Objectives of today's lecture
• review supervision of machine learning models
• review multi-task learning

Section 1. Training Data for Machine Learning

The revolution in AI is largely due to the availability of data. Consider ImageNet for a moment as well as more recent benchmarks in Natural Language Processing tasks. The core commonality of these benchmarks is that they make available labeled training datasets with (hundreds of) thousands of training examples.

We will next see why labeled training data is fundamental in machine learning (we will review supervision) and we will provide an overview of data collection in machine learning pipelines.

Section 1.1 Traditional Supervision Review

Standard supervised learning setup: We are given a training set of input-output \((x, y)\) pairs, the learning algorithm chooses a predictor \(h: X \to Y\) from a hypothesis class \(H\) (set of all “predictors” considered by the learning algorithm) and we evaluate it based on unseen test data.

Hypothesis: \(h\)  
Loss function: \(\ell\)

Training error: \(\ell(h)\) is an average of i.i.d. random variables; loss on each example \((x, y)\)

\[
\ell(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i), y_i)
\]

Testing error: \(L(h)\) is the expectation of the loss function

\[
L(h) = E(\ell(h(x), y)) = \int \ell(h(x), y) dP(x, y)
\]

We will use the training error (empirical risk) to find a hypothesis that minimizes the testing error (expected risk). Supervised learning is a minimization problem:

\[
\hat{h} = \arg\min_{h \in H} \ell(h)
\]

Depending on the task in hand (classification vs regression) we can use different loss functions. We present some examples below:

- **Squared Loss**: \((y_i - h(x_i))^2\)
- **Hinge Loss (SVM)**: \(\max(0, 1 - y \cdot h(x_i))\) \([\pm 1\ \text{targets}]\)
- **0-1 Loss**: \(\mathbf{1}(y_i \neq h(x_i))\)
- **Cross Entropy**: \(-y_i \cdot \log(h(x_i)) + (1 - y_i) \cdot \log(1 - h(x_i))\) \([0,1\ \text{targets}]\)
Section 1.2 Data Collection Overview

Please read the overview in the Survey Paper of Roh et al., 2019

Figure 1. An overview of data collection problems in machine learning

Figure 2. The data collection flow chart.

Three core data collection tasks

→ Data acquisition

→ Data labeling

→ Data improvement

We focus on data labeling:

- **Tedious**: repetitive
- **Expensive**: human hours required to label looks data points
- **One-off**: traditional labeling performed once, effort cannot be automatically applied to new data points
Section 2. Noisy Data Labeling

We will cover three state-of-the-art methods for scaling data labeling (the goal is to reduce the time cost associated with data labeling):

- we can scale to multiple labelers (using crowdsourcing/expert-sourcing); labels can be noisy and have conflicts.
- we can convert human intuition and knowledge regarding the class or target of an example to programatic functions that can be applied to large collections of input, unlabeled data; rules can be noisy and correlated.
- In many cases we do not have explicit labels but we want the model to capture structure and dependencies in the data generating distribution. We can use the context available in the raw data to enable self-supervised learning.

Section 2.1 Fusing Noisy Data (from the Crowd)

The generative model behind the labels generated by independent labelers for a data point; we consider items with binary labels.

Let \( Y^* \) be the true (hidden/unknown) label of the data point. Let \( Y^* \) take values in \{-1, 1\}. Consider \( K \) labelers and let \( X_{k} \) with \( k = 1, 2, ..., K \) be the labels assigned to the data point by the \( K \) labelers. We only observe labels \( X_{k} \) and need to fuse them into a single label \( \hat{Y} \) for the data point.

We consider the following generative process for generating labels \( X_{k} \)

Step 1. Sample \( Y^* \sim P(Y) \)

Step 2. Iterate over each labeler:

- Given \( Y^* \), for labeler \( k \) flip a coin following a Bernoulli distribution with parameter \( p_{k} \) (we will refer to \( p_{k} \) as the accuracy of labeler \( k \))
- If the coin returns 1 then set \( X_{k} = Y^* \) else set \( X_{k} = -Y^* \)

Goal: Assuming that the labelers are independent and that we only know the values for labels \( X_{k} \) generated from the process above how can we find the unknown value \( Y^* \)?

Section 2.1.1 Majority Vote (All labelers have the same accuracy \( p \))

Majority Vote (MV) decides for type \( t \) if more than one half of the ratings are in favor of \( t \) (can be extended to plurality vote when we consider categorical and not binary types; here we focus on binary types)

\[
\hat{Y} = \begin{cases} 
1 & \text{if } \sum_{k} 1(X_{k} = t) \geq \left\lfloor \frac{1}{2} K \right\rfloor + 1 \\
-1 & \text{otherwise}
\end{cases}
\]

What is the probability that \( \hat{Y} \) is correct?

