Machine Learning Concepts

Professor Marie Roch
San Diego State University

Basic definitions & concepts

• Task – What the system is supposed to do
e.g. ASR: \( f(\text{input speech}) \rightarrow \text{list of words} \)
• Performance measure – How well does it work?
• Experience – How does the machine learn the task?
Types of experiences; how a learner learns...

- Supervised learning – Learn class conditional distributions: $P(\omega | x)$ implies class labels are known

- Unsupervised learning – No labels are provided, learn $P(x)$ and possibly group x’s into clusters.

Types of experiences

- Reinforcement learning – Learner actions are associated with payouts for actions in environment.
Experience data set:
Design matrix

Feature vector

\[
\begin{pmatrix}
X_{1,1} & X_{1,2} & X_{1,3} & \cdots & X_{1,D} \\
X_{2,1} & \cdots & y_2 \\
X_{3,1} & \cdots & y_3 \\
\vdots & \vdots & \vdots \\
X_{N,1} & X_{N,2} & X_{N,3} & \cdots & X_{N,D} & y_N
\end{pmatrix}
\]

Feature sets may be used if some features may be missing.

A Cardinal Rule
OF MACHINE LEARNING

THOU SHALT
NOT TEST ON
THY TRAINING
DATA
Performance

• A metric that measures how well a learner is able to accomplish the task
• Metrics can vary significantly (more on these later), examples:
  – loss functions such as squared error
  – cross entropy

Partitioning data

• Training data – experience for learner
• Test data – performance measurement
• Evaluation data
  – Only used once all adjustments are made
  – It is common to:
    • Adjustments are a form of training (see 5.3)
    • Evaluation provides an independent test
Regression

Given a set of features and response, predict a response

Linear regression
A simple learning algorithm

Predict response from data

\[ \hat{y} = w^T x \quad w, x \in \mathbb{R}^N, \hat{y} \in \mathbb{R} \]

- \( w \) is the weight vector
- Goal: Maximize performance on test set.

Learn \( w \) to minimize some criterion

E.g. mean squared error (MSE)

\[ \text{MSE}_\text{test} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{m} \| \hat{y}^\text{test} - y^\text{test} \|_2^2 \]

\( \| \cdot \|_2 \) denotes the \( L_2 \) norm \( \sum_{i=1}^{N} | \cdot |^2 \) read Goodfellow et al. 2.5
Linear regression

- Cannot estimate \( w \) from test data
- Use the training data
- Minimize

\[
\text{MSE}_{\text{train}} = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{m} \left\| \hat{y}_{\text{train}} - y_{\text{train}} \right\|_2^2
\]

For convenience, we will usually omit the variable descriptor train when describing training.

Linear regression

MSE minimized when gradient is zero

\[\nabla_w \text{MSE} = 0\]

\[\nabla_w \frac{1}{m} \left\| \hat{y} - y \right\|_2^2 = 0\]

\[\nabla_w \left\| \hat{y} - y \right\|_2^2 = 0\]

\[\nabla_w \left\| Xw - y \right\|_2^2 = 0 \text{ as } Xw = \hat{y}\]
Linear regression

\( \nabla_w \|Xw - y\|_2^2 = 0 \)
\( \nabla_w (Xw - y)^\top (Xw - y) = 0 \) \( L_2^2 \) norm in matrix notation
\( \nabla_w \left( (Xw)^\top - y^\top \right)(Xw - y) = 0 \) transpose distributive over addition
\( \nabla_w \left( w^\top X^\top - y^\top \right)(Xw - y) = 0 \) as \((AB)^\top = B^\top A^\top\) Goodfellow et al. eqn 2.9
\( \nabla_w \left( w^\top X^\top Xw - w^\top X^\top y - y^\top Xw + y^\top y \right) = 0 \)
\( \nabla_w \left( w^\top X^\top Xw - y^\top Xw - y^\top Xw + y^\top y \right) = 0 \) as \( w^\top X^\top y = y^\top \left( w^\top X^\top \right)^\top = y^\top Xw \)
\( \nabla_w \left( w^\top X^\top Xw - 2y^\top Xw + y^\top y \right) = 0 \)

