# Advanced Machine Learning

Lecture 1

# What is Machine Learning?

*“The use and development of* ***computer systems*** *that are able to learn and* ***adapt without following explicit instructions****, by using algorithms and* ***statistical models*** *to analyze and draw inferences from patterns in data.”*

*Oxford English Dictionary*

# A more technical description

Given a set of training examples $(x_1, x_2, ..x_N)$ potentially with corresponding labels $(y_1, y_2, ..y_N)$ we would like to infer some properties of the population distribution $p(x, y)$

Name some ML tasks, what is x and y and how the desired output is related to $p(x, y)$ ?

# Why is ML Hard?

- The data x are often high dimensional
- The number of training samples N is limited
- We may not have labels y for all data
- The probability density $p(x, y)$ can be quite complex

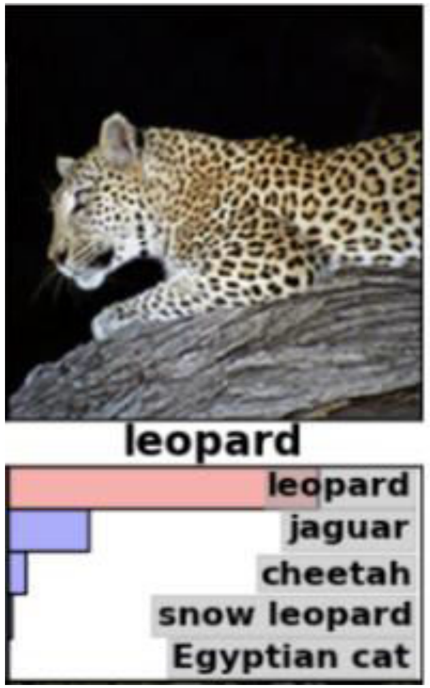
What are some other issues that make ML hard?

# Running example

- Let us introduce a running example for this lecture.
- ImageNet is a large-scale dataset (maybe, mid-scale these days)
- ~1 million high-resolution images ($x_1$,$x_2$..$x_n$), n=1e6
- Images labelled as 1 of 1000 categories ($y_1$,$y_2$..$y_n$) $y_i \in [1, 1000]$

# Two important directions

- Machine learning tasks are often split into 2 categories:
- Discriminative: estimate P(y|x)
    - Example: object category classication given image
- Generative: estimate P(x|y)
    - Example: generate new images given category label



# More exciting than it sounds!

- The formal descriptions sound a little dry
- We’ll see that it important to have a clear formalism
- Many super important (and/or cool) applications
- One of the faster moving part of computer science at the moment

# Discriminative examples

- Face recognition methods can recognize 1 in $10^9$ people
- Task: estimate $p(y|x)$

# Generative Example

- DallE-2/Imagen/Stable Diffusion 2 generate images by text guidance


A small cactus wearing a straw hat and neon sunglasses in the Sahara desert.

Give examples of the most important applications of discriminative and generative ML methods

# Relation between ML to Statistics

- ML is at the intersection of computer science and statistics
- CS: can we find the solution with high computational efficiently?
- Statistics: can we learn with a small number of samples?
- See: Machine learning: Trends, perspectives, and prospects, Science 2015
- In practice:
  - ML often does not assume distribution, statistics often does
  - ML cares less about confidence bounds, core task for statistics
  - Modern ML methods are often more heuristic

# Relation between ML to Optimization

- ML: often finding parameters to optimize some loss objective
- This is within the scope of optimization – a deep, older discipline
- Most optimization theory applicable for convex objectives
- Modern ML objectives are not convex

# Topics for This Course with Estimated Times

- Weeks 1-2: Review of supervised learning and deep NNs
- Weeks 3-7: Generative models
- Weeks 8-10: Representation learning
- Weeks 11-12: Learning with limited supervision
- Weeks 13-14: Miscellaneous

# Weeks 1-2: Supervised Learning, Deep NNS

- Week 1:
  - Motivation for ML and general problem definition
  - Course overview and requirements (**you are here**)
  - Review of key supervised learning material from IML
- Week 2:
  - Definition of deep learning and basic building blocks
  - A general framework for modern deep architectures
  - Applications