Under the assumption that all labelers have the same accuracy \( p \), we can use the Binomial distribution to model the probability that more than \( k/2 \) labelers give us the correct label. We have:

\[
P(\hat{Y} = Y^*) = \sum_{k=\left\lceil \frac{1}{2} K \right\rceil + 1}^{K} \binom{K}{k} p^{k} (1-p)^{K-k}
\]

Increases monotonically in \( K \) (goes to 1), decreases monotonically in \( K \) (goes to 0).

Proposition: Under the assumption that all labelers have the same accuracy, given that \( K \) is odd and given that \( p > 0.5 \) and \( 1 - p < P(t) < p \) for both \( t \in \{-1, 1\} \), then MV is an optimal decision rule.

Try to prove it at home!
Section 2.1.2 Weighted Majority Vote and The Maximum A Posteriori Label (Labelers are independent but have different accuracy values)

Different labelers have different accuracies. But remember that each accuracy is a parameter of a Bernoulli distribution that characterizes the assigned labels. We need to solve an inference problem over a simple Bayesian model. We set \( \hat{y} = \arg \max_t P(Y^* = t | X_1, X_2, \ldots, X_k) \) [this is a simple MAP problem].

This is equivalent to \( \hat{y} = \text{sign} \left( \log \frac{P(Y^* = t | X_1, X_2, \ldots, X_k)}{P(Y^* = -t | X_1, X_2, \ldots, X_k)} \right) \). We have for \( P(Y^* = t | X_1, X_2, \ldots, X_k) \):

\[
P(Y^* = t | X_1, X_2, \ldots, X_k) = \frac{p(X_1, X_2, \ldots, X_k | Y^* = t) \cdot P(t)}{P(X_1, X_2, \ldots, X_k)}
\]

but we considered independent labelers, hence we have that \( P(X_1, X_2, \ldots, X_k | Y^* = t) = \prod_{k=1}^{K} P(X_k | Y^* = t) \). The log-odds becomes

\[
\log \frac{P(Y^* = t | X_1, X_2, \ldots, X_k)}{P(Y^* = -t | X_1, X_2, \ldots, X_k)} = \log \frac{P(t)}{P(t)} + \sum_{k=1}^{K} \log \frac{P(X_k | Y^* = 1)}{P(X_k | Y^* = -1)}
\]

Notice that if \( X_k = 1 \) then \( P(X_k = 1 | Y^* = 1) = P_k \) (the accuracy of labeler \( k \))
otherwise if \( X_k = -1 \) then \( P(X_k = -1 | Y^* = 1) = 1 - P_k \).

How can we learn the accuracy of each labeler?

Section 2.1.3 Learning the Labeler Accuracies

**Approach 1: Expectation Maximization**

We will use an iterative algorithm: the Dawid-Skene Algorithm (from 1979).

1. **Step 1.** Initialize the accuracy of each labeler to a value \( > 0.5 \)
   (e.g. set it to 0.5).
2. **Step 2.** Estimate the MAP value for each \( \hat{y} \) (using the above expression).
3. **Step 3.** Estimate the empirical accuracy for each labeler, update \( P_k \).
4. **Step 4.** Go to Step 2, and iterate until convergence.

**Approach 2:** Let's use stochastic gradient descend.

See attached notes by Chen, Sala, and Re' (Section 3.1)

Section 2.2 Generating Labeled Data Programmatically (Data Programming)

Figure 3: The typical programmatic labeling pipeline. Programs and heuristics are used as labelers (similar to the crowdsourcing setting).

Section 2.3 Self-Supervised Learning

The goal of self-supervised learning: Learn a “good” representation of the data in an unsupervised manner and then fine-tune it to different downstream tasks (using minimal labeled examples).

Typical loss:

\[ L_{un}(f) := \mathbb{E}_{x,x^{+},x^{-}} \left[ -\log \left( \frac{e^{f(x)^T f(x^+)}}{e^{f(x)^T f(x^+)} + e^{f(x)^T f(x^-)}} \right) \right] \]

(1)

We rely on semantic context (similarities) to define \( x^+ \) and \( x^- \) given \( x \)

Simple Lemma: The average classification loss on downstream binary tasks is upper bounded by the unsupervised loss.

\[ L_{reg}(f) \leq \alpha L_{un}(f), \forall f \in \mathcal{F} \]

(2) where \( \alpha \) depends on \( p, (\alpha \to 1 \text{ when } |\mathcal{C}| \to \infty , \text{for uniform } p) \)

Link: http://www.offconvex.org/2019/03/19/CURL/
Section 3. Multi-task Learning

Example of related tasks:

![Diagram of related tasks]

Section 3.1 Two Forms of Multi-Task Learning

Soft-parameter sharing: regularization term in the loss so that weights of different network components “align”.

![Diagram of hard-parameter sharing](Figure 1: Hard parameter sharing for multi-task learning in deep neural networks)

![Diagram of soft-parameter sharing](Figure 2: Soft parameter sharing for multi-task learning in deep neural networks)

Section 3.2 Why does Multi-Task Learning work?

- Implicit increase of the training data for each network.
- Representation bias: representations that perform well in multiple tasks are learned (less overfitting)