These are referred to as the normal equations
Linear regression

Regression formula forces line through origin
Remove restriction:
• Add bias term $\hat{y} = w^T x + b$
• To use normal equations, use modified $x$
• Last term of new weight vector $w$ is bias
Notes on learning

- A learner that performs well on unseen data is aid to generalize well.
- When will learning on training data produce good generalization?
  - Training and test data drawn from the same distribution
  - Large enough training set to learn the distribution

Underfitting & Overfitting

- Underfit
  Model cannot learn training data well

- Overfit
  Model does not generalize well

- These properties are related to model capacity
Capacity

What kind of functions can we learn?

Underfitting
\[ \hat{y} = w_0 x_0^1 + w_1 x_0^0 \]

Appropriate capacity
\[ \hat{y} = w_0 x_0^2 + w_1 x_0^1 + w_2 x_0^0 \]

Overfitting
\[ \hat{y} = \sum_{i=0}^{D} w_i x_0^{D-i} \]

Shattering points

Points are \textit{shattered} if a classifier can to separate them regardless of their binary label.

We can shatter the three points, but not four with a linear classifier.
Representational capacity

• Best function that can be learned within a set of learnable functions

• Frequently a difficult optimization problem
  – We might learn a suboptimal solution
  – This is called *effective capacity*

Measuring capacity

• Model order is not always a good predictor of capacity

Label determined by sign of function. Increasing frequency of sinusoid enables ever finer distinctions…
Measuring capacity

• Vapnik Chervonikis (VC) dimension
  – for binary classifiers
  – Largest # of points that can be shattered by a family of classifier.

• In practice, hard to estimate for deep learners… so why do we care?

Predictions about capacity

• Goal: Minimize the generalization error
• Hard; perhaps minimize difference $\Delta$
  $\Delta = \text{training error} - \text{generalization error}$.
• Learning theory
  – Models with higher capacity have higher upper bound on $\Delta$
  – Increasing amount of training data decreases $\Delta$’s upper bound
Recap on capacity

The NO FREE LUNCH Theorem

Expected performance of any classifier across all possible generalization tasks is no better than any other classifier.

A classifier might be better for some tasks, but no classifier is universally better than others.
N-fold cross validation

• Problem:
  – More data yields better training
  – Getting more data can be expensive

• Workaround
  – Partition data into N different groups
  – Train on N-1 groups, test on last group
  – Rotate to have N different evaluations

Regularization

• Remember: Learners select a solution function from a set of hypothesis functions.
• Optimization picks the function that minimizes some optimization criterion
• Regularization lets us express a preference for certain types of solutions
Example:
High Dimensional Polynomial Fit

- Suppose we want small coefficients
  Remember: \( \hat{y} = w^T x \)
- This happens when \( \Omega(w) = w^T w \) is small and results in shallower slopes
- We can define a new criterion:
  \[
  J(w) = MSE + \lambda \Omega(w) = MSE + \lambda w^T w
  \]
  where \( \lambda \) controls regularization influence

9\textsuperscript{th} degree polynomial fit with regularization
Point estimators

- An approximation of interest
- Examples:
  - a statistic of a distribution, e.g.
    \[ \hat{\mu} = \frac{1}{N} \sum_{i} x^{(i)} \] is an approximation of \( \mu \)
  - a parameter of a classifier

In general, it is a function of data

\[ \hat{\Theta}_m = f(x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}) \]

and may even be a classifier function that maps data to a label (function estimation).
Bias

How far from the true value is our estimator?

\[ \text{bias}(\hat{\theta}_m) = E[\hat{\theta}_m] - \theta \]

Goodfellow et al. give an example with a Bernoulli distribution that we have not yet covered (read 3.9.1). Bernoulli distributions are good for estimating the number of times that a binary event occurs (e.g. 42 head in 100 coin tosses).

