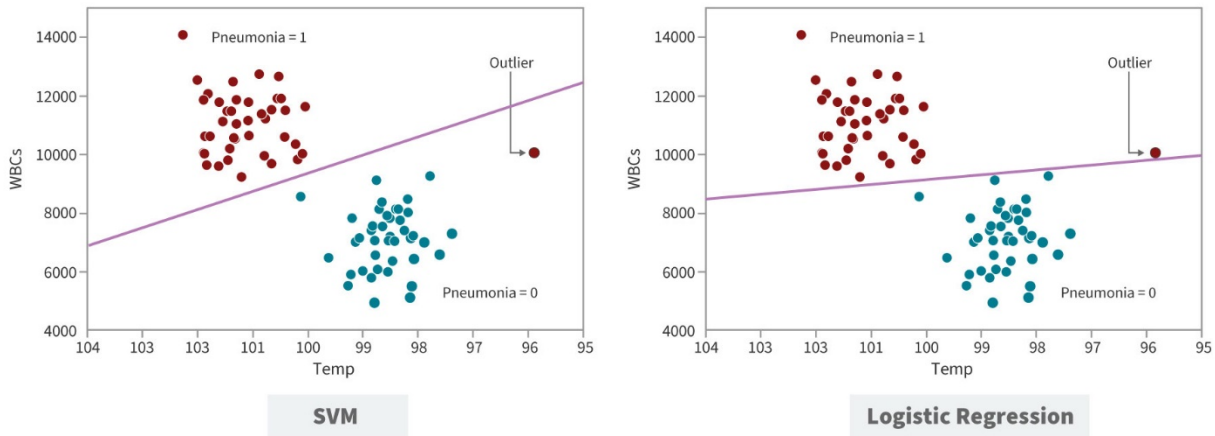
- Advantages:
  - The diversification of decision trees may produce some bad classifiers, but many other trees will be right. So, as a group, the trees are able to classify things more correctly than any single decision tree.
- Disadvantages:
  - Slower to train than decision trees.

Support Vector Machine (SVM) is another supervised learning machine learning algorithm used for classification problems similar to Logistic Regression (LR).

- Logistic Regression considers all samples equally

Copyright © Stanford University

- SVMs consider samples that are near the decision boundary more strongly than Logistic Regression, which in turn makes SVMs more robust to outliers



A large dataset with no labels at all and no feasible way to label them on your own

- **Unsupervised learning** seeks to examine a collection of unlabeled examples and group them by some notion of shared commonality.
- **Clustering** is one of the common unsupervised learning tasks. We can use clustering to define the label or category

In unsupervised learning the difficulty lies not in obtaining the grouping, but in evaluating it or determining whether the grouping that is found is actually meaningful.

The challenge, then, is whether the presence of the groups (i.e., clusters) or learning that a new patient is deemed a member of a certain group is actionable in the form of offering different treatment options or making a clinical decision.

CITATIONS AND ADDITIONAL READINGS

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning: data mining, inference, and prediction.* New York: Springer.

Matheny, M., Israni, S. T., Ahmed, M., & Whicher, D. (2020). Artificial intelligence in health care: The hope, the hype, the promise, the peril. *Natl Acad Med*, 94-97. https://nam.edu/artificial-intelligence-special-publication/

Copyright © Stanford University

Shah, S. J., Katz, D. H., Selvaraj, S., Burke, M. A., Yancy, C. W., Gheorghiade, M., Bonow, R. O., Huang, C. C., & Deo, R. C. (2015). Phenomapping for novel classification of heart failure with preserved ejection fraction. *Circulation*, *131*(3), 269–279. https://www.ncbi.nlm.nih.gov/pubmed/25398313

Shah, S. J., Katz, D. H., Selvaraj, S., Burke, M. A., Yancy, C. W., Gheorghiade, M., Bonow, R. O., Huang, C. & Deo, R. C. (2015). Phenomapping for novel classification of heart failure with preserved ejection fraction. *Circulation*, *131*(3), 269-279. https://www.ahajournals.org/doi/10.1161/circulationaha.114.010637

MODULE 3 - CONCEPTS AND PRINCIPLES OF MACHINE LEARNING IN HEALTHCARE PART 2

LEARNING OBJECTIVES

- Attain basic grasp of the mechanics through which neural networks are trained (i.e. understand the roles of Loss, Gradient Descent, and Backpropagation during the model fitting procedure) and which clinical use cases are best suited for these modeling approaches.
- Learn about common loss functions for network training and the relative differences and advantages
- Attain a comprehensive understanding of the concepts and mechanisms of DNN, CNN, and RNN architectures and begin understanding applications in medicine
- Learn about some of the most common original and important neural networks like AlexNet, VGG, GoogLeNet, ResNet
- Recognize the opportunities and challenges with reinforcement learning in healthcare applications
- Learn about advanced neural network architectures for tasks ranging from text classification to object detection and segmentation

DEEP LEARNING AND NEURAL NETWORKS

Why "deep" learning?

- In most traditional machine learning methods (SVMs, linear regression), the number of parameters is limited to the number of input features

Copyright © Stanford University

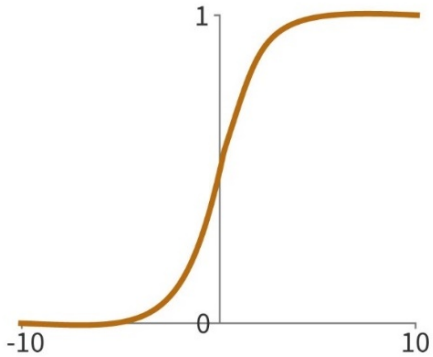
- Deep learning models are machine learning models that organize parameters into hierarchical layers
- Features are multiplied and added together repeatedly, with the outputs from one layer of parameters being fed into the next layer – before a prediction can be made
- This increased interaction between features and model parameters increases the complexity of the functions learnable by the model

Parameters and Linear Combinations

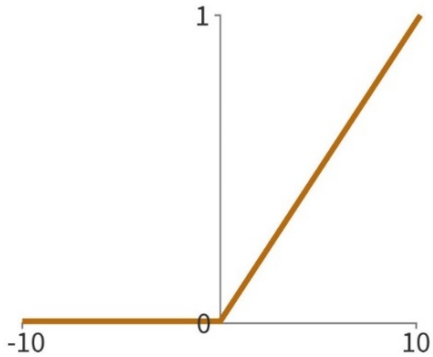
- Recall: Parameters are the set of numbers within a model that are used to define the model function -- you can think of it as the coefficients of the function -- that are adjusted or "trained" during the machine learning process to accurately transform inputs into desired outputs. They are just numbers that are multiplied and added in various ways
- The combination of parameters with a set of inputs via multiplication and addition is known as a **linear combination**, and this comes from the fact that in cases where we have one feature and one parameter, the resulting function is a line, and this can be naturally extended to higher dimension

Activations

- Recall: We use logistic regression when the output label is categorical. We used the sigmoid function to transform the function from a line to an "S" shape in order to fit our categorically labeled data
- The sigmoid function, as we had mentioned earlier, is known as a nonlinear transformation– in comes a line, out comes something else. In deep learning terminology, we often use the term activation functions to refer to the nonlinear transformations we use, and we call the result of a linear combination followed by an activation function as the activations associated with the parameters involved



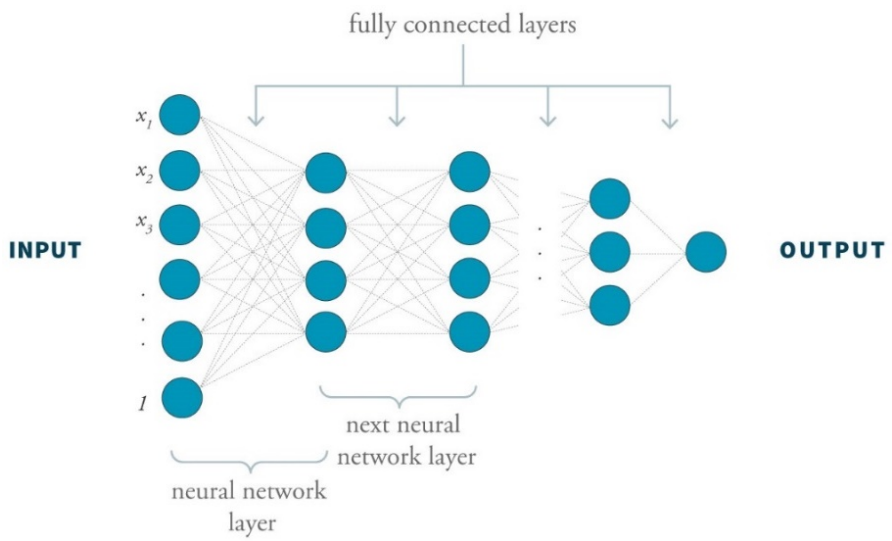
Sigmoid Function
(an activation function)



ReLU Function
(another activation function)

- Examples of activation functions
    - Sigmoid
    - ReLU (more popular): Many recent models use what is the ReLU activation function, which stands for Rectified Linear Unit. This function passes input values as-is if they're positive, but turns all negative inputs into zero. It's called a rectified linear unit because it rectifies, or passes through, only the positive side of the input range, and what it does pass through is linear. And the rectification makes the function not a line, such that it is a nonlinear transformation

Neural Network Building Block: Dense Layers



- The neurons of a neural network are, for all intents and purposes, miniature logistic regression models
- A **layer** of a neural network consists of a set of neurons that each take the same input
- **Fully connected layer** (also called a **dense layer** or a **linear layer**) is a set of neurons that take the same input. These neurons execute a linear combination of the inputs and the layer's parameters produce an output. These outputs can then be fed into another layer.
- The architecture of a neural network is how these layers are organized

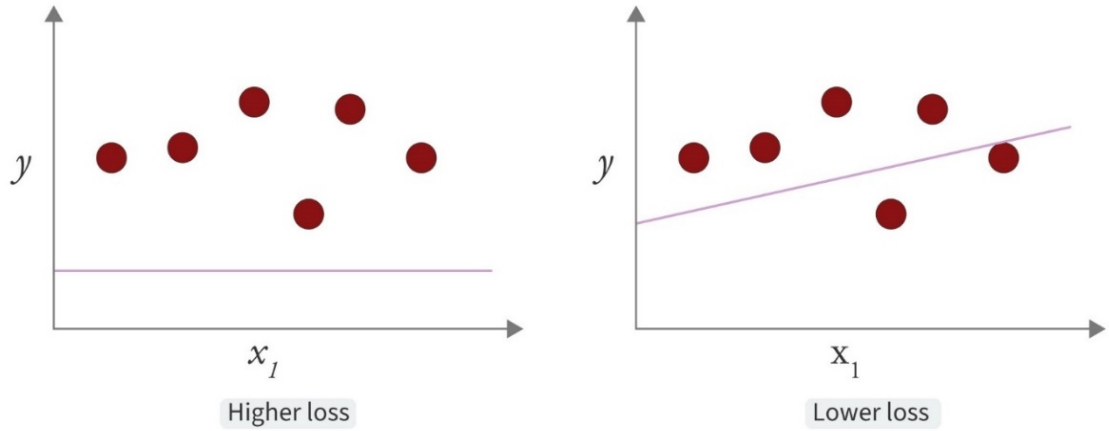
Reviewing the training loop:

- We split our data into train, validation, and test datasets
- We take a pass through our training dataset:
    - Our model will make a prediction based on the sample's features

Copyright © Stanford University

    - We will compute the loss between the model's prediction and the sample's label. The loss is a numerical value representing how far the prediction is from the label. Low loss is good, and high loss is bad
    - The model will then update its parameters in a way that will reduce the loss it produces the next time it sees that same sample. This is known as an **optimization step**
- Periodically, for example after we have taken a pass through our training dataset, we can evaluate our model on a validation set
    - In this phase, we assess if the parameters the model has learned produce accurate predictions on data that it has not yet observed, in other words the validation set.
    - The model does not learn from these samples because we do not execute the optimization step during this phase. Without the optimization step, the model cannot update its parameters, which in turn prevents learning
    - The validation set is a measure of how the model will do "in the real world." We save a version of the model if it gives us the best validation performance we have seen so far
- The process is repeated multiple times, each time with different training configurations. This is known as **hyperparameter** tuning.
- This module focuses on the optimization step, which is comprised of three components:
    1. Loss
    2. Gradient Descent
    3. Backpropagation

Loss: Informs the way in which all of the different supervised machine learning algorithms determine how close their estimated labels are to the true labels



Copyright © Stanford University

- The goal of training an algorithm is to find a function/model that contains the best set of parameters that results in the lowest loss across all of the dataset examples
- There are multiple loss functions that exist, and some are better adapted for some problem types than others. Some factors to consider:
    - What kind of problem are we solving?
    - How much weight should we put on outlier labels?
    - Are the number of samples we have in each category roughly equal? (for classification)

Loss functions: A mathematical function that processes all of the individual losses to come up with a way to decide how well a model performs

- Mean Squared Error (MSE)
    - The simplest and most common loss function
    - To calculate the MSE, you take the difference between the model predictions and the true label, which is also known as the ground truth, square it, and average it out across the whole dataset
    - Squaring (1) gets rid of the sign (+/-) of the difference between the prediction and the ground truth and (2) emphasizes outliers
- Mean Absolute Error (MAE)
    - To calculate the MAE, you take the difference between the model predictions and the ground truth, then take the absolute value to that difference. Finally you average the sample losses across the whole dataset
    - Since we did not square the error, all of the errors will be weighted on the same linear scale. Thus, we do not put more weight on our outliers

IMPORTANT CONCEPTS IN DEEP LEARNING

Cross Entropy Loss: The most common loss function in classification settings. This function determines the loss by the difference in the probabilities of the predictions.

The function is simple– you sum the negative log of the model's predicted probability for the ground truth class. Because probabilities are between 0 and 1, the log value is some negative number.