# Course Details

- The lectures will take place on Wednesday 10-13
- My reception hour is at Wed at 9 (might change)
- You must physically attend 50% of the lectures
- Students who attend more lectures, will receive a small bonus
- Two compulsary practical exercises (20% of the grade)
- Final exam (80% of the grade)

# Course Etiqutte

- Do come to the lectures
- Do ask questions
- Do come to speak to me during the break, even just for an intro

# Supervised ML

- Given:
  - A set of training examples $x_1,x_2..x_n$ sampled IID from distribution P
  - Corresponding labels $y_1,y_2..y_n$
- Objectives: learn a function f, which can predict y for all x in P
- Examples: ImageNet classification, face recognition

# Fitting to Training Set

- Let's assume our training set is massive, no difference from test
- Want to find the function that maps x to y
- Assume we specify this function by a set of parameters (weights) W
- Need an objective L(W), s.t. lowest value corresponds to best solution

# First Attempt - Accuracy

- The naïve idea iis to directly optimize accuracy

$$L(W) = \sum_i 1_{f(x_i) \neq y_i}$$

- Problem: not continuous, cannot efficiently optimize weights W

# Better Solution: Cross-Entropy

- Let us assume that f outputs a probability over the K output classes
- We use the differentiable cross-entropy objective:

$$L(W) = -\sum_{x_i}\sum_{k} p(\tilde{y}_i = k) log(f_W(\tilde{y}_i = k))$$

- As the labels are assumed to be deterministic this simplifies as:

$$L(W) = -\sum_{(x_i, y_i)} log(f_W(x_i)[y_i])$$

# Optimization

- Many methods were proposed to find the best weights W
- Hard to beat Stochastic Gradient Descent (SGD) in practice

$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$

- Computing this looks like O($N^2$), but due to backpropgation is O(N)

# SGD with Momentum

- Momentum typically used in practice
- Intuition: reduces noise by biasing step in direction of previous steps

- Choosing the learning rate, momemtum is very important
- Many hand-crafted or automatic tricks for doing so, not our scope

**Gradient Descent Update Rule**

$$w_{t+1} = w_t - \eta \nabla w_t$$

**Momentum based Gradient Descent Update Rule**

$$v_t = \gamma * v_{t-1} + \eta \nabla w_t$$

$$w_{t+1} = w_t - v_t$$

# Adaptive Optimizers

- SGD does not automatically adjust step size
- Adaptive methods do e.g. ADAM, AdamW
- Compute average gradient and square norm
- Scale momentum by average gradient norm
- Less sensitive to parameters than SGD-M

$$
\begin{aligned}
&\textbf{for } t = 1 \textbf{ to } \ldots \textbf{ do} \\
&\quad \textbf{if } \mathit{maximize}: \\
&\quad\quad g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1}) \\
&\quad \textbf{else} \\
&\quad\quad g_t \leftarrow \nabla_\theta f_t(\theta_{t-1}) \\
&\quad \theta_t \leftarrow \theta_{t-1} - \gamma\lambda\theta_{t-1} \\
&\quad m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t \\
&\quad v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\
&\quad \widehat{m_t} \leftarrow m_t / \big(1-\beta_1^t\big) \\
&\quad \widehat{v_t} \leftarrow v_t / \big(1-\beta_2^t\big) \\
&\quad \textbf{if } \mathit{amsgrad} \\
&\quad\quad \widehat{v_t}^{max} \leftarrow \mathrm{max}(\widehat{v_t}^{max}, \widehat{v_t}) \\
&\quad\quad \theta_t \leftarrow \theta_t - \gamma \widehat{m_t} / \big(\sqrt{\widehat{v_t}^{max}} + \epsilon\big) \\
&\quad \textbf{else} \\
&\quad\quad \theta_t \leftarrow \theta_t - \gamma \widehat{m_t} / \big(\sqrt{\widehat{v_t}} + \epsilon\big)
\end{aligned}
$$

