Machine Learning 2.03: Incremental Learning

Tom S. F. Haines
T.S.F.Haines@bath.ac.uk

UNIVERSITY OF BATH

Traditionally. . .

- Collect data set.
- Train model.
- Test model.
- Start using model.

- Takes days, weeks, months.
- Rarely updated.

Incremental learning

Incremental Learning: **Update model as data arrives**
(Also called online learning)

UNIVERSITY OF BATH

Incremental learning

Incremental Learning: **Update model as data arrives**
(Also called online learning)

- Traditional approach too slow for some problems.
  - High frequency trading
  - CCTV
  - Intrusion detection
- Data set too large to fit in RAM.

Engineering

- Incremental algorithms:
  - Hard to code (expensive).
  - Lower performance (usually, though not always).
- Avoid if possible!

Avoiding

- Batch approach (traditional):
  - Keep data.
  - Train new model regularly (e.g. overnight).
  - Disadvantage: Always out of date.
  - Disadvantage: Storage and training time increase with data...

UNIVERSITY OF BATH

Avoiding

- Batch approach (traditional):
  - Keep data.
  - Train new model regularly (e.g. overnight).
  - Disadvantage: Always out of date.
  - Disadvantage: Storage and training time increase with data...
- Batch + Correction:
  - Incrementally train model to correct from last batch.
  - Advantage: Complex batch, simple incremental.
  - Disadvantage: Loss of performance.
  - Disadvantage: Catch up problem with new batch run.

UNIVERSITY OF BATH

Avoiding

- Batch approach (traditional):
  - Keep data.
  - Train new model regularly (e.g. overnight).
  - Disadvantage: Always out of date.
  - Disadvantage: Storage and training time increase with data...
- Batch + Correction:
  - Incrementally train model to correct from last batch.
  - Advantage: Complex batch, simple incremental.
  - Disadvantage: Loss of performance.
  - Disadvantage: Catch up problem with new batch run.
- Remember: Computers are cheaper than people!
  (but not free)

Iterative initialisation

- Algorithms optimise parameters.
- At time t learn θ_t from data D_t.
- At time $t+1$ with D_{t+1} parameters θ_{t+1} will *probably be similar*.

UNIVERSITY OF BATH

Iterative initialisation

- Algorithms optimise parameters.
- At time t learn θ_t from data D_t.
- At time $t+1$ with D_{t+1} parameters θ_{t+1} will *probably be similar*.

- Exploit: Use θ_t to initialise, e.g. gradient descent when learning θ_{t+1}.
- Iteration – add extra data – iteration – add extra data...

- This works for most algorithms...

Data overload

- . . . until you have too much data.
- Eventually you run out of memory, or the iterations take too long.

UNIVERSITY OF BATH

Data overload

- . . . until you have too much data.
- Eventually you run out of memory, or the iterations take too long.

- Challenge is too much data!
- Incremental learning = summarising data (to have finite size).

UNIVERSITY OF BATH

Throwing data away

- Easiest summary: Only keep n exemplars!

UNIVERSITY OF BATH

Throwing data away

- Easiest summary: Only keep n exemplars!
- Reservoir sampling: Incrementally select n items from a list (equal probability, potentially infinitely long):
  1. Keep first n exemplars.
  2. For each further item:
     - Keep with probability n/i, where i is the items index (1 based).
     - If keeping replace a current exemplar at random (uniform).

  (Can do both steps with single random draw)
- Also variants for weighted exemplars.

UNIVERSITY OF BATH

Throwing data away

- Easiest summary: Only keep n exemplars!
- Reservoir sampling: Incrementally select n items from a list (equal probability, potentially infinitely long):
  1. Keep first n exemplars.
  2. For each further item:
     - Keep with probability n/i, where i is the items index (1 based).
     - If keeping replace a current exemplar at random (uniform).

  (Can do both steps with single random draw)
- Also variants for weighted exemplars.
- Flaws:
  - Limits confidence (algorithm never sees more than n exemplars).
  - Indiscriminate (can throw away rare cases).