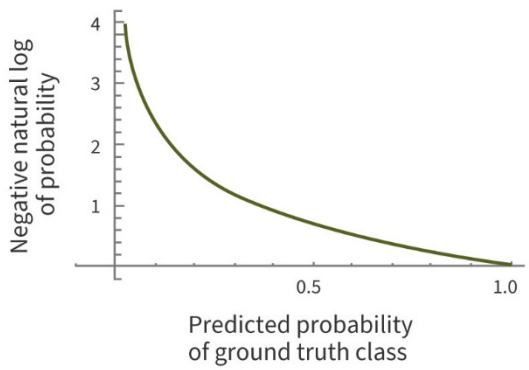
Copyright © Stanford University

Cross Entropy Loss:

In practice often use ln (natural log)

01 Loss for one example = - ln (probability of ground truth class)

02 Loss over dataset = average of loss over all examples

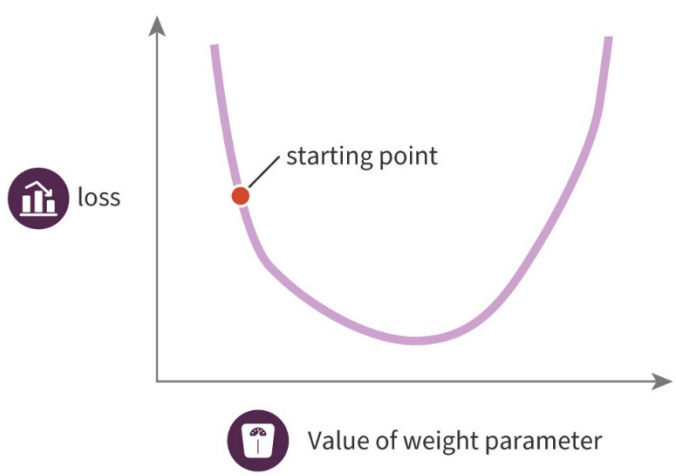


Sanity check:

- The log of a value that is close to 0 is a large negative number. Because we are using the negative log, this flips to being a large *positive* number
- The negative log of a value that is close to 1 is close to 0
- These dynamics are in line with what we need from a good classification loss function. In order to achieve a low loss, a classifier will have to produce probabilities for the ground truth class that are close to 1

Gradient Descent: Optimization algorithm to find good model parameters by minimizing a loss function

- The loss computed for a given sample gives each model parameter a sense of where to go; whether or not it needs to increase or decrease its value in order to allow the model as a whole to produce a better prediction the next time



- Each parameter has a starting value – often this number is set randomly

- The job of the gradient descent algorithm is to first figure out where the current parameter value is in relationship to the parameter value that produces the optimal loss. Then, it adjusts the parameter value in the correct direction
- To do this, it will find the slope (a.k.a. the derivative) of the loss function with respect to the parameter and figure out which direction to move. This is called calculating the gradient

Backpropagation: The key technique that breaks down gradient computation into easier gradient computations that are then combined together, and is the secret sauce for allowing us to obtain gradients for large neural network models

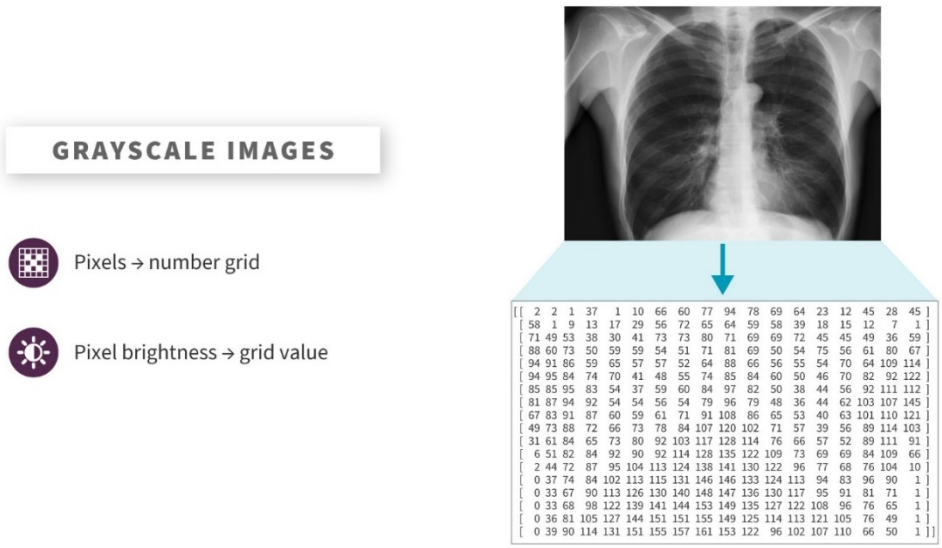
- At a high level: backpropagation allows parameters near the end of the network to send information about how to adjust to the parameters near the beginning of the network. The process goes from the end of the network to the beginning (back) and sends information from layer to layer (propagation)

Things to keep in mind about Gradient Descent:

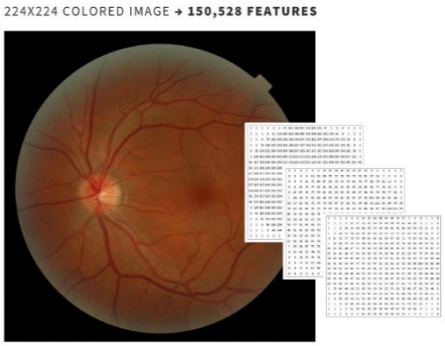
1. There are many variations of gradient descent
2. There are a number of other factors that can customize the way you move towards minimizing loss according to the gradient

REPRESENTING UNSTRUCTURED IMAGE AND TEXT DATA

Images are represented as one or many grids of numbers stacked on top of each other.

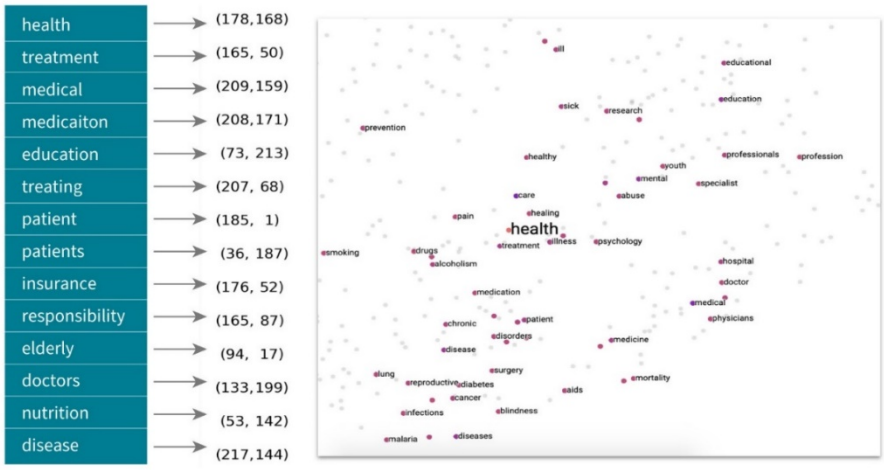


A low number is black and a high number is white. The numbers generally range from 0 to 255, where 255 is the maximum number representable by a single byte in a computer. And if we imagine overlaying numbers that represent each pixel brightness value over a black and white image what you will have as a result is a grid.



Instead of one grid, there are three grids for color images, each grid of numbers representing the brightness of Red, Green, and Blue, respectively. The magnitude of each number represents how much of a color exists in a given pixel. If a certain location in the grids has high red and blue values, it will show up to the human eye as purple.

Words are represented using **word embeddings**. A word embedding is a geometric (think "spatial") representation of a word. Here is a simple example of some 2-dimensional word embeddings.



Typically the number of dimensions are very large, and typically 300- or 1024-dimensional word embeddings are used, so that all the nuanced relationships in language can be captured encoded via coordinates in space.

Copyright © Stanford University

The number of input features becomes enormous when we move into the unstructured data scenario.

- The number of features in a color image that sized to 224 pixels x 224 pixels (common among machine learning models) would 150,528 features for each image.
- A single sentence of text in the English language consists of about 10 words, and common word embeddings are in 1024-dimensional space. That would be be 10 * 1024 = 10,240 features for each sentence.

In order to train neural network models on high-dimensional input data, we have to be creative in the way that the network architectures are constructed.

TYPES OF NEURAL NETWORKS AND APPLICATIONS

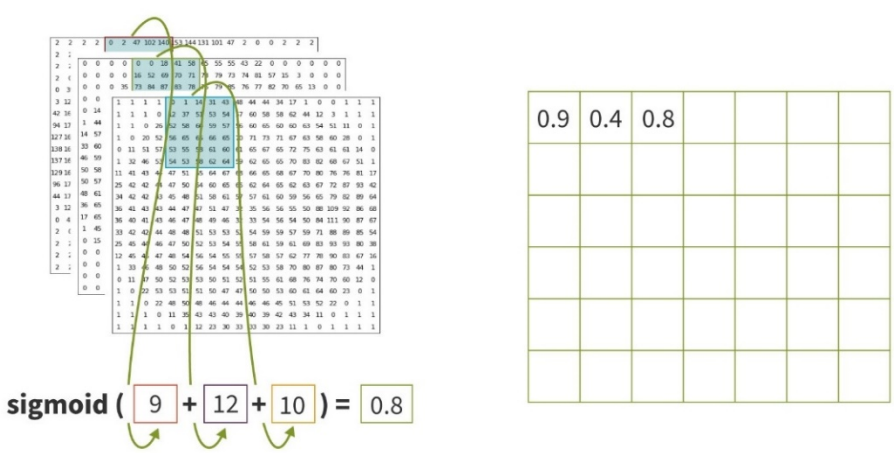
CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNNs): Designed with images in mind.

Recall: Images are pixel grids. Using dense layers, it would be extremely difficult to process these pixel grids.

- Issue 1: Each parameter would be dedicated to only one or a handful of pixel locations at a time. In nearly all practical settings, images present objects in highly variable positions.
- Issue 2: The pixels above, below, and next to a given pixel are highly related. Flattening the image into a feature vector destroys the spatial information we have about the image.

Convolutional layers solves these two issues using convolutional filters (kernels).



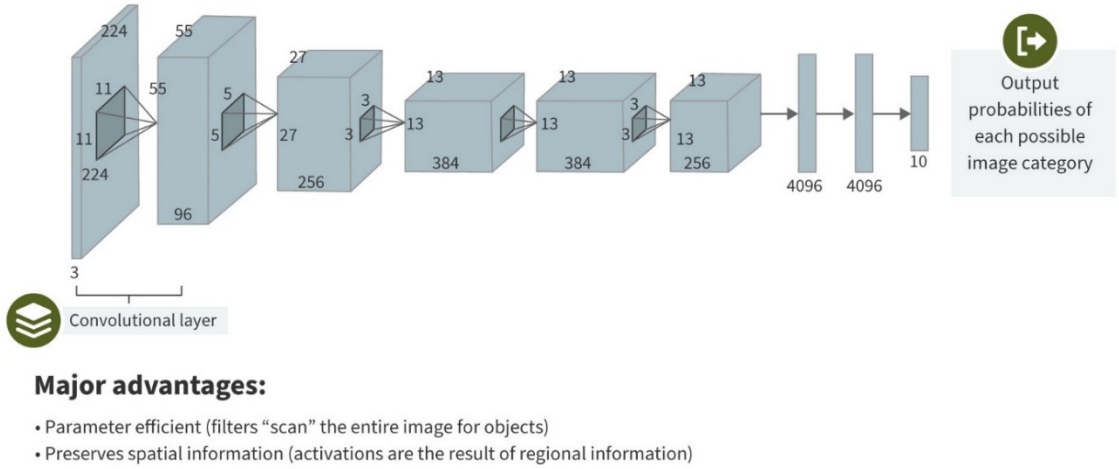
Copyright © Stanford University

Convolutional filters are small groups of parameters that are multiplied and summed (i.e. a linear combination) with 1 small patch (or window) of the image at a time. The output of each linear combination is placed relative to the location of the input patch in a new, sometimes smaller number grid.

You can think of the filter as an "feature detector." Since only one value is produced for every patch in the image, in a trained model one can imagine a high number being computed if something of interest is found in the patch and a low number if not.

One can stack convolutional layers: the grid of activations produced by one convolutional layer can act as the "image" input of the next layer.

- Since each pixel in the next layer is comprised of information from a patch of pixels in the inputs of the previous layer, the later layers of the CNN are then able to "see" larger and larger patches of the images. The amount of raw pixel information a convolutional filter can see at any given moment is called its "receptive field."



Convolutional layers are what make CNNs, CNNs. CNNs are trained just like dense neural networks, albeit computing the gradient for gradient descent becomes a slightly more complex procedure. CNNs are immensely popular in subdomains of medicine such as radiology and nuclear medicine, among others.

NATURAL LANGUAGE PROCESSING AND RECURRENT NEURAL NETWORKS

Natural Language Processing (NLP) is a machine learning method that can enable computers to understand and organize human languages. It requires a different type of neural network architecture.

Copyright © Stanford University

The difficulty lies in the neural networks ability to understand not the vocabulary, but the meaning and context behind each word.

There are two major architectures currently being used for NLP– **Recurrent Neural Networks (RNNs)** and **Transformers**.

Recurrent Neural Networks, or RNNs, are a type of model architecture typically used in scenarios where the unstructured data comes in the form of sequences. Most commonly, they are used to solve Natural Language Processing (NLP) tasks.

- NLP tasks often take a slightly different form than typical machine learning tasks due to the fact that **inputs and outputs can take the form of sequences**.
- A sentence can be thought of as a sequence of words, which we discussed would look like a sequence of word embeddings.
- Like with images, we can consider flattening this time sequence data into one vector and feed it into a dense neural network. This has a few issues:
  - Issue 1: Just like with images, each parameter of the first layer of a DNN would be assigned to a single feature of a word embedding at a single timestep. Sequences (especially in language) are far too dynamic for this.
  - Issue 2: There would be no way to vary the length of the output. The final layer of a DNN always produces an output of fixed size.

RNNs address the above two issues by doing the following:

- They process information from one timestep at a time. In language, this means processing word embedding at a time.
- They can store information about past timesteps in what is called the *context* vector. The context vector from the previous timestep is used as additional input to the current timestep feature vector in order to give the RNN information about the past.

The core component of a recurrent neural network is known as an **RNN cell**.

- RNN cells can take as input both the output from earlier layers of the neural network and an additional "context" set of values that can be used to pass information from one timestep to the next.