# Empirical Risk Minimization (ERM)

- So far, we assumed the training set was very, very large
- In practice, the training set is limited
- We minimize the empirical risk = error on training set
- Really care about true risk = error on all images in the world
- In other words, we want **generalization**

# Formal Definition of PAC Learning

- PAC = probably approximately correct

DEFINITION 3.3 (Agnostic PAC Learnability) A hypothesis class $\mathcal{H}$ is agnostic PAC learnable if there exist a function $m_{\mathcal{H}} : (0,1)^2 \rightarrow \mathbb{N}$ and a learning algorithm with the following property: For every $\epsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, when running the learning algorithm on $m \geq m_{\mathcal{H}}(\epsilon, \delta)$ i.i.d. examples generated by $\mathcal{D}$, the algorithm returns a hypothesis $h$ such that, with probability of at least $1 - \delta$ (over the choice of the $m$ training examples),

$$L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon.$$

*Definition from Shai Shalev Shwartz's book*

# Classical Bounds on Learnability

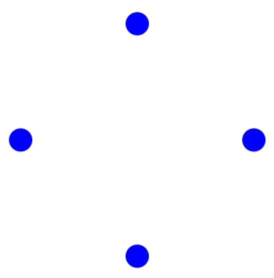
- In IML, you have learned to bound the number of examples:

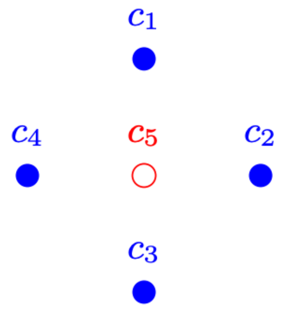
$$m_{\mathcal{H}}(\epsilon, \delta) \leq m_{\mathcal{H}}^{UC}(\epsilon/2, \delta) \leq \left\lceil \frac{2 \log(2|\mathcal{H}|/\delta)}{\epsilon^2} \right\rceil$$

- Which contains the number of possible configurations of parameters
- Infinite for continuous H, but can be made finite by discretization

# VC-Dimension

- A major breakthrough by Vapnik and Chervonenkis
- Learnability is measured by set size that can be “shattered” by H
- Quantified by VC dimension *d*
- Sometimes finite, even for continuous hypothesis classes
  - Linear classifier with n parameters -> d scales with n






$$m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon}$$

# When Theory from IML Works Well

- The theory we learned in IML works when hypotheses are simple
- Example: linear functions, decision trees
- Really useful to know how many examples are needed for learning
- A big deal for making decisions in science and industry
  - Budget allocation
  - Go/ no go etc.

# Real World Hypothesis Class Requirements

- The classifier functions must satisfy several properties
- Being expressive
  - providing a good fit for the training data
- Being amenable to optimization
  - need to find function parameters minimizing the objective at reasonable time
- Generalization
  - Not requiring too many training samples for reducing the true risk

# Linear functions are often insufficient

- Linear models or simple trees are often not enough
- They are good at:
  - Optimization - they are easy to optimize
- They are ok at:
  - Generalization - they do not overfit much for not too many parameters
- They are poor at:
  - Expressivity – they cannot classify complex data e.g. pixel to object category

# Compare this to deep NNs

- They are good at:
  - Expressivity – they can classify complex data e.g. pixel to object category
  - Generalization – they overfit much less than their parameter count suggests
- They are ok at:
  - Simple optimization algorithms tends to find good local optima
- They are poor at:
  - Optimization takes a lot of time (much compute and memory)

# Classical ML Theory and Modern Methods

- Classical ML theory does not generlize well to deep networks:
  - Deep NN should overfit
  - SGD on deep NNs should not converge

What do you think is the place of theory in modern ML?

# Improving Generalization

- In classical ML expressivity comes at the price of generalization
- Improving generalization can occur by giving up expressivity
- We would like to give up in areas that are not useful
- Simplest method: choose a suitable hypothesis class
    - Are trees or linear models most suitable for my task?
    - Do I really need deep NNs?