Statistics I

- Some statistics can be incrementally updated:

Statistics I

- Some statistics can be incrementally updated:
- Mean:

```
count = 0
mean = 0.0
for value in data:
  count += 1
  mean += (value - mean) / count
```

Statistics II

- Some statistics can be incrementally updated:
- Variance (includes mean):

```
count = 0
mean = 0.0
scatter = 0.0 # variance * count
for value in data:
  count += 1
  delta = value - mean
  mean += delta / count
  scatter += delta * (value - mean)
variance = scatter / count
```

(Note how scatter uses the delta before then after the mean update!)

Statistics III

- Some statistics can be incrementally updated:
- Covariance (generalisation of variance):

```
count = 0
mean_x = 0.0
mean_y = 0.0
coscatter = 0.0
for x, y in data:
  count += 1
  delta_x = value - mean_x
  mean_x += delta_x / count
  mean_y += (y - mean_y) / count
  coscatter += delta_x * (y - mean_y)
covar = coscatter / count
```

(One delta has to be from before mean update, other after!)

Statistics IV

- Some statistics can be incrementally updated:
- Median (converges in the limit):

```
delta = 0.01
median = 0.0
for value in data:
  if value<median:
    median -= delta
  else:
    median += delta
```

Delta: Smaller = Slower convergence but more accurate estimate.
Can adjust delta dynamically for better performance, a histogram may be a better choice.

Statistics V

- Some statistics can be incrementally updated:
- Percentile (converges in the limit, p is the desired quantile):

```
delta = 0.01
quantile = 0.0
for value in data:
  if value<quantile:
    median -= delta * (2 - 2*p)
  else:
    median += delta * (2*p)
```

Statistics VI

- Some statistics can be incrementally updated:
- Min/max – obvious!

Sufficient statistics

- Mean, variance and covariance are enough for (Multivariate) Gaussian.
- True for many others:
  e.g. Gamma distribution, `Gamma`(α, β) (shape, scale)

$$\alpha = \frac{\mu^2}{\sigma^2}$$

$$\beta = \frac{\sigma^2}{\mu}$$

where μ = mean, σ^2 = variance.

Incremental Linear Regression

- Model: $y = \vec{\beta}^T \vec{x}$ (assuming $\vec{x}$ contains a 1)
- Inference: Solve $\vec{\beta} = (X^T X)^{-1}(X^T y)$ (X is a data matrix, exemplars as rows).

UNIVERSITY OF BATH

Incremental Linear Regression

- Model: $y = \vec{\beta}^T \vec{x}$ (assuming $\vec{x}$ contains a 1)
- Inference: Solve $\vec{\beta} = (X^T X)^{-1}(X^T y)$ (X is a data matrix, exemplars as rows).

- $(X^T X)_{rc} = n\texttt{covariance}(x_r, x_c)$
- $(X^T y) = n\texttt{mean}(X^T y)$
  (Note that the n cancel out, which is good for numerical stability)

UNIVERSITY OF BATH

Incremental Linear Regression

- Model: $y = \vec{\beta}^T \vec{x}$ (assuming $\vec{x}$ contains a 1)
- Inference: Solve $\vec{\beta} = (X^T X)^{-1}(X^T y)$ (X is a data matrix, exemplars as rows).

- $(X^T X)_{rc} = n\texttt{covariance}(x_r, x_c)$
- $(X^T y) = n\texttt{mean}(X^T y)$
  (Note that the n cancel out, which is good for numerical stability)

- Maintain statistics.
- Solve with gradient descent initialised from last estimate.

Conjugate priors

- Bayes rule (yet again, M for model parameters, D for data):

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)}$$

- Conjugate means:
  - $P(M|D)$ and $P(M)$ are same kind of distribution.
  - Normalisation is automatic – ignore $P(D)$.
  - Limited model choices, which may not fit your data.

Conjugate priors

- Bayes rule (yet again, M for model parameters, D for data):

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)}$$

- Conjugate means:
  - $P(M|D)$ and $P(M)$ are same kind of distribution.
  - Normalisation is automatic – ignore $P(D)$.
  - Limited model choices, which may not fit your data.
- Conjugate priors are naturally incremental!