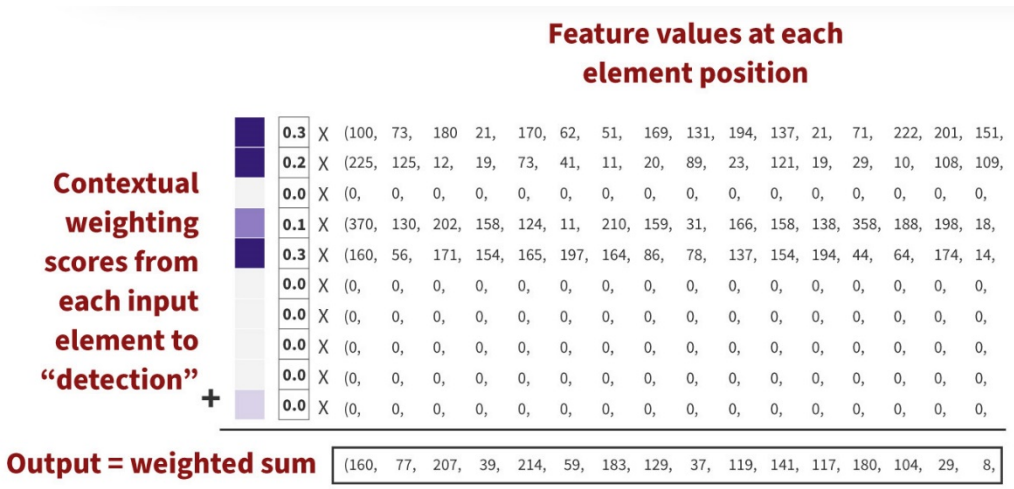
THE TRANSFORMER ARCHITECTURE FOR SEQUENCES

The Transformer architecture is quickly replacing RNNs in sequence-based problems.

Copyright © Stanford University

- RNNs start to lose effectiveness if they have to deal with long-range dependencies in a given sentence.
- Further, on the more technical side, it is hard to "parallelize" the process of an RNN. One of the most powerful features of modern computing systems is the ability to do many tasks at the same time. For a given element in a sequence, RNNs need the information from the past elements in order to output predictions, therefore their computations have to happen sequentially. For long sequences, this becomes a problem.

The Transformer architecture is built around a layer known as the **self-attention layer**, which allows for the processing of sequences all at once, while at the same time producing outputs that are aware of the rest of the sequence.



- Self-attention layers compute a contextual relatedness signal, or weight, between every element and the others in an input sequence.
- The element at each position is transformed using a weighted sum of feature values from the elements in other positions, based on the strength of the contextual relatedness. Self-attention layers can produce multiple weighted outputs to encode different types of context.
- They look at the entire input sequence and then "pay attention" to context elements variably based on what element values actually are.

Transformers directly address some of the problems associated with RNNS.

- Self-attention layers looks at the entire input sequence at once, so it avoids the forgetting problem associated with RNNs on long sequences.
- Since entire input sequences are processed at once, self-attention layers are more efficient than RNNs which have to sequentially process input elements one by one.

Copyright © Stanford University

Transformer architectures stack many self-attention layers on top of each other. So far, they have proven themselves to be both faster and better performing than RNN architectures in many settings.

OVERVIEW OF COMMON NEURAL NETWORKS

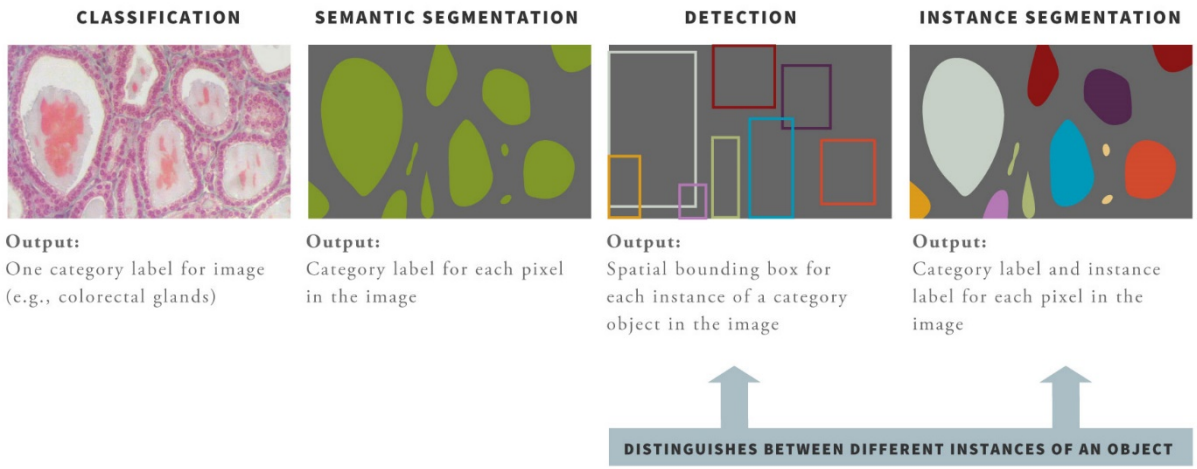
ImageNet: Benchmark dataset that encouraged the creation of numerous now widely-used CNN architectures.

- AlexNet, named for its primary inventor, Alex Krizhevsky, was introduced in 2012 and made a splash by cutting the best previous error rate on the ImageNet challenge by almost half.
    - Architecture: 8 layers -- 5 convolutional layers and 3 fully connected layers at the end. First architecture to use the ReLU activation function.
- VGG and GoogLeNet architectures were introduced in 2014. VGG was named for the Visual Geometry Group at Oxford University where it was developed, and GoogLeNet, as you probably guessed, came from Google.
    - Both the VGG and GoogLeNet architectures were still CNNs but significantly deeper - 19 and 22 layers - respectively, and came in at the top of the ImageNet challenge that year with significant further improvements.
    - The VGG architecture looks much like the AlexNet one, but found success through smaller filter or receptive field sizes at each layer, which enabled a deeper network to work well.
    - The GoogLeNet architecture, on the other hand, looks a little more different, with modules that they called Inception modules stacked on top of each other.
        - Each Inception module has the structure of several parallel convolutional pathways.
        - Called Inception because these were like mini-CNNs within a CNN.
        - Because of the Inception modules, GoogLeNet is also interchangeably referred to and perhaps more commonly known as the Inception network.
- ResNet architecture, developed by Microsoft Research in 2015. The first architecture to beat a decently accurate human benchmark on the ImageNet dataset
    - Variants of the architecture had 152 layers or even more, compared to the previous architectures of around 20 layers, and this moment was known as the "Revolution of Depth" in deep learning
    - ResNets introduced a new type of mini-module that they stacked in their CNN called residual blocks. These blocks is have skip connections which pass inputs or

Copyright © Stanford University

intermediate inputs to later portions of the network, which allowed information from the beginning of the network to reach the end of the network more directly

Semantic Segmentation and Object Detection and Instance Segmentation:



- Semantic segmentation
  - Semantic segmentation allows us to obtain pixel-level granularity of where categories are present in an image. However, it does not allow us to differentiate between distinct instances of category objects in the image, for example distinct tumors or lung nodules
- Object detection
  - In this case, the neural network outputs bounding boxes corresponding to the center and height and width of a box that tightly borders each instance such as an individual lung nodule in the image
- Instance segmentation
  - The output of an instance segmentation neural network is both the bounding box of each instance, as well as a pixel mask corresponding to the segmentation of the object within each bounding box

Each of these tasks, because they produce different types of outputs, require different neural network architectures or structures to produce each type of output.

- For segmentation, U-Nets are commonly used. The U-Net architecture contains a downsampling and upsampling pathway. The downsampling pathway extracts important information from the raw image information. The upsampling pathway generates a pixel map of the same shape as the raw image, where each pixel has a value that corresponds with the segmentation prediction

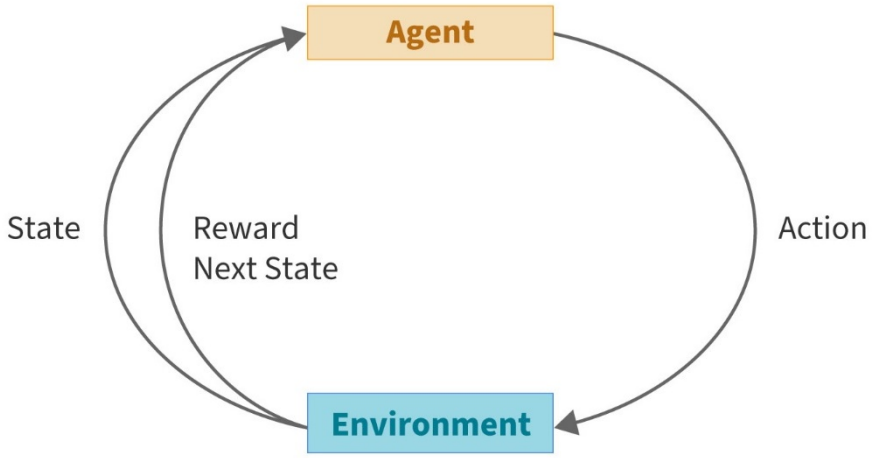
Copyright © Stanford University

- Object detecting neural networks predict spatial bounding boxes of objects and have both classification and regression branches. The classification branch predicts the category of objects as we've seen before, while the regression branch outputs numerical values corresponding to the location and extent of the object bounding box
- Finally, instance segmentation combines these two tasks. One branch is single category classification for each object or bounding box. Another is a branch of the network that produces a pixel map
    - This technique is especially attractive for imaging tasks that deal with a large number of objects that are repeated or crowded
- These tasks of semantic segmentation, detection, and instance segmentation are more detailed than classification, and so they also require more detailed labels in the training dataset

Deep Learning:

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

Reinforcement learning, currently the most challenging area of machine learning. It is one of the least explored for healthcare applications.



- Reinforcement learning is centered around the idea of the model interacting with an environment as an "agent", continuously observing the current state of the environment and making decisions based on it.

Copyright © Stanford University

- We use the word agent here because taking actions is the central concept in reinforcement learning, the AI agent here is basically our algorithm. This is often used in the setting of a gaming or simulation environment.
- Main challenges in healthcare:
    - There is not a clear methodology towards environment simulation. In games, environment simulation is easy, because one can just run the game. In healthcare, experiments are much more high stakes.
    - Not immediately clear how to reward the agent. In games, there are explicit scoring mechanisms. In healthcare, it would be dangerous to naively index patient wellness on a single metric.

CITATIONS AND ADDITIONAL READINGS

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Komorowski, M., Celi, L.A., Badawi, O. *et al.* The Artificial Intelligence Clinician learns optimal treatment strategies for sepsis in intensive care. *Nat Med* **24,** 1716–1720 (2018). https://www.nature.com/articles/s41591-018-0213-5

MODULE 4 - EVALUATION AND METRICS FOR MACHINE LEARNING IN HEALTHCARE

LEARNING OBJECTIVES

- Learn important approaches for leveraging data to train, validate, and test machine learning models
- Assess model training behavior and understand important concepts like underfitting and overfitting in healthcare settings.
- Begin developing an intuition regarding hyperparameters and the downstream effects of hyperparameter tuning.
- Determine the correct set of metrics for model evaluation and understand the most common metrics used in machine learning for clinical research especially the receiver operating curve and precision recall curve

Copyright © Stanford University

CRITICAL EVALUATION OF MODELS AND STRATEGIES FOR HEALTHCARE APPLICATIONS

It is very important to consider how we evaluate the performance of our models.

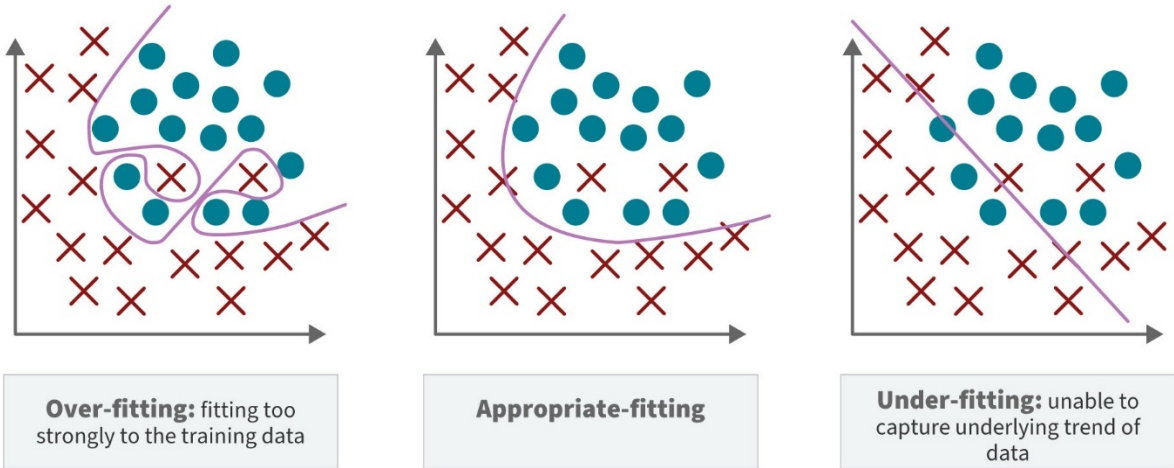
Recall that we typically split our data into 3 sets - **train, validation**, and **test**.

- Usually 70-80% of your data is used for training and the rest for testing the model performance. (i.e we use a 80%–20% or 70%–30% **train-test split**)
    - In the special case of time series data, most people hold aside all data from the most recent time point and train the model on the data before that (i.e. training and validation data from 2012-2015 and test 2016-2017)
- You may also hear "train-validation-test" split
    - Recall that the validation set is used to choose hyperparameters. It is different from the test set, which contains data you never see while tuning your model
    - Note that sometimes you may hear the validations set referred to as the dev or development set. Some research scientists reserve the term validation set to refer to the test set as the validation set

Another method for shuffling the training and testing data is something called **cross-validation,** or **"k-fold cross validation**"

- Similar to a train/test split, but instead of creating one split, multiple splits are applied to more subsets multiple times
- In a typical 80-20 train-test split, you simply assign 80% of the data at random to the train set, the rest goes to test. In k-fold cross validation, we repeat this process k times, so that a different, random, 80% of the data ends up in train each time. Each split creates a "fold." You then train on k-1 of these folds, holding out the last one to use as the test set
- Doing this over and over allows us to get many different estimations model parameters
- This method can be particularly advantageous when there is not much data to begin with

Copyright © Stanford University



Overfitting occurs when the model begins to memorize the random fluctuations, anomalies, and noise in the training dataset. At this point, the model will have a high training accuracy despite the fact that it will no longer be extracting relevant generalizable signal from the data, meaning the test accuracy will suffer.

