Lecture 1

Supervised Learning

Examples are annotated with labels.

Classification

Our model will provide labels from a finite set of classes/categories.

CISC684 Notation

- **Datasets**

  $$D = \{\langle x^i, y^i \rangle \mid 1 \leq i \leq N\}$$

  D is our dataset, where we have N training instances, and x^i is a training instance, and y^i is a target value.

- **Classification**

  $$y^i \in C = \{c_1, \ldots, c_k\}$$

  Where y^i is the tartget value for x^i. When $k = 2$, we say it is binary classification, and for $k > 2$, we say it is multi-class classifcation.

- **Regression**

  We allow y^i to be a real-valued number in regression.

Input Instances

Typically our input are specific values for d attributes or features. These values can be discrete/categorical or numerical. Our input may also be images, or texts, emails, etc.

Each of our input instances can be represented as vectors of dimension d.

Lecture 2

Inductive Bias

Strong bias may result in an inability to fit data. The term *high bias* is often used to indicate the inability to fit training data.

High bias leads to simple models. We expect simple models to not change much with small changes to training data (*low variance*).

Noise

What is noise?

- Errors in labeling data - *teacher noise*
- Imprecision in recording the input attributes (shifted data points).

  $\langle \ldots, 98.4, \ldots \rangle$ instead of $\langle \ldots, 100.4, \ldots \rangle$
- Factors or attributes that are not considered in input representation. (*latent* or *hidden* attributes, which are unobserved).

Overfitting

A stringently trained model may struggle to produce accurate results given noisy data.

Evaluation

We may split our dataset into a training set and a test set. The training set being larger, and the test set being statistically meaningful.

The typical split is 90:10, or 80:20.

During testing time, we predict y' given x, and check if $y' = y$.

n-fold Cross-Validation

Motive: we may not have a large amount of annotated data.

In this case, we partition our data into n partitions, and use $n-1$ of these partitions for training, and one partition for testing. Then we repeat n times.

The common practice is 10-fold cross-validation with a training/test split of 90:10.

Learning as Searching

Each machine learning method defines a hypothesis space, that is, a certain class of functions to delineate our data. During training, we search for a "hypothesis" from amongst these functions which optimizes our model.

Loss Function

Loss functions characterize problems a chosen model has in fitting the data. Often times, loss functions are tied to *error* made by the model on a certain data set.

The goal of training is to *minimize* the loss function.

Overfitting with Training

With extensive training, overfitting may occur. We may observe this directly if perhaps validation loss increases as training loss decreases.

Linear Regression

House Prices: Can we predice the price of a house based on

- a_1: Floor area
- a_2: Number of rooms
- a_3: Number of bathrooms
- a_4: Availability of a swimming pool

We assume that house prices may be modeled by some linear function, then we may learn it.

Let $x^i = \langle a_0, a_1, a_2, a_3, a_4 \rangle$, where $a_0 = 1$.

Then our training set $D = \{\langle x^1, y^1 \rangle \ldots, \langle x^n, y^n \rangle\}$.

Let $f(x) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + w_0$.

Then our model is specified by $w = \langle w_0, w_1, w_2, w_3, w_4 \rangle$.

And our output vector $O = w \cdot x$

Let $E(w)$ be the errors on the training set by w.

For a training instance, x^t,

$$\begin{aligned} e^t(w) &= y^t - O^t \\ &= y^t - w \cdot x^t \end{aligned}$$

Then $E(w)$ is simply

$$\sum_{i=1}^{n} (y^t - w \cdot x^t)^2$$

Lecture 3

Gradient Descent

$$\Delta(w') = -\eta \frac{\nabla E(w')}{\nabla w'}$$

where $0 < \eta < 1$ is the learning rate.

$$\rightarrow W = w + \Delta w$$

We use

$$W_i = W_i + \Delta w_i$$
$$W_i = -\eta \frac{\partial E(w)}{\partial w_i}$$

$$\begin{aligned}
\frac{\partial E(w)}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_t (y^t - O^t)^2 \\
&= \frac{1}{2} \sum_t 2 \cdot (y^2 - O^t) \cdot \frac{\partial}{\partial w_i} (y^2 - O^t) \\
&= \sum_t (y^t - O^t)(-x_i^t)
\end{aligned}$$

$$\rightarrow \Delta w_i = \eta \sum_t (y^t - O^t) x_i^t$$

Batching: Computes gradient descent for partitions or batches of the dataset.