# Regularization

- Choose very expressive function class
- Reducing unneccesary expressivity by:
  - Forcing weights to be small

$$L_{reg}(W) = |W|^2$$

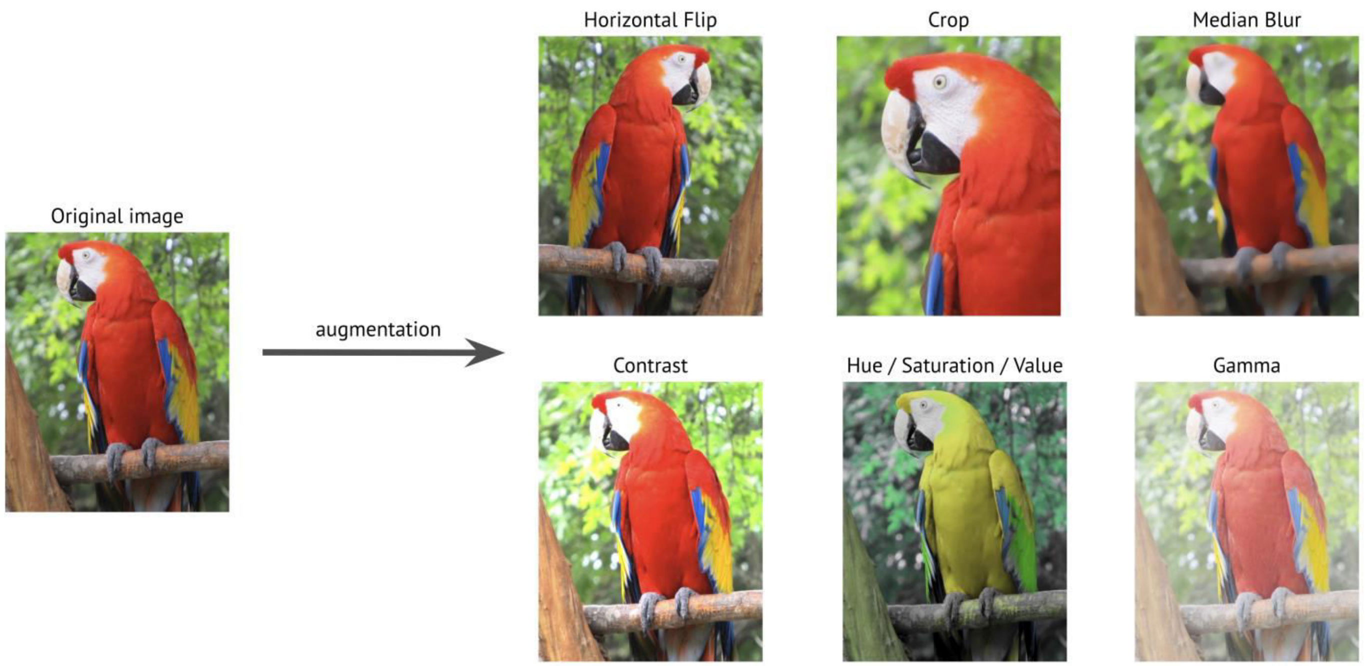
  - Forcing function to be smooth

$$L_{reg}(W) = E_x|\nabla f_W(x)|^2$$

# Augmentation

- Synthetically increase the number of training data
- Assume we know an operation T(x) which changes x but not the label
  - Examples for ImageNet: add noise, slight color change, cropping
- New objective:

$$L(W) = \sum_{i} \sum_{t \in T} \ell(f_W(t(x_i)), y_i)$$



# Supervised ML pipeline

- Input: $(x_1, y_1),(x_2, y_2)..(x_n, y_n)$
- Choose a hypothesis class (linear, tree, DNN) with parameters W
- Choose loss function e.g. cross entropy
- Choose regularization (e.g. L2) and augmentation (e.g. add noise)
- Train parameters with SGD+momentum

$$L(W) = -\sum_{(x_i, y_i)} log(f_W(x_i)[y_i]) \qquad w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t}$$

# General comment

- In this course, we will not put too much emphasis of the function f
- In most modern systems, it is a deep neural network
- In the future, it may be something else
- We will devote the next week to reviewiing DNN implementation
- DNNs not always the answer now, may not be popular in the future
- Keep an open mind

# Code practice

- https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html