- Extended **training time** can lead to overfitting
  - If trained for too long on the same data, a model will start to overfit, harming the performance of the system on new data
  - To try and guarantee that our model is extracting only generalizable signal from the train set, we want to stop just before the error on the validation set starts increasing
- Excessive **model complexity** can lead to overfitting
  - A very complex model (i.e. a very deep neural network) that can accommodate a high number of feature weights, will frequently experience overfitting - especially on small datasets
    - For instance, if your feature space was the same size as your data set, the model could directly memorize your data by associating each data point with a unique feature

Underfitting occurs when a statistical model is unable to obtain a good fit of the trends in the data, leading to poor performance on the training and testing data.

- This occurs when a model is too simple to fit more complex trends in the data

Appropriate fitting is the goal, and one of the major challenges facing machine learning practitioners.

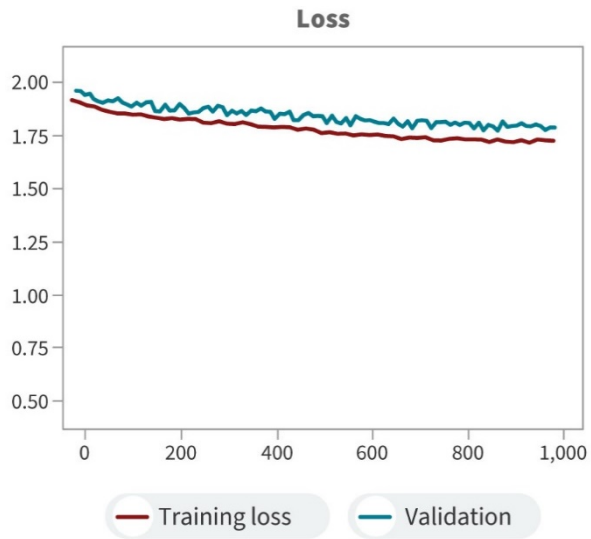
Copyright © Stanford University

- In order to hit the sweet spot of appropriate fitting, we can tweak hyperparameters and modify algorithmic design choices
    - The gains achieved in this way will grow smaller over time as the model gets closer and closer to the optimal weights

We use **learning curves** - plots of model performance over time - to monitor the learning process in algorithms that learn incrementally from training data

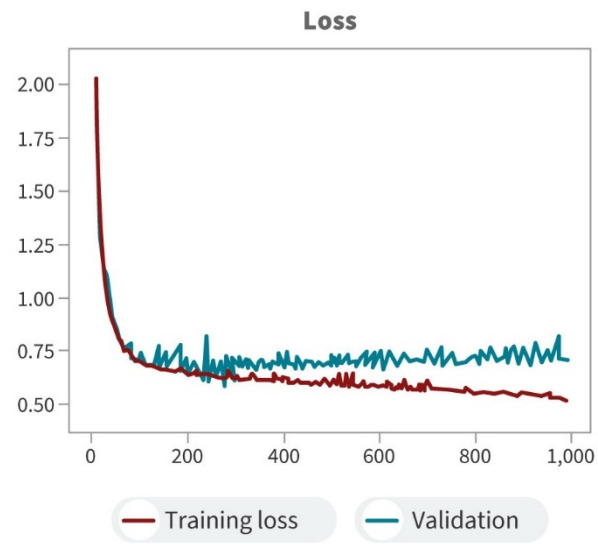
- They are used to diagnose problems such as model underfitting or overfitting, to sanity-check and to debug code and implementations
- Often learning curves plots **model loss** over time, or as the model is exposed to more and more training data

Plotting the model's learning curve for both the training and validation sets side by side can be very **useful for debugging**.



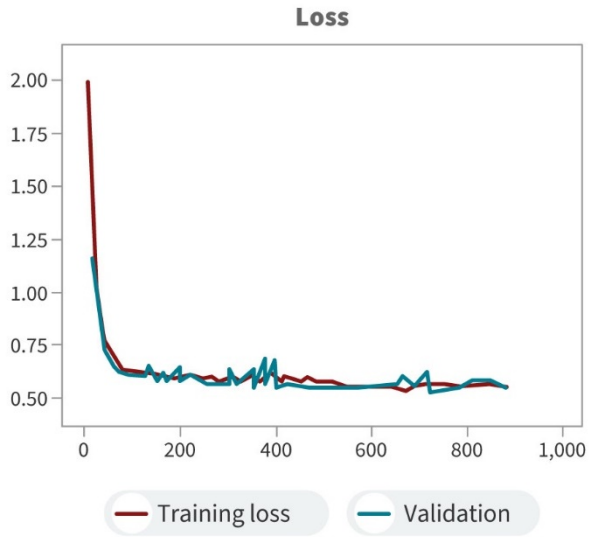
- The loss curve for an **underfit** model usually looks something like this, where both the training and validation loss curves don't decrease much and stay relatively flat lines at high loss, indicating that the model is unable to learn from the training dataset to reduce the loss. Indeed, even seeing just a flat training loss curve, without plotting a validation loss curve, is a good reason to suspect underfitting. Sometimes the curves may also show oscillating noisy values but these will still be centered around high loss values without significant trends of decrease.

Copyright © Stanford University



- The loss curve for an **overfit** model usually looks something like this where the training loss continues to decrease over time, but the validation loss decreases to a point and then begins to increase again.

A good fit is our goal when training machine learning models, and occurs at the sweet spot where the model is neither underfitting nor overfitting.



- When we have a good fit, the training and validation losses will both decrease, at a large rate of decrease initially and then smaller over time, until they reach a point of stability (in other words they converge). People also refer to this as **reaching a plateau**
- Ideally, the training and loss curves will plateau to similar values, but you will often see a small gap between the two where the training curve converges to a lower loss value than the

Copyright © Stanford University

validation loss. This gap between the training loss and validation loss is referred to as the "**generalization gap**", and intuitively we can expect this to happen because the model is directly optimizing to perform well on the training set

A second type of learning curve, the plot of the final performance metric (e.g accuracy) to **visualize model progress during training**

- This is useful to get a sense of actual model performance, since lower loss usually corresponds to better model performance but it doesn't tell us whether the accuracy is at a level we are happy with or not

STRATEGIES TO ADDRESS UNDERFITTING AND OVERFITTING

A model **underfits** when it is unable to learn features from the training set, which means that it performs poorly on both the training set and the validation set.

Reasons why a model underfits:

- The model is not expressive enough for the data that you have.
- The samples in the training set do not have the information required to make the right decisions.

To fix underfitting:

- Train your model for more time
- Increase the capacity of your model

Reasons why the model **overfits**:

- It is learning features that are specific to the training set that cannot be found in the validation set

To fix overfitting, broadly speaking, consider **regularization**– which means to force the model to learn and retain general insights that allow it to extrapolate what it has learned to unseen data.

- **Weight decay** or **L1/L2 regularization** means penalizing the model for using too many of its parameters.
    - Using weight decay means adding a value in the loss that represents the magnitude of the parameter values in the model. If the model has many non-zero or large weights, then the magnitude will be large

Copyright © Stanford University

    - If the model has few non-zero or large weights, the magnitude will be small. Because the model seeks to minimize the loss, this constraint forces the model to have small-valued weights
    - The strength of the effects of weight decay is a very common hyperparameter
- **Dropout** means randomly setting parameter values to zero in the model during training.
    - Picking a layer of the model that randomly "drops out" output values and sets them to zero
    - Intuitively, this makes that layer more unreliable, and the model thus has to build in redundancy into its subsequent layers
    - The need for redundancy means that the model cannot be as complex, hence dropout has the effect of regularization
    - The probability of dropping out a given neuron is a very common hyperparameter.
- **Data augmentation** means randomly warping / transforming the samples in the training set in order to prevent the model from learning any features that are too specific.
    - Common augmentations include: Rotation, random crops, resizing, color and brightness, flipping them horizontally / vertically

The model overfits the training set because it gets too familiar with the samples. Data augmentation constantly changes what the samples look like, so you slow down / prevent the model from memorizing the training set.

Note: Be careful with the transformations. The samples must still be representative of the label they are affiliated with.

STATISTICAL APPROACHES TO MODEL EVALUATION

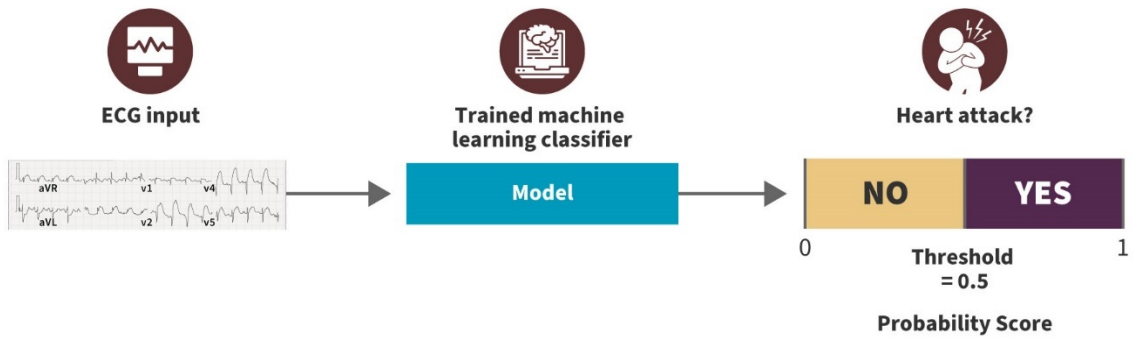
Picking the right metric is critical for evaluating machine learning models.

Accuracy is not always the best metric. If 90% of samples in the test set are benign, then a model that predicts that everything is benign would achieve a 90% accuracy.

Choosing a threshold is not always straightforward.

- The output of a trained machine learning classifier for categorical labeling tasks will typically be a probability score (usually between 0 and 1) for the desired output label
- If we are trying to understand the performance of our classifier in more concrete terms, in particular typical ways in which we might use it in the real world, then we have to choose a "threshold" that will binarize the predicted label into a specific category prediction, i.e. convert that probability to either 0 or 1

Copyright © Stanford University

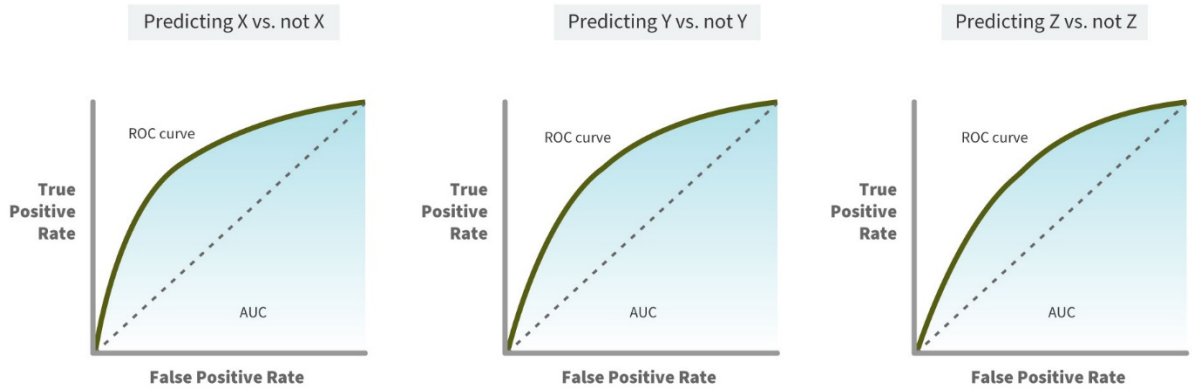


1. The most common approach here is to choose a threshold of 0.5 as the middle ground so that anything greater is a "positive" decision for the label and anything less is a "negative" decision for the label
2. With that threshold the common metrics used in medical testing can then be calculated

- However, 0.5 seems relatively arbitrary given the model's ability to produce more nuanced values

The **receiver operating characteristic (ROC)** curve is a metric for evaluating the model performance that considers all thresholds simultaneously.

1. Algorithms that were trained using discrete labels (such as disease / no disease) are most suited to this approach



2. If our model can detect multiple classes we would plot an AUC ROC curve for each one - so for example, If you have three classes named X, Y and Z, you will have one ROC for X classified against Y and Z, another ROC for Y classified against X and Z, and a third one of Z classified against Y and X

Copyright © Stanford University

In order to understand the implications for our medical task in an ROC analysis, the knowledge on basics of statistical testing is needed. The fundamental analysis of performance for machine learning classification problem is a table that contains different combinations of predicted and actual values, which is known as the **confusion matrix**.

		PREDICTED VALUES	
		POSITIVE	NEGATIVE
ACTUAL VALUES	POSITIVE		
	NEGATIVE		

The table allows us to derive metrics such as: Precision, Recall / Sensitivity, Specificity, Accuracy.

Example: We have a smartphone app that can predict pregnancy using the heart rate function on a wearable device.