Stochastic Gradient Descent: Makes updates based on batch size = 1.

Gradient Descent Algorithm

- Start with a random set of small weights (w_i)
- While loss is not acceptable:
  - Compute loss and gradients
  - Apply gradients
  - Recompute model

Batching vs. Stochastic

Stochastic Gradient Descent can approximate batching with an arbitrary small error.

SGM can also better avoid local minimas by updating w more frequently. It is also faster to train, deals with dynamic datasets, and learning rate is usually smaller in SGM.

Batch training is typically more stable, and easier to check for convergence.

Lecture 4

Logistic Regression

$$\mathrm{P}(Y = 1 \,|\, \hat{x}, \hat{w})$$

The probability that $\hat{x}$ is a positive instance.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

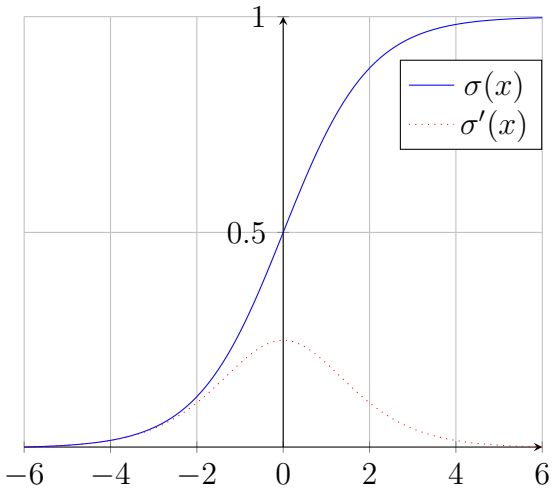
The sigmoid function will squash our linear output $\rightarrow [0, 1]$

$$\text{Output} = \begin{cases} 1 & P(y = 1 \,|\, \hat{x}, \hat{w}) = \sigma(\hat{w} \cdot \hat{x}) \geq 0.5 \\ 0 & \text{else} \end{cases}$$

Maximum Likelihood Estimation: MLE finds the parameter values that maximize the likelihood of making the observations given the parameters.

For a model θ, what is the likelihood of the observed data (training data).

Figure 1: Sigmoid Function



Given n training instances given by $x^1, x^2, \ldots, x^n$, a model given by parameters w, and labels $y^1, y^2, \ldots, y^n$, the likelihood of the training data is given by

$$\mathrm{P}(y^1, y^2, \ldots, y^n \,|\, w, x^2, x^2, \ldots, x^n)$$

These instances are independent, thus

$$\begin{aligned}\mathrm{P}(y^1, y^2, \ldots, y^n \,|\, w, x^1, x^2, \ldots, x^n) &= \mathrm{P}(y^1 \,|\, w, x^1) \cdot \ldots \cdot \mathrm{P}(y^n \,|\, w, x^n) \\ &= \prod_{0 \le i \le n} \mathrm{P}(y^i \,|w, x^i)\end{aligned}$$

$\prod_t \mathrm{P}(y^t \,|\, w, x^t)$ can be extremely small, leading to underflow problems, so we take the log and maximize.

$$\sum_t \log(\mathrm{P}(y^t \,|\, w, x^t))$$

Since y^t is either 0 or 1, we can write this as

$$\sum_t y^t \log(\mathrm{P}(y = 1 \,|\, w, x^t)) + (1 - y^t) \log(\mathrm{P}(y = 0 \,|\, w, x^t))$$

We use gradient descent to find the w that maximizes this.

Wald Statistic

The **Wald test** is a method for testing whether **explanatory variables** in a model is **significant**.

"Significant" indicating that the variable adds to the model. Variables which do not add anything to the model can be deleted without affecting the model in any meaningful way.

Logits

Rather than converting regression to probability, in statistics, a link function maps probability to regression.

Logit is the log of oddds.