Bias of sample mean

\[
\text{bias}(\hat{\mu}_m) = E[\hat{\mu}_m] - \mu \\
= E\left[ \frac{1}{N} \sum_i x^{(i)} \right] - \mu \\
= \frac{1}{N} E\left[ \sum_i x^{(i)} \right] - \mu \\
= \frac{1}{N} N \cdot E[X] - \mu \quad x^{(i)} \text{ is a random var} \\
= \mu - \mu = 0 \quad \text{unbiased estimator}
\]
Bias

- Read more examples in Goodfellow et al.
- Bias of classifier functions?
  - We are trying to estimate the Bayes classifier.
  - Bias is amount of error over that

Variance

- Already defined: \( \text{Var}(X) = E[(X - \mu)^2] \)
- Variance of classifier functions
  - Variance of the mean
    \[
    \text{Var} \left( \frac{1}{m} (X_1 + X_2 + \ldots + X_m) \right) = \frac{1}{m} \text{Var}(X_1 + X_2 + \ldots + X_m) = \frac{m}{X \text{Var}(X_1)} = k \text{Var}(X)
    \]
  - \( \frac{1}{m} m \sigma^2 = \frac{1}{m} \sigma^2 \) or equivalently, standard error \( \text{SE}(\hat{\mu}_m) = \frac{\sigma}{\sqrt{m}} \)
Bias & Variance

- Variance gives us an idea of classifier sensitivity to different data
- Together can estimate with 95% confidence that the real mean lies within:

\[
\hat{\mu}_m - 1.96SE(\hat{\mu}_m) \leq \mu_m \leq \hat{\mu}_m + 1.96SE(\hat{\mu}_m)
\]

Information Theory

A quick trip down the rabbit hole..
- Details in Goodfellow 3.13
- Needed for maximum likelihood estimators
Quantity of information

- Amount of surprise that one sees when observing an event.

\[ I(x_i) = \log \frac{1}{P(x_i)} \]

- If an event is rare, we can derive a large quantity of information from it.

Quantity of information

- Why use log?
  - Suppose we want to know the information in two independent events:

\[
I(x_1, x_2) = \log \frac{1}{P(x_1, x_2)}
\]

\[
= \log \frac{1}{P(x_1)P(x_2)} \quad x_1, x_2 \text{ independent}
\]

\[
= \log \frac{1}{P(x_1)} + \log \frac{1}{P(x_2)}
\]

\[
= I(x_1) + I(x_2)
\]
Entropy

• Entropy is defined as the expected amount of information (average amount of surprise) and is usually denoted by the symbol \( H \).

\[
H(X) = E[I(X)] \\
= \sum_{x_i \in S} P(x_i)I(x_i) \\
= \sum_{x_i \in S} P(x_i) \log \frac{1}{P(x_i)} \quad \text{definition } I(x_i) \\
= E[-\log P(X)]
\]

Discrete vs continuous

**Discrete**

- Shannon Entropy
- Use \( \log_2 \)
- Units called
  - bits, or sometimes
  - Shannons

**Continuous**

- Differential entropy
- Use \( \log_e \)
- Units called nats
Example

- Assume
  - $X = \{0, 1\}$
  - $P(X) = \begin{cases} p & X = 0 \\ 1 - p & X = 1 \end{cases}$

- Then
  - $H(X) = E[I(X)]$
    - $= -p \log p - (1 - p) \log(1 - p)$

Comparing distributions

- How similar are distributions $P$ and $Q$?
- The Kullback-Leibler (KL) divergence tells us:
  - How many bits/nats extra are needed to represent $P$’s distribution using $Q$?
  - $D_{KL}(P \parallel Q) = E_{X \sim P} \left[ \log \frac{P_P(X)}{P_Q(X)} \right] = E_{X \sim P} \left[ \log P_P(X) - \log P_Q(X) \right]$
    - $D_{KL}(P \parallel Q) = 0$ iff $P = Q$, otherwise $D_{KL}(P \parallel Q) > 0$

  note: $D_{KL}(P \parallel Q) \neq D_{KL}(Q \parallel P)$, not a distance measure
Cross entropy