- We have a trained our machine learning model
- We want to see how it performs on the hold- out test set of 200 cases with 120 positives (user was pregnant) and 80 negatives (user was not pregnant)
- When we ran our model, it predicted 100 negatives and 100 positives

		PREDICTED VALUES		
		POSITIVE (pregnant)	NEGATIVE (Not pregnant)	
ACTUAL VALUES	POSITIVE (pregnant)	TP = 80	FN = 40	Total actual positive = **120**
	NEGATIVE (Not pregnant)	FP = 20	TN = 60	Total actual negative = **80**
		Total predicted positive = **100**	Total predicted positive = **100**	

- The four boxes of the confusion matrix:
  - True positive (TP): Cases that were positive (pregnant) and our model predicted positive (pregnant)
  - True negative (TN): Cases that were negative (not pregnant) and our model predicted negative (not pregnant)
  - False positive (FP): Cases that were negative (not pregnant) but we predicted positive (pregnant)

Copyright © Stanford University

- False negative (FN): Cases that were positive (pregnant) but we predicted negative (not pregnant)

Metric Definitions

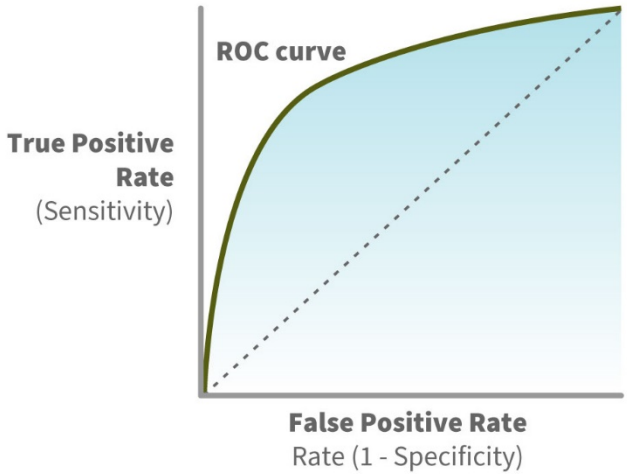
- **Accuracy**: Number of all correct predictions divided by the total number of the dataset. The best accuracy is 1.0, whereas the worst is 0.0.
- **Sensitivity** or **recall**: When the patient is pregnant how often does the test predict pregnant? In other words, out of all positive datapoints, how many did the model predict as positive?
- **Specificity**: When the patient is not pregnant how often does the test predict not pregnant?
- **Precision** (**positive predictive value**): How often is the model correct when predicting positive?
- **Negative predictive value**: How often is the model correct when predicting negative? Note that both positive and negative predictive values are influenced by the prevalence of conditions in the test set and this can be misleading

IMPORTANT METRICS FOR CLINICAL MACHINE LEARNING

The receiver-operating characteristic curve, or **ROC curve** is a plot where the sensitivity of the model is shown on the y-axis and the false positive ratio is shown on the x-axis.

ROC curves enable us to assess the performance of the model over its entire operating range. In other words, we can see what happens to our model's performance with thresholds from 0.0 to 1.0.

evaluate model performance considering a range of thresholds

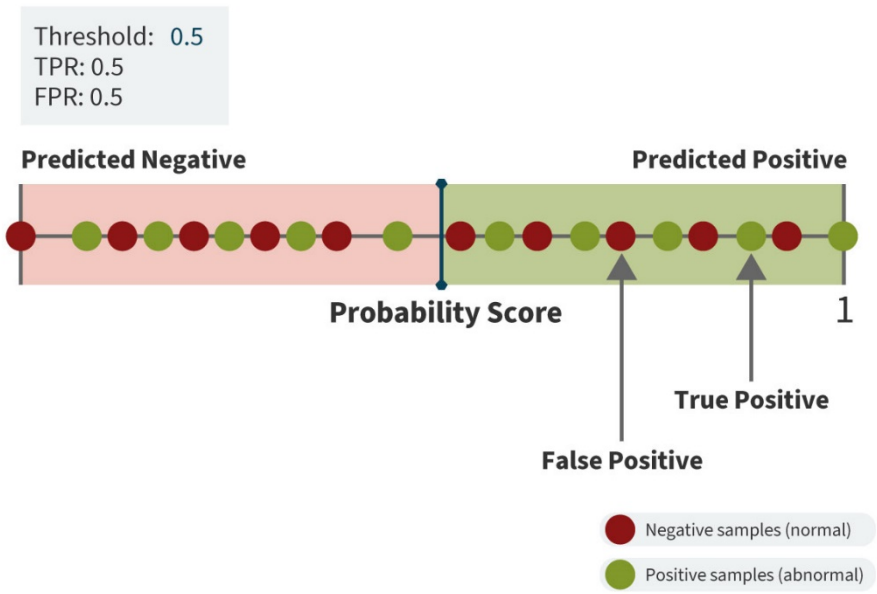


The area under the ROC curve (ROC-AUC or AUROC) gives us a single number that summarizes the efficacy of our model as measured by the ROC curve.

- The maximum AUROC achievable is 1.0– a perfect classifier
- A random AUROC is 0.5– a completely random classifier

How ROC curves work:

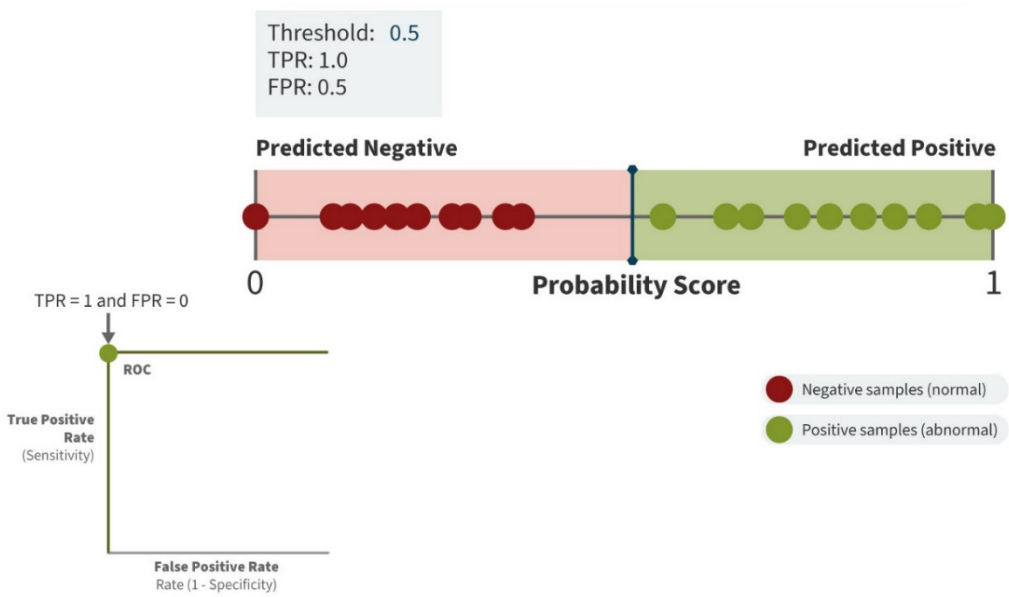
- Our random classifier evaluated on this dataset, will output a random probability score of each example being a positive that's between 0 and 1. We can visualize where each example falls on the probability spectrum like this



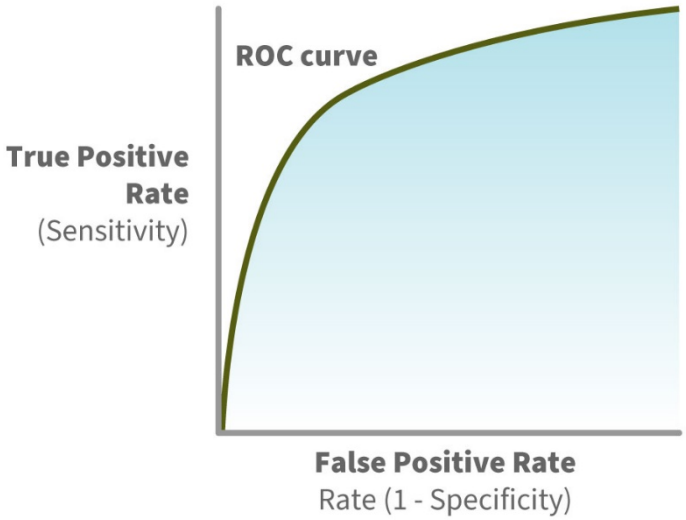
- If we set our threshold to be 0.5 such that all examples with scores above 0.5 are predicted to be positives, in this case you can see that the true positive rate (in other words the number of true positives, which are the red dots in the predicted positive region, divided by the number of actual positives, which are the red dots everywhere) will be about 0.5. Similarly, looking at the red dots, the false positive rate (the number of false positives divided by the number of actual negatives) will be about 0.5
- If we increase our threshold, our true positive rate increases to 0.75, but the FPR also increases to 0.75. In other words, decreasing the threshold increases true positives, but many more false positives are predicted as well. This leads to another point on the ROC curve where TPR and FPR are now both 0.75. In general, every adjustment to the threshold to increase TPR equivalently increases FPR, which creates the diagonal line ROC curve that we showed earlier, with Area under the Curve of 0.5

Copyright © Stanford University

- A perfect classifier, one that predicts a probability score between 0.5 and 1 for every actual positive and a probability between 0 and 0.5 for every actual negative, would have a very different probability spectrum
- For all thresholds of interest, we have a TPR of 1 and a FPR of 0. Thus, the AUROC of this classifier looks like the following:



- You are unlikely to have either a random classifier or a perfect one. The ROC for a "good" classifier would be something like this, with AUC of 0.9



- The probability scores output by the model largely separate the examples, but the model cannot perfectly discriminate between the two classes. So at a threshold of 0.5, you'll have a

TPR of 0.8 and a FPR of 0.2, for example, and at a threshold of 0.4, you'll get a higher TPR of 0.9, but also a less desirable FPR of 0.4

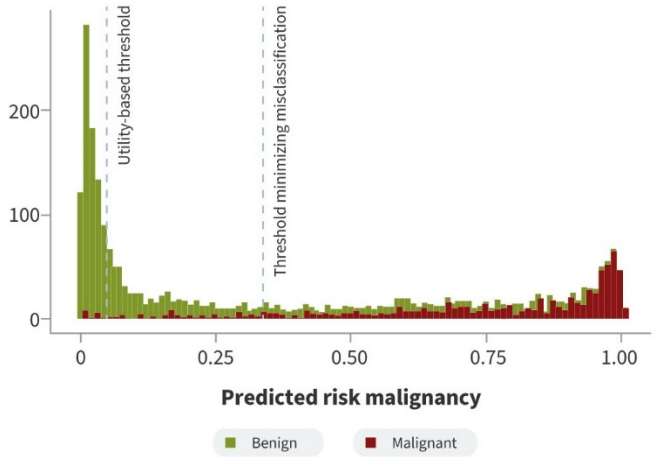
There is a fundamental tradeoff between TPR and FPR: as you increase sensitivity, or TPR, you typically decrease specificity, or 1-FPR. The threshold you choose for the classifier implementation of predicting positive vs. negative is also referred to as the **operating point**.

Comparing the performance between two or more classifiers:

- Method 1: Performance characteristics at particular operating points can be compared
- Method 2: Overall performance can be compared using the AUC.
- **Caution:** many published reports compare AUCs in absolute terms: "Classifier 1 has an AUC of 0.9, and classifier 2 has an AUC of 0.8, so classifier 1 is clearly better". But this does not necessarily hold. Statistical analyses are necessary to verify that these claims are significant

Choosing an operating point:

- Example 1: If we are choosing an operating point for a classifier to screen for cancer, for example, we'd probably rather put up with a higher false positive rate in order to make sure we catch as many true positives as possible as well, since it's most important to identify potential cancer sufferers.
- Example 2: If our classifier will be used to make a decision about whether to pursue a high-risk treatment or not, we probably want a different threshold of sensitivity / specificity trade-off, since we don't want to subject a patient to unnecessary risks unless we are very certain that they need it.



Frequencies of predicted risk of malignancy and possible risk thresholds

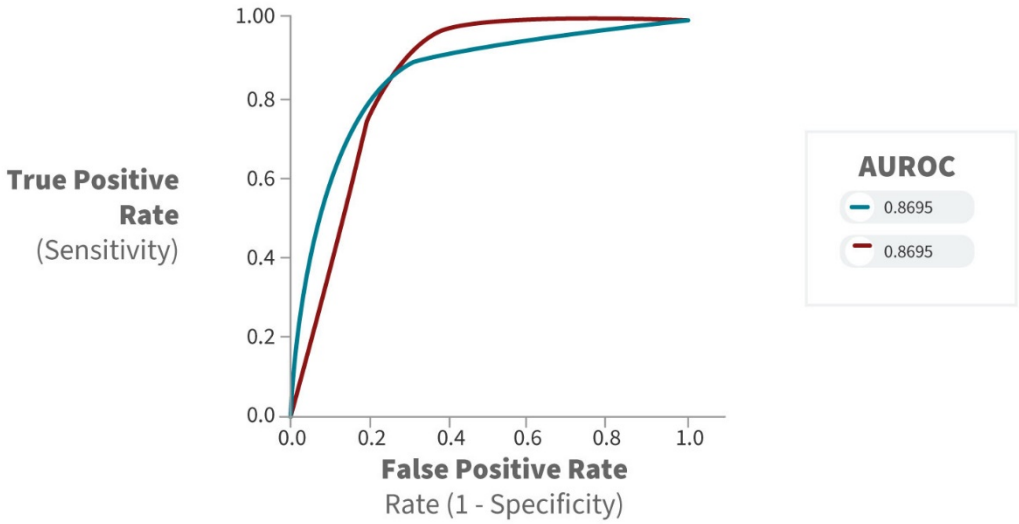
1. By **Minimizing Misclassification**: choose an operating point that minimizes the sum of false positives and false negatives.
2. By **Optimizing Utility:** assign a different utility value for each type of outcome and optimize total utility. NOTE: the utility values are usually subjective!

Copyright © Stanford University

- Choosing an operating point can be based on maximizing a utility measurement. We can subjectively measure utility by manually assigning a value to true positives, false positives, true negatives, and false negatives
    - The utility or cost of different possible outcomes can be expressed in any framework that makes sense for the clinical problem at hand

Other important considerations when using ROC curves:

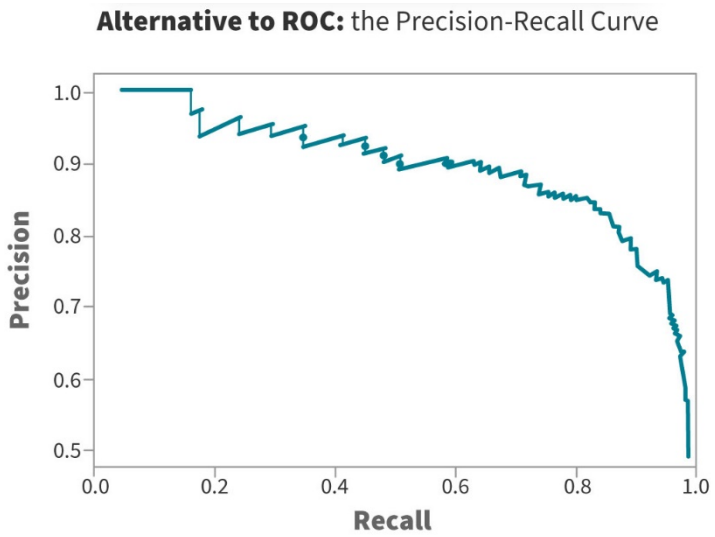
- The shape of ROC curves can matter in evaluation, for example, a classifier can have a lower AUROC than another classifier yet a higher utility because of the shape of its ROC curve



    - At the far left of the plot, the model doesn't predict any positives and so both the TPR and the FPR are zero
    - At the far right of the plot, where the model classifies everything as positive, we have a TPR of 1, but also a FPR of 1 since all true negatives are predicted as positive in this case
    - The shape of the curve as we progress from left to right tells us about the tradeoff between TPR and FPR. If a model's ROC curve arcs towards the top-left more than another model's curve, that tells us that the first model achieves a higher TPR than the second at some fixed FPR
- ROC curves can also be misleading with imbalanced datasets, where the number of true positive labels is very different from the number of true negative labels

Precision recall (PR) curves are more robust to imbalance data.