- $F(x)$ is the probability of success.
- Probability of failure is $1 - F(x)$.
- Odds are defined as $\frac{F(x)}{1-F(x)}$

$F(x) \to [0, 1]$, odds thus range between 0 and ∞. As $p \to 1$, odds tend towards infinity.

$$\lim_{p \to 1} \log(\frac{p}{1-p}) = \infty, \quad \lim_{p \to 0} \log(\frac{p}{1-p}) = -\infty$$

Lecture 5

Decision Trees

Figure 2: Example Decision Tree

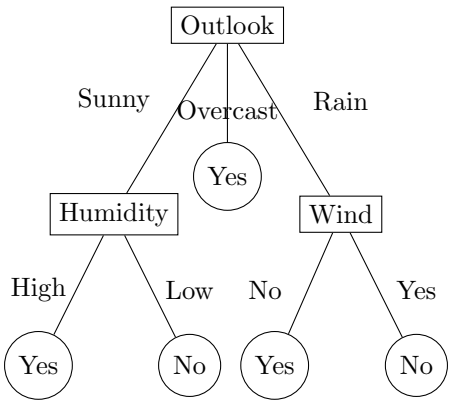


Table 1: Example Dataset

A	B	Target
a_1	b_1	0
a_2	b_2	1
a_1	b_2	1
a_2	b_1	0

We begin building a decision tree by considering an attribute A, B, etc. We consider the values which A may take, a_1, a_2. The leaf nodes of our decision tree **must be decisions**.

If we cannot make a decision based on the attributes we have considered, we consider a new attribute.

Uncertainty

Suppose we have attributes A_1, A_2. A_1 has three values:

c_1	4+/2-
c_2	2+/2-
c_3	2+/4-

and A_2 has values:

d_1	5+5-
d_2	2+/1-
d_3	1+/2-

Which attribute do we prefer to begin with? **The attribute which we observe has less uncertainty.** A_2 has a concentration of outcomes on d_1 which are evenly stacked, leading to the overall attribute holding significant uncertainty.

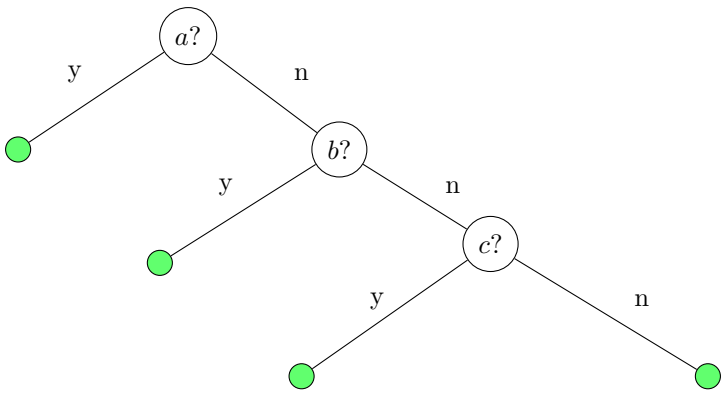
Entropy

Let x be a random variable with outcomes $\{a, b, c, d\}$. How many "yes-no" questions do you need to ask to know the outcome of x?

Suppose

$$P(x = a) = \frac{1}{2}, P(x = b) = \frac{1}{4}, P(x = c) = P(x = d) = \frac{1}{8}$$

and we form the following tree:



The expected number of questions we ask is

$$\frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3 = 1.75$$

We want to minimize our expected number with an optimal line of inquiry. We can do this by **balancing our tree in terms of probability**

i	P_i	i	$f(P_i)$
1	$\frac{1}{2}$	1	$\log_2 \frac{1}{P_i}$
2	$\frac{1}{4}$	2	$\vdots$
3	$\frac{1}{8}$	3	$\vdots$
4	$\frac{1}{8}$	3	$\vdots$

We define entropy,

$$\begin{aligned} H(x) &= \sum_i P_i \log_2 \frac{1}{P_i} \\ &= -\sum_i P_i \log_2 \frac{1}{P_i}, \end{aligned}$$

$$0 \leq H(x) \leq \log_2(N),$$

as our measure of uncertainty.

(Joint entropy, conditional entropy in lecture slides)

Mutual Information