• Related to KL divergence
  \[ H(P, Q) = E_{X \sim P} \left[ - \log P_Q(X) \right] = H(P) + D_{KL}(P \parallel Q) \]

• Interesting as we sometimes want to minimize KL divergence…
Maximum Likelihood Estimation (MLE)

• Method to estimate model parameters
• Suppose we have \( m \) independent samples drawn from a distribution
• Can we fit a model with parameters \( \Theta \)?

\[
\mathcal{X} = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}
\]

\[
\theta_{ML} = \arg \max_{\theta} P(\mathcal{X} | \theta)
\]

MLE

• The \( x^{(i)} \)'s are independent, so

\[
\theta_{ML} = \arg \max_{\theta} \prod_{i} P(x^{(i)} | \theta) = \arg \max_{\theta} \prod_{i} P(x^{(i)} | \theta)
\]

• Transform to something easier…

\[
\theta_{ML} = \arg \max_{\theta} \prod_{i} P(x^{(i)} | \theta) = \arg \max_{\theta} \sum_{i} \log P(x^{(i)} | \theta)
\]

• Traditional to use derivatives and solve for the maximum value.
MLE through optimization

• Let’s reframe this problem:

\[ \theta_{ML} = \arg \max_{\theta} \sum_i \log P_{model}(x^i \mid \theta) \]

\[ = \arg \max_{\theta} E_{x \sim \hat{P}_{data}} [\log P_{model}(x \mid \theta)] \]

• Maximized when \( P_{model} \) is most like \( \hat{P}_{data} \)

• What does this remind us of?

MLE through optimization

• \( D_{KL}(\hat{P}_{data} \mid\mid P_{model}) \) minimized as \( P_{model} \)
becomes more like \( \hat{P}_{data} \)

• Recall

\[ D_{KL}(\hat{P}_{data} \mid\mid P_{model}) = E_{x \sim \hat{P}_{data}} \left[ \log \hat{P}_{data}(X) - \log P_{model}(X) \right] \]

• and we can minimize one term of cross entropy

\[ -E_{x \sim \hat{P}_{data}} [\log P_{model}(X)] \]
Linear regression as MLE

• Instead of thinking of predicting value \( \hat{y} = w_x \), what if we predicted a conditional probability?
  \[ P(\hat{y} \mid x) \]

• Regression: only one possible output.

• MLE estimates a distribution: support for multiple outcomes, can think of this as a noisy prediction.

Linear regression as MLE

• Assume \( Y \mid X \sim n(\mu, \sigma^2) \)
  – learn \( \mu \) such that \( E[Y \mid x^{(i)}] = y^{(i)} \)
  – \( \sigma^2 \) fixed (noise)

• Can formulate as an MLE problem
  \[ \theta_{\text{ML}} = \arg \max_{\theta} P(Y \mid X; \theta) \]

where parameter \( \Theta \) is our weights \( w \).
Linear regression as MLE

\[ \theta_{\text{ML}} = \arg \max_{\theta} P(Y \mid X; \theta) \]
\[ = \arg \max_{\theta} \prod_{i} P(y^{(i)} \mid x^{(i)}; \theta) \quad \text{if } x^{(i)} \text{'s independent & ~ identically} \]
\[ \log \theta_{\text{ML}} = \arg \max_{\theta} \sum_{i} \log P(y^{(i)} \mid x^{(i)}; \theta) \]
\[ = \arg \max_{\theta} \sum_{i} \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y^{(i)} - \mu)^2}{2\sigma^2}} \quad \text{prediction of } y^{(i)} \text{ is } \hat{y}^{(i)} \]
\[ = \arg \max_{\theta} \sum_{i} \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y^{(i)} - \mu)^2}{2\sigma^2}} \quad \text{as we want } E[Y \mid x^{(i)}] = y^{(i)} \]