Copyright © Stanford University



- The y-axis describes precision and the x-axis describes recall at various thresholds
- PR curves cross each other more frequently and it can be more difficult to interpret and compare. However, overall, a curve that appears above another curve in a PR plot generally corresponds to better performance.
- The key difference: the number of true-negative results is not used for making a PRC
- And similar to ROC curves, the AUC (the area under the precision-recall curve) score can be used as a single performance measure for precision-recall curves. As the name indicates, it is an area under the curve calculated in the precision-recall space

The best practice is to evaluate and report both the AUROC curve and area under the PR curve, along with statistical error bars around the average classifier performance so that the most complete information on the performance can be evaluated in general.

CITATIONS AND ADDITIONAL READINGS

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning: data mining, inference, and prediction.* New York: Springer.

Wang, F., Kaushal, R., & Khullar, D. (2020). Should health care demand interpretable artificial intelligence or accept "black box" medicine?. https://www.acpjournals.org/doi/10.7326/M19-2548

Copyright © Stanford University

MODULE 5 - STRATEGIES AND CHALLENGES IN MACHINE LEARNING IN HEALTHCARE

LEARNING OBJECTIVES

- Recognize the pitfalls and utility of correlative and causative machine learning models in healthcare
- Describe the importance of missing and subclass variables in healthcare applications
- Discuss the important concepts and principles behind model interpretability and performance in medicine including approaches for demystifying the "black box"
- Learn the best approach for handling data in clinical machine learning applications including common challenges like missing data and class imbalance
- Understand how dynamic medical practice and discontinuous timelines impact clinical machine learning application development and deployment
- Become familiar with the relationship of data quantity and error or noise and how it can impact clinical machine learning

CHALLENGES AND STRATEGIES FOR CLINICAL MACHINE LEARNING

Correlation vs Causation:

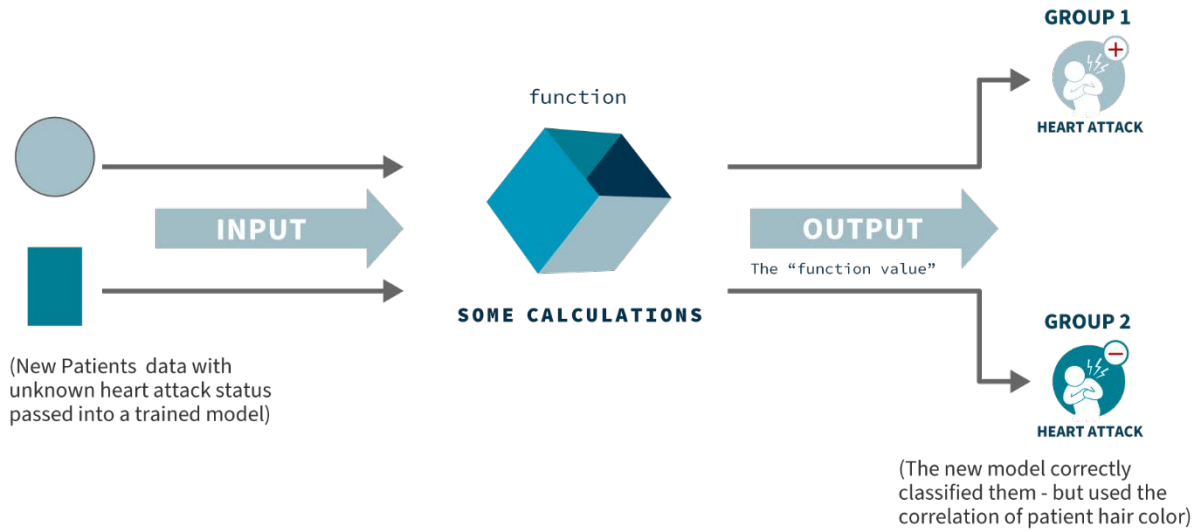
- Machine learning methods such as neural networks work by learning associations between inputs and outputs based on whatever signal it can extract from the data
- It is often difficult, or even impossible, to know if the patterns your model exploits when drawing these associations are the result of correlations in the data rather than causative truths
- Lurking variables: Unforeseen variable which cause a model to fit the data based on useless correlations. Known as common response variables, confounding factors
- A serious issue in machine learning more broadly - when a model exploits unexpected or even unknown confounders that have no relevance to the task, it can severely impair or invalidate the model's ability to generalize to new datasets
- Example 1: "Russian tank problem:" An AI model for identifying tanks recognized the weather in the pictures and not the tanks
- Example 2: Chest Xray image: Model which was thought to be accurate was found to be focusing on non-medical cues in the image to draw conclusions

Copyright © Stanford University

- Example 3: Pneumonia death risk: A high performing model used patient information to identify their risk of dying from pneumonia. It was found that the model was heavily indexing on a correlation between asthma and good patient prognosis in the data
    - In this case the model was not wrong in identifying the correlation between asthma and good patient outcomes
    - However, upon inspection doctors realized that the correlation between asthma and good patient prognosis was the direct result of a hospital policy to admit and aggressively treat asthmatic patients with pneumonia
    - The mistake would be to assume the model's prediction meant that having asthma *causes* a good outcome for pneumonia patients

Sometimes medically irrelevant correlations can still be useful - the best way for this is to **reconfigure the model's application context**, and framing the context of our model applications is something we will spend a lot more time on.

Scenario - Heart Attack Risk. You have one year of retrospective data on 1 million people, labeled with heart attack or no heart attack. You use this labeled data to train a supervised machine learning model to predict heart attack risk within 12 months.



- If you the training data had only medically relevant features, then you may be able to train a medically accurate model which predicts **causal** relationships
- If your model ends up using **correlations** in the data (e.g. between grey hair and incidents of heart attack) it may have different, but still important, use cases
    - If your application context was not for treating patients in a clinic, but instead a model applied for financial, population health or medical practice management, you

could actually be pretty okay if you built an accurate model based only on correlative features in your prediction model

- Model which identifies correlation as well as those which make inferences which are plausibly causal can be useful - the trick is to identify what factors the model is indexing on and to consider the relevance of those factors for a given use case

Two reasons are supervised ML models are prone to solving the wrong problem:

1. By design, they develop their own ways of problem solving independent of the programmer
2. Models lacks contextual knowledge and general common sense - this is why we need multi-disciplinary domain experts to help develop, evaluate and deploy models

A tension between **"black box"** and **"interpretable"** algorithms:

- **Black boxes**: Complex models that can make it difficult to understand exactly how the model made a given decision or prediction
- "Interpretable" model algorithms, often more linear models or models with fewer features, make predictions that are more "explainable"
- Tradeoffs then need to be made as deeper networks with more features are often better predictors, while models with fewer features are easier to visualize, understand and explain

Approaches for increasing interpretability of complex models:

- Leveraging multi-disciplinary teams to review false positive and false negative cases predicted by the model
- Testing the model on external datasets, to try to gain insight into causal vs. correlative features learned by the model.
- Focus on developing computational methods to interpret neural network prediction. One example of this is building "saliency maps"
    - Saliency: The part of an input that matters the most to the model in making its prediction

Different ways to produce saliency maps:

- Class activation maps (CAM): Analyzing the neurons in the final layer of some types of neural networks, to compute how much the neurons that are important for any particular class are firing at every spatial location in an image
- To visualize the relative importance of spatial locations for predicting a particular class, we can plot a weighted sum of the neuron firing in heatmaps. Intuitively, heatmaps show the

Copyright © Stanford University

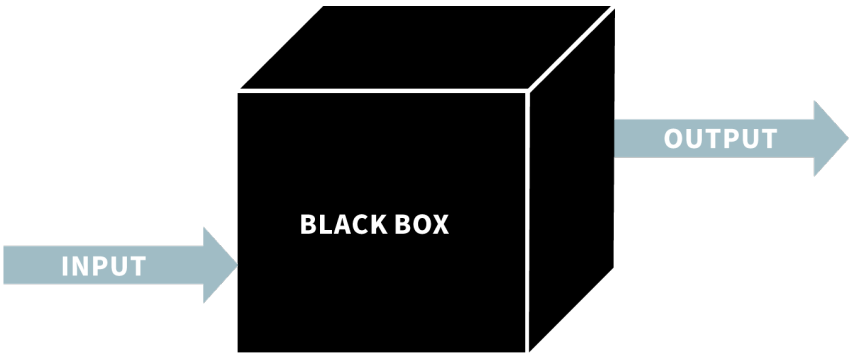
spatial regions that most strongly trigger the firing of neurons important for predicting the class of interest

Another way of codifying saliency:

- We can compute the change in prediction score that would result from a small change in the pixel value at a particular location of the input. Input locations where a change would greatly affect the prediction score can be interpreted as "salient" for the model. Mathematically, we can compute this since this is just the gradient of the classifier score with respect to the pixel values. And we can also compute and plot a spatial heatmap of these gradient-based saliencies

Frequently used vocabulary for talking about the concept of interpretability: **Transparency, explainability, and inspectability**

- Transparent model: Allows us to easily understand how it works
- Explainable model: Should easily communicate why any particular output is produced
- Inspectable model: Should allow us to probe and inspect the functioning of any part of the model
- The terms' more or less mean what they sound like, and are not generally super technical. They are often used in overlapping and overloaded ways, and many people use them more generally to get at the overarching idea of model interpretability. In other words, they are all ways to get at the notion of opening up the black box of complex models such as deep neural networks.



Internal behavior of the code is unknown

It has become quite common these days to hear people refer to machine learning systems as "black boxes". The black box metaphor refers to a system for which we can only observe the inputs and outputs, but not the internal workings. There have been discussions and debates on the topic of interpretability, and referential metaphors about the black box.

Copyright © Stanford University

- There remain concerns about black box models, even if they have been properly vetted and can reliably achieve high performance. We have seen how things can go wrong when models learn spurious correlations in the data
- "clinicians and regulators should not insist on explainability because people can't explain how they work for most of the things they do" - Geoff Hinton

There are two distinct "flavors" of machine learning model explainability: **intrinsic** and **post-hoc** interpretability.

- **Intrinsic interpretability** is simply referring to models, often simple models, that are self-explanatory from the start
- **Post-hoc interpretability** is used to understand decisions by complex models that do not have prescriptive declarative knowledge representations or features

INTERPRETABILITY AND PERFORMANCE OF MACHINE LEARNING MODELS IN HEALTHCARE

Intrinsic interpretability is often advantageous in healthcare. It allows doctors to more easily adjust and add to their interpretation of model predictions

- Example: The LACE index predicts 30-day hospital readmission risk and is calculated using 4 intuitive and transparent feature inputs: Length of current admission, admission acuity, patient comorbidities, and number of emergency department visits in the past 6 months
- Systems like LACE allow clinicians to add their own assessment of the relevant factors. It also allows them to consider other features not included in the LACE model and decide their relative importance

With complex machine learning models which consider multitudes more features than LACE, it is practically impossible to apply intrinsic interpretability. Instead we consider post hoc interpretability on a case-by-case basis.

- In such cases it is be harder to adjust the use of the model based on clinical judgement because it would not be possible to know which features, and combinations of features, contributed to the model's the recommendation

Choosing between performance and interpretability is not easy, and often the choice comes down to trust.

Copyright © Stanford University

- Trust in the development methods, data, and metrics as well as, when available, data about outcomes when using the model are all important.

Use cases suited to black box solutions:

- Text summarization
- Hospital resource triaging
- Pathology slide quantification
- Medical image reconstruction

MEDICAL DATA FOR MACHINE LEARNING

Data types and sources include:

- EHR data
- Both structured and unstructured data types
- Imaging and other pixel based diagnostics
- Genomics
- Peripheral sources of data

In general, it is a good idea to start with a small sample dataset that represents the type of data that you expect will be used in the model. Sample and preliminary analysis and discussion should be done **before** investing a ton of time and resources.

Important things to consider, or to try and glean from your sample data before investing too much time and resources in the project:

- How is the sample data generated?
- How might it fit into a ML workflow?
- What metrics might be useful in evaluating the data?
- How much data might be needed for the project to be successful based on the use case?
- What are the potential use cases in the context of clinical care?
- What is the project's timeline? When and how will data come in?
- Is the data idea needed actually available in the real-world?
- if you are building an application that is expected to produce real-time results, how will the real-time data be sent to the model?

Copyright © Stanford University

- What preprocessing for feature engineering of the data will be needed in order to run the model?

Note that using clean, pre-processed historical data is likely to give an overly optimistic view of models' performance and an unrealistic impression of their potential value. This is one of the strongest arguments to have domain experts and stakeholders involved early on in development.

When evaluating your data, look out for heterogeneous, incomplete, and sometimes duplicative data types that are created in the routine practice of medicine.

- The heterogeneity of data sources and types can complicate the problems of maintaining data provenance, timely availability of data, and knowing what data will be available for which patient at what time

You can run into problems when your dataset includes a relatively small number of examples of one of the labeled output classes, for instance when you are trying to identify a rare event. This is called a **class imbalance:**

- Class imbalance: Refers to the output labels in the data, and where there is a lot more of one label and much less of another label. This is extremely common in many medical datasets

Having imbalance data does not mean you cannot get good results; it effects how much data you may need.

In evaluation, it is important to look out for the accuracy paradox problem.

- The accuracy paradox: Where your model accuracy turns out to be outstanding, but you have a very imbalanced dataset
- If you have a very imbalanced dataset with - say a data set with a ratio of 1:100 abnormal to normal scans - a model may be able to get very high accuracy in predictions by classifying all the scans as normal, as it will be right 99% of the time. Thus, you may have a very high prediction accuracy while having a totally useless, and thus functionally inaccurate model.

There are alternative accuracy metrics and methods of sampling data which help avoid falling into this problem

- Remember that in a classifier model, if you are simply randomly subsampling your total data for the test set to derive your metrics, then your test data will also have the same class imbalance which can skew some metrics that are tied to prevalence like PPV and NPV

Dealing with the accuracy paradox:

Copyright © Stanford University

- Choose the proper metrics and re-evaluate performance of your classifier
- Artificially sample a small hold out test set from your data that contains more of a balance of classes. Where possible, simply collect more data, especially instances of the minority class
- Resample your dataset
    - You may try over-sampling (or more formally known as sampling with replacement)
        - This is best for situations where you do not have a lot of data in the first place and your data is also imbalanced
    - You may try under-sampling (remove instances from the over-represented class)
        - This works best when you have sufficient data for the smaller class
- Adjustment your model to account for the imbalance
    - E.g. Train your machine learning systems in a setting that includes "rewarding" (via math!) the model more for correctly classifying an important rare class than for the more common or prevalent label.
- Think about metrics
    - Pay special attention to Precision-Recall curves when there is a moderate to large class imbalance
    - Rely more on ROC curves if there is even class distribution
- In serious cases, consider sticking with algorithmic approaches that tend to tolerate these imbalances
    - I.e. decision trees (and related algorithms that extend upon them such as random forests!)

It is important to consider how much data you will need to train your model. Often how much data you can get will come down to how expensive/ cumbersome it is to acquire, curate, clean, label etc a good dataset (personnel, licensing fees, equipment run time, etc.).

In most machine learning algorithms, as you increase the size of your dataset, performance grows accordingly and then reaches a plateau. This plateau can vary depending on the complexity of the algorithm.

- For regression and simpler machine learning models you may have heard of the "1 in 10" rule that suggests the need for at least 10 examples of each label class
- For neural networks and data with more complex features a rule of thumb is somewhere around 1,000 examples for each label class

General factors which impact how much data you need for your model:

Copyright © Stanford University

- The number of features in the dataset relative to the number of uncorrelated or weakly correlated attributes in the dataset
- Whether or not model performance is up to par on any number of metrics (including but not limited to accuracy)
    - Making a more complex model and or tuning hyperparameters to increase performance can only improve the model in limited ways and can also run the risk of overfitting. So unless the performance of the model is very close to the goal, the best next step is probably still acquiring more data.

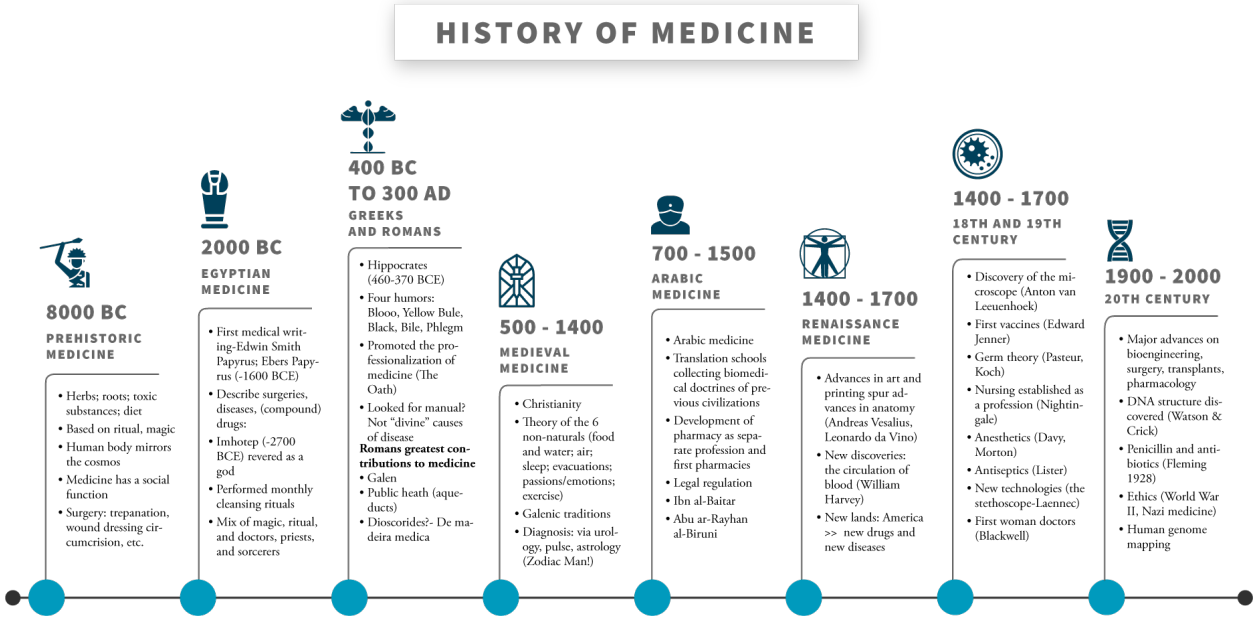
Though more data is often good, naively adding more data may not be helpful, and in some cases could decrease performance, especially in a dynamic field such as healthcare.

There is a crucial distinction between "adding data" and "adding information." Adding more data does not equal adding more information.

In healthcare in particular, we often find that growing datasets by adding historical data often amounts to accumulating arbitrary or outdated correlations.

- As the number of these useless correlations in the data expands it can lead to models that learn correlations and cannot generalize and are limited in practice
- These spurious correlations can be hard to detect and can lead to medical decision making based on false correlations rather than real factors.

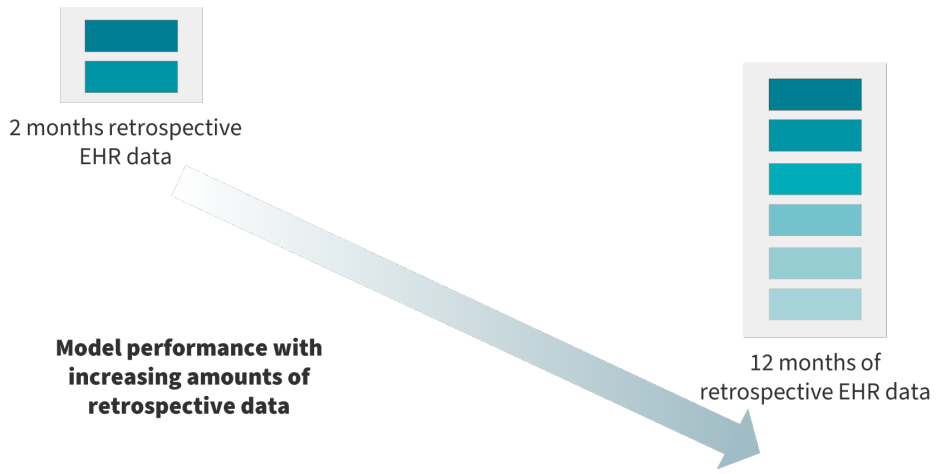
Conceptually, retrospective medical data has a "shelf life" or "expiration date".



Copyright © Stanford University

- The dynamic nature of clinical practices over time challenges the presumption that learning from historical clinical data will inform current and future clinical practices

An important relationship emerges in the separation between when data is generated relative to the time learned prediction models are applied and evaluated.



- Example: For example, Stanford research that used retrospective EMR datasets of varying size found that a small dataset that covers about 2k patients and one month of the most recent data was MORE effective in the final performance of a machine learning prediction model than a much larger dataset composed data collected over a 1 year period.

While old data probably can't be used off the shelf, there are techniques we can use to adjust for the context of the time

- By considering, what were the medical practices at the time? How limited were diagnoses? Etc. it may be possible to salvage data, and may be worth doing in some circumstances, but doing so may introduce more noise than relevant information.

Trained clinical models in healthcare that are able to incorporate a continuous stream of data could allow automated methods to rapidly detect and adapt to shifting practice changes to avoid hitting an "expiration date" for effectiveness.

"Garbage in garbage out!" - bad data will result in bad models. No matter how sophisticated the machine learning algorithm is or the data engineering techniques

The choice of data and problem to solve is infinitely more important than the algorithm or the model. **We want high quality data.**

The assessment of, and methodology to create a high-quality dataset are not standardized.

Copyright © Stanford University

- Among other things, this means that data coming from different sources may vary in its organization. In particular, phenotyping is very important for models that are expected to be deployed across hospitals.
- To help with this problem, when you choose data for any model learning exercise, the label of interest should be noted and described in your work in a *reproducible* manner.

It is also important to be clear how your data and labeling scheme relate to ground truth.

- Labels like mortality have a relatively straightforward relation to readily available determinations of ground truth
- With other labels, like pneumonia, it can be much more difficult to codify ground truth, as that truth may only be expressed in clinical and medical imaging data, which can be hard to mechanistically interpret, and can be fraught with inaccuracies as well as confounding information.
- Look out for labels (like with diabetic patients) that rely on numerical cutoffs that can changed over time in medical practice, and or those that vary by age in terms of the upper and lower bounds.
    - For these labels the consideration of data "shelf life" and treatment changes is very important.

We can expect that our labels will not be 100% accurate compared to a ground truth, thus we need to find ways to **estimate and understand our label noise.**

- To evaluate the noise in your labels in a large dataset, start by taking a subset of the data, then use best practices (often with domain experts) to label this subset of data using multiple reviewers. From there, compare the agreement to the original label to evaluate the accuracy of the original labels.
- Note that the label noise could be also a reflection of the difficulty of the labelling task. To investigate the difficulty of a task, try to determine if there is disagreement in the labels among experts.

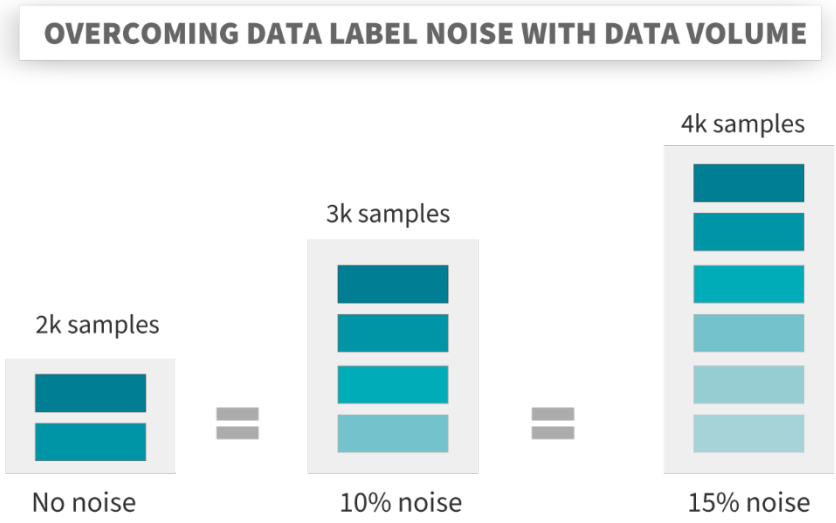
There are many strategies to address label noise, many of which are outside the scope of this course, but some of the simpler approaches include triangulating multiple labels together.

- This approach works towards reducing noise by adding additional confirmation labels. Applying multiple noisy labels can help narrow down a cohort closer towards ground truth, nearly invariably at the expense of dataset volume.
- After adding the additional labels, you can compare again any change in noise with a subset labeled by hand.

Copyright © Stanford University

It is important to note that data label noise is inevitable and even very noisy data can train a very good model.

- In fact, there are many cases in which you can overcome data label noise by increasing data volume.



- A study showed that the rule of thumb result was that with 10% noise you need 50% more data and if there is 15% noise you need to double the data.

Formally, the version of supervised learning that involves noisy, bad, or "weak" labels is called **weak supervision**.

MODULE 6 - BEST PRACTICES, TEAMS, AND LAUNCHING YOUR MACHINE LEARNING JOURNEY

LEARNING OBJECTIVES

- Describe best practices for developing and evaluating clinical machine learning applications
- Understand the "output-action" pairing as a framework related to machine learning in healthcare applications
- Learn what skillsets are useful for multidisciplinary and diverse teams and how each contributes to success

Copyright © Stanford University

- Recognize the basic challenges around regulatory and ethics related in clinical machine learning
- Become familiar with challenges with human computer interaction for machine learning applications including automation bias and the consequences in healthcare
- Analyze the potential impact on the future clinical workforce by machine learning on delivery of healthcare

DESIGNING AND EVALUATING CLINICAL MACHINE LEARNING APPLICATIONS

Understanding of clinical machine learning project development - The name of the game is to "find problems worth solving".

One of the ways to understand the value of the potential solution is to consider all actions and repercussions that would come from a solution - and this is where having input from multiple stakeholders and domain experts is important.

FINDING PROBLEMS WORTH SOLVING

	SCIENCE	PRACTICE	DELIVERY
CLASSIFY	Finding sybtypes of heart failure with preserved injection fraction	Who might be at high risk for a thromboembolism?	Who is burnt out?
PREDICT	Estimating the disease risk conferred by genetic variations	Which patients are at risk of dying in the next 3-12 months?	Who will be a no show?
ACT/TREAT	XYZ solid tumors can be treated by allogeneic chimeric antigen receptor T-cell By	What is a good second line drug to manage diabetes after metformin?	Request four back up nurses on Wed, for the Ortho OR.

Three categories of model application:

- Scientific exploration / discovery

Copyright © Stanford University

- Clinical care / decision support
- Care delivery / managing medical practice

Three categories of model output we think about:

- Classification, prediction, recommendation

The Output Action Pair (OAP): The action that will result from the model output.

- Consider what a correct model prediction will entail
- Consider what an incorrect model prediction will entail

Without the right problem, it doesn't matter if you have the best scientists, infinite computational resources, or the most perfect dataset.