Linear regression as MLE

\[ \log \theta_{\text{ML}} = \arg \max_{\theta} \sum_{i} \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\hat{y}^{(i)} - y^{(i)})^2}{2\sigma^2}} \]
\[ = \arg \max_{\theta} \sum_{i} \left( -\frac{1}{2} \log(2\pi) - \log \sigma - \frac{(\hat{y}^{(i)} - y^{(i)})^2}{2\sigma^2} \right) \]
\[ = \arg \max_{\theta} \left( -\frac{m}{2} \log(2\pi) - m \log \sigma - \sum_{i} \frac{(\hat{y}^{(i)} - y^{(i)})^2}{2\sigma^2} \right) \]

Equivalent to maximizing \( \sum (\hat{y}^{(i)} - y^{(i)})^2 \) which has the same form as or MSE optimization:

\[ MSE_{\text{train}} = \frac{1}{m} \left\| \hat{y}_{\text{train}} - y_{\text{train}} \right\|_2^2 \]
MLE

• MLE can only recover the distribution if the parametric distribution is appropriate for the data.
• If data drawn from multiple distributions with varying parameters, MLE can still estimate distribution, but information about underlying distributions is lost.

Optimization

• We have seen closed form (single equation) optimization with linear regression.
• Not always so lucky… how do we optimize more complicated things?
Optimization

- Select an *objective* function $f(x)$ to optimize e.g. MSE, KL divergence

- Without loss of generality, we will always consider minimization.

  *Why can we do this?*

---

**Gradient descent**

- Global minimum at $x = 0$. Since $f'(x) = 0$, gradient descent halts here.
- For $x < 0$, we have $f'(x) < 0$, so we can decrease $f$ by moving rightward.
- For $x > 0$, we have $f'(x) > 0$, so we can decrease $f$ by moving leftward.
Critical points

- Minimum
- Maximum
- Saddle point

Global vs. local minima

- This local minimum performs nearly as well as the global one, so it is an acceptable halting point.
- This local minimum performs poorly and should be avoided.

Ideally, we would like to arrive at the global minimum, but this might not be possible.
Functions on $\mathbb{R}^m \to \mathbb{R}^1$

- Gradient vector of partial derivatives
  \[ \nabla f(x) = \begin{bmatrix} \frac{d}{dx_1} f_i(x) \\ \frac{d}{dx_2} f_i(x) \\ \vdots \\ \frac{d}{dx_n} f_i(x) \end{bmatrix} \]
- Moving in gradient direction will increase $f(x)$ if we don’t go too far…
- To move to an $x$ with a smaller $f(x)$
  \[ x' = x - \epsilon \nabla f(x) \]
  what we pick for $\epsilon$ will make a difference

IMPORTANT: $f$ is fn to be optimized, e.g. MSE, not learner

Putting this all together

- Want to learn: $f_\theta(x_i) = y_i$
- To improve $f_\theta(x_i)$ we define an objective fn $J(\Theta)$
- Optimize $J(\Theta)$
  - Gradients defined for each sample
  - Average over data set and update $\Theta$
Putting this all together

- Sample $J(\theta)$, a loss function:

$$J(\theta) = E_{x,y \sim p_{data}} [L(x,y,\theta)] = \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, y^{(i)}; \theta)$$

where $L(x, y, \theta) = -\log P(y | x, \theta)$  
remember $D_{KL}$?

or $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} -\log P(y | x, \theta)$

- which is like our effort to get a MLE by minimizing KL divergence

"There’s the rub"

Shakespeare’s Hamlet

- Computing $J(\Theta)$ is expensive
  One example not so bad…
  massive data sets…

- Sample expectation relies on having enough samples that the $1/m$ term estimates $P(X)$.
- What if we only evaluated some of them…
Stochastic gradient descent (SGD)

while not converged:
    pick minibatch of samples (x,y)
    compute gradient
    update estimate

minibatch is usually up to a 100 samples