Utilizing the OAP Framework:

1. Suppose we would like to design a machine learning project to reduce deaths from sepsis in the ICU
2. Assume we have a prediction + practice sepsis model that can ingest real-time patient EMR data, and then predict the patients likely to develop sepsis at the point of care.
    - The output will be the prediction and the action will be an alert to the clinical team if positive above a set threshold.
3. Output here will be a sepsis diagnosis, which will be the label that we will use to build our model. What is the definition of sepsis?
    - In this case sepsis has distinctly different definitions depending on what you are attempting to address.
    - One definition is the Sepsis-3: This is a medically accurate consensus definition that uses specific clinical criteria and applied to a patient as a formal diagnosis
    - Another definition of sepsis is the Medicare sepsis identifier SEP-1.
        - A quality measure used by Medicare and Insurance reporting for billing and quality reporting
        - In contrast to the other definition, this measure does not represent medically useful sepsis definitions
        - This definition considers only a subsample of patients in a given hospital so you would not have all the sepsis patients labeled with this approach
4. Be sure that the labeling procedure matches the objective of your model

Copyright © Stanford University

OUTPUT	ACTION/COST/RESULT
TRUE	Escalation in sepsis related medical therapy - possible improved outcome - increased costs of care possibly outweighed by earlier intervention - avoid costs/consequences of untreated sepsis
TRUE NEGATIVE	Avoid unneeded sepsis related medical therapy - no expected change in outcome - reduce unneeded costs
FALSE POSITIVE	Escalation in medical therapy - unclear impact on clinical outcome - increased costs of care - detrimental impact on model effectiveness/adoption
FALSE NEGATIVE	Avoid needed sepsis related medical therapy - worsened outcome related to delayed sepsis therapy - increased cost of untreated sepsis - detrimental impact on model effectiveness/adoption

5. OAP utility analysis affords a rough understanding of the minimum acceptable performance and how the output would lead to action in many possible scenarios
6. Consider how humans will interact with the model in production
    - Models are often evaluated using statistical metrics for success:
        - Positive predictive value, sensitivity/recall, specificity, calibration
    - Factors to consider:
        - Lead time offered by the prediction
        - The existence of a mitigating action
        - The cost intervening and the cost of false positives and negatives
        - The logistics of the intervention
        - Incentives both for the individual and the healthcare system
        - Alert fatigue
        - Cognitive biases
    - Models could lead to complacency from those who begin to trust the model too strongly

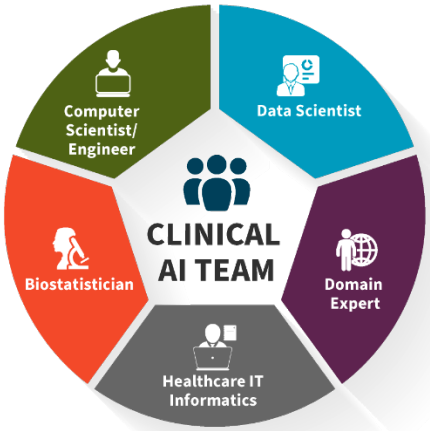
It is important to build a team with diverse expertise:

- Build a team with expertise in: clinical medicine, clinical trials, statistical study design, healthcare finance and incentives, data primary and biases, end user environment

Copyright © Stanford University

- Not everyone on the team needs to be able to write out the math that explains how weights are updated in backpropagation
- The entire team benefits if everyone has a high-level understanding of machine learning concepts and principles because that foundation of knowledge serves as a common language allowing everyone's unique expertise and experience to apply to the problem

Archetype areas of expertise:



- Data scientist
  - Focused on data mining, feature engineering, analytics, metrics of model(s) performance
  - A deep knowledge working with healthcare data in this role is critically important because this role requires delivering and manipulating the data that the rest of the team can work with
  - Feature engineering, pre-processing, and other tasks that will be key to a successful project like preliminary simple model building to get a sense of which machine learning approach is best or what features to use or not use
- Machine learning engineer
  - An expert in computer science, that would ideally team up alongside the data scientist co-developing the model
  - Focusing on the machine learning techniques needed to obtain high-performing models
  - May also play a leading role in writing the more formal code for final software deployment, and the entire workflow pipeline, in other words building out formal, production-ready models and setting up the tools to integrate them with the rest of the clinical enterprise

Copyright © Stanford University

    - Often the machine learning engineer is knowledgeable in more advanced machine learning techniques, especially deep learning, computer vision, and natural language processing
- Statistician
    - Help form conclusions safely beyond the data analyzed and back that up with either trial design or statistical analyses
    - In a lot of circumstances, the statistical skills sound a lot like our discussion of data scientists evaluating pilot model performance - after all there is a lot of statistical knowledge needed to evaluate models and review metrics
- Healthcare IT
    - Integration and deployment in a healthcare environment. It is incredibly common for teams to build models that work only to hit a long delay in integration
    - It is important especially for models that are geared toward clinical deployment to engage healthcare IT professionals early in the model development process
        - Knowledgeable about the details of when and where certain data become available, whether the mechanics of data availability and access are compatible with the model being constructed, and the important interactions within the existing healthcare ecosystem
- Domain expert
    - Provide context and guide the development of the overall application, help decide metrics, and make key development decisions including where to choose a threshold
    - What populations and data should be included, and how deployment might take shape?

Domain Experts	Category	Examples of Applications
Device product developers, clinicians, end users (patients and families)	Health monitoring Benefit/risk assessment	Devices and wearables Smartphone and tablet apps, websites
	Disease prevention and management	Obesity reduction Diabetes prevention and management Emotional and mental health support
	Medication management	Medication adherence

Copyright © Stanford University

	Rehabilitation	Stroke rehabilitation using apps and robots
Clinician care teams	Early detection, prediction, and diagnostics tools	Imaging for cardiac arrhythmia detection, retinopathy Early cancer detection (e.g., melanoma)
	Surgical procedures	Remote-controlled robotic surgery AI-supported surgical roadmaps
	Precision medicine	Personalized chemotherapy treatment
	Patient safety	Early detection of sepsis
Public health program managers	Identification of individuals at risk	Suicide risk identification using social media
	Population health	Eldercare monitoring
	Population health	Pollution epidemiology Water microbe detection
Healthcare administrators	International Classification of Diseases, 10th Rev. (ICD-10) coding	Automatic coding of medical records for reimbursement
Healthcare administrators	Fraud detection	Health care billing fraud Detection of illegal prescribing patterns
Healthcare administrators	Cybersecurity	Protection of personal health information
Healthcare administrators	Physician management	Assessment of quality of care, outcomes, billing
Geneticist	Genomics	Analysis of tumor genomics

Copyright © Stanford University

Pharmacologist	Drug Discovery	Drug discovery and design

GOVERNANCE, ETHICS, AND BEST PRACTICES

At the very least there should be a plan for medical data stewardship that everyone involved in the project can agree to. It is critical that all members of the team be trained and follow strict best practices when working with healthcare data, even if de-identified, since a breach or leakage of data could be catastrophic to the project.

Medical data stewardship can be very different than other forms of data.

Up front training for all who are involved can ensure that ensure that everyone involved in the project has had at least basic medical research and data stewardship training.

- Be careful of the 'context transgressions' that can occur in collaborations or partnership where data may flow between medical, social and commercial contexts governed by different privacy norms

De-identified patient data is not considered private health information because of the anonymization process. However, research has shown that it may be possible to re-identify de-identified patients given the right kind of data.

When curating large medical datasets for clinical machine learning applications, it is important to ensure that the data will not be used in a way that could cause harm or is unethical. Members of the development team and ecosystem must share a common understanding of the ethical and regulatory issues around the development and use of clinical machine learning tools.

When selecting members of the team it is critically important to consider how the project will promote equitable clinical machine learning solutions that does not suffer from bias.

- Working to ensure that new clinical machine learning applications are free from biases can include the composition of the team, and forming a team that is diverse with respect to gender, culture, race, age, ability, ethnicity, sexual orientation, socioeconomic status, privilege, etc.

The development of health care AI tools requires a diverse team with varying skillsets:

- Information technologists
- Data scientists
- Ethicists and lawyers

Copyright © Stanford University

- Clinicians
- Patients
- Clinical teams
- Organizations

These teams will need a macro understanding of the data flows, transformations, incentives, levers, and frameworks for algorithm development and validation, as well as knowledge of ongoing changes required post-implementation.

Consider machine learning tools that, if successful, can achieve significant cost reductions or improved patient outcomes and create further competitive advantage and exacerbate existing healthcare disparities.

Best practices should be driven by an implementation science research agenda and should engage stakeholders with an effort to reduce cost and complexity of AI technologies. In summary - the best clinical machine learning outcomes will come from team-based approaches composed of individuals with complimentary skillsets, essential expertise, and diversity of backgrounds.

HUMAN FACTORS IN CLINICAL MACHINE LEARNING - FROM JOB DISPLACEMENT TO AUTOMATION BIAS

Tasks that can be automated:

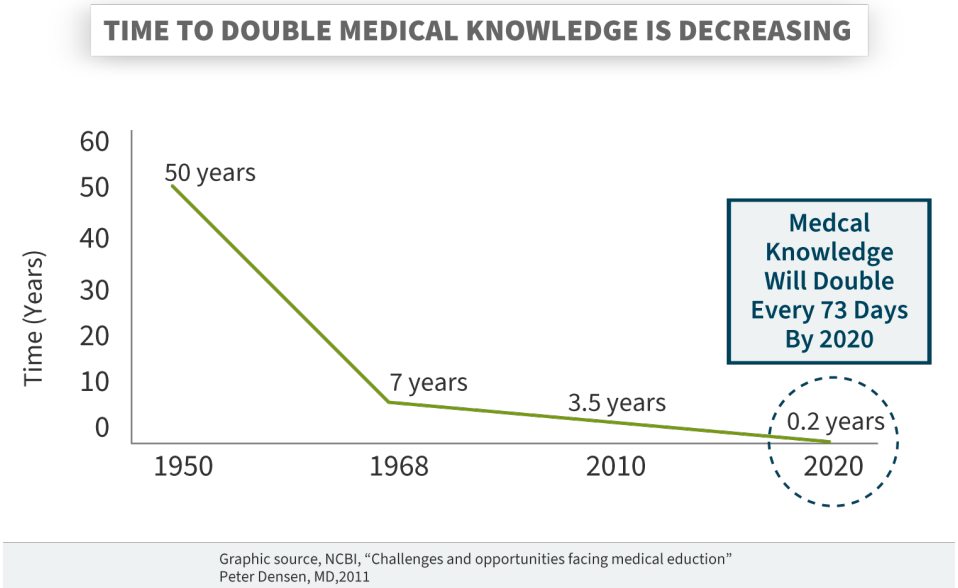
- Transcribing clinical documentation
- Image analysis
- Billing and coding
- Practice management
- Staffing and resources optimization
- Prior authorization forms
- Triaging routine diagnosis

Machine learning applications remain unlikely to displace many human skills such as complex reasoning, judgment, analogy-based learning, abstract problem solving, physical interactions, empathy, communication, counseling, and implicit observation.

Copyright © Stanford University

- Gradually there will thus likely be a shift in health care toward jobs that require direct physical(human) interaction, including specialized surgical or procedural specialties, which are not so easily automated

It is increasingly evident that a transition into the machine learning era of health care will require education and redesign of training programs.



- Medical training institutions already acknowledge that emphasizing rote memorization and repetition of information is suboptimal
- Medical knowledge doubles every 3 months on average

Jobs could be lost or replaced, but the process of the "machine learning healthcare" era will create new jobs for scientists and engineers who understand healthcare and healthcare professionals who understand machine learning.

Important aspects of this new machine learning medical hybrid profession will include:

- Training machine learning systems and deliberate effort to evaluate and stress test them
- Leading multi-disciplinary teams within the healthcare system to provide ongoing machine learning education and guide strategy for clinical practice
- Ongoing evaluation and testing of machine learning models in the active healthcare environment, in particular due to the dynamic and changing healthcare landscape

Education for the healthcare workforce:

Copyright © Stanford University

- Principles and impacts of machine learning
- How to interpret the model recommendations
- The flaws and biases
- How to identify unintended consequences of machine learning system behavior

There are further risks of machine learning in healthcare which could negatively impact the field

- **Deskilling (aka "skill rot")**: A risk of over-reliance on computer-based systems for cognitive work

"Automation bias" refers to the fact that when humans have the guidance of automated systems they begin to act as if they are in a lower risk environment.

- **Automation misuse**: When a healthcare work's inherent trust in an automated system leads to overreliance on automated aids resulting in a decreased performance
- The airline industry has found that the design of many decision support systems has contributed to problems such as automation bias and automation misuse. The pilots performed best when the automated system recommendation was presented with a trend display of system confidence
- Engineering systems that could provide a recommendation AND the system probability calculation provides best performance

Calibrated risk could allow for better clinical decision making:

CANCER RISK	CANCER RISK CATEGORIES
0-25%	Low
26-75%	Moderate
76-100%	High

- More specific numerical risk scores may provide better information and lead to more nuanced treatment discussions than predicting simply "low risk"

Copyright © Stanford University

- A calibrated continuous risk estimate is more informative and allows the user to set their own thresholds or decide if they will "trust" the output - avoiding the hazards of automation bias and misuse

CITATIONS AND ADDITIONAL READINGS

Kelly, C. J., Karthikesalingam, A., Suleyman, M., Corrado, G., & King, D. (2019). Key challenges for delivering clinical impact with artificial intelligence. *BMC medicine*, *17*(1), 195. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6821018/

Office, U. (2020, January 21). Artificial Intelligence in Health Care: Benefits and Challenges of Machine Learning in Drug Development [Reissued with revisions on Jan. 31, 2020.]. https://www.gao.gov/products/GAO-20-215SP

Wynants, L., van Smeden, M., McLernon, D.J. *et al.* Three myths about risk thresholds for prediction models. *BMC Med* **17,** 192 (2019). https://bmcmedicine.biomedcentral.com/articles/10.1186/s12916-019-1425-3

THINGS TO REMEMBER

Simplicity has benefits. If you go with the simplest approach you need, you will likely be happier with the final implementation.

- Start with a minimal set of data and to learn, based on the metrics, how to drive the decision about how much data you ultimately need

Be skeptical of your data, your model, and any metric numbers no matter if they are bad but especially if they are good.

Dig into the data with your team.

- Take a close look at the false positives/false negatives cases in detailed error analyses with your team to look for error patterns
- Look for systemic biases and really try and explain why the model fails in certain circumstances

Copyright © Stanford University

Work toward acquiring external datasets if it makes sense for the prediction-output pairing, provided you can line up the labels and data types, because it is an incredibly useful way to get a true sense of your model performance.

Ensure that the test set is as free from noise as possible, is independent, and is truly providing all the information needed to make decisions on performance.

Be wary of hidden data subgroups not reflected in your test set.

Incorporate multi-disciplinary teams with feedback loops to catch and improve on errors.

Copyright © Stanford University