Conjugate prior example I

- Learn class ($c \in C$) membership probabilities, M (vector),
  $$\therefore x \sim \texttt{Cat}(M), \qquad P(x = c) = M_{[c]}$$
- Conjugate to categorical is Dirichlet,
  $$\therefore M \sim \texttt{Dir}(\vec{\theta}_0) = P(M).$$
- Assume uniform prior, $\therefore \vec{\theta}_0 = [1, 1, 1, \ldots]^T$.

Conjugate prior example II

- At step t new data x_t arrives:

$$P(M_t|x_t, x_{t-1}, \ldots) \propto P(x_t|M_{t-1})P(M_{t-1}|x_{t-1}, x_{t-2}, \ldots)$$

UNIVERSITY OF BATH

Conjugate prior example II

- At step t new data x_t arrives:

$$P(M_t|x_t, x_{t-1}, \ldots) \propto P(x_t|M_{t-1})P(M_{t-1}|x_{t-1}, x_{t-2}, \ldots)$$

$$\texttt{Dir}(M_t|\theta_t) \propto \texttt{Cat}(x_t|M_{t-1})\texttt{Dir}(M_{t-1}|\theta_{t-1})$$

Conjugate prior example II

- At step t new data x_t arrives:

$$P(M_t|x_t, x_{t-1}, \ldots) \propto P(x_t|M_{t-1})P(M_{t-1}|x_{t-1}, x_{t-2}, \ldots)$$

$$\texttt{Dir}(M_t|\theta_t) \propto \texttt{Cat}(x_t|M_{t-1})\texttt{Dir}(M_{t-1}|\theta_{t-1})$$

$$\prod_{c\in C} M_{[c],t}^{\theta_{t,[c]}-1} \propto \prod_{c\in C} M_{[c],t-1}^{\delta(x_t=c)} \prod_{c\in C} M_{[c],t-1}^{\theta_{[c],t-1}-1}$$

(Have omitted normalisation, δ is Kronecker delta)

Conjugate prior example II

- At step t new data x_t arrives:

$$P(M_t|x_t, x_{t-1}, \ldots) \propto P(x_t|M_{t-1})P(M_{t-1}|x_{t-1}, x_{t-2}, \ldots)$$

$$\texttt{Dir}(M_t|\theta_t) \propto \texttt{Cat}(x_t|M_{t-1})\texttt{Dir}(M_{t-1}|\theta_{t-1})$$

$$\prod_{c\in C} M_{[c],t}^{\theta_{t,[c]}-1} \propto \prod_{c\in C} M_{[c],t-1}^{\delta(x_t=c)} \prod_{c\in C} M_{[c],t-1}^{\theta_{[c],t-1}-1}$$

(Have omitted normalisation, δ is Kronecker delta)

$$\prod_{c\in C} M_{[c],t}^{\theta_{t,[c]}-1} \propto \prod_{c\in C} M_{[c],t-1}^{(\theta+\mathbb{1}(x_t))_{[c]}-1}$$

($\mathbb{1}(x_t)$ is the *indicator function*: zero everywhere except for index x_t, which is 1)

Conjugate prior example II

- At step t new data x_t arrives:

$$P(M_t|x_t, x_{t-1}, \ldots) \propto P(x_t|M_{t-1})P(M_{t-1}|x_{t-1}, x_{t-2}, \ldots)$$

$$\texttt{Dir}(M_t|\theta_t) \propto \texttt{Cat}(x_t|M_{t-1})\texttt{Dir}(M_{t-1}|\theta_{t-1})$$

$$\prod_{c\in C} M_{[c],t}^{\theta_{t,[c]}-1} \propto \prod_{c\in C} M_{[c],t-1}^{\delta(x_t=c)} \prod_{c\in C} M_{[c],t-1}^{\theta_{[c],t-1}-1}$$

(Have omitted normalisation, δ is Kronecker delta)

$$\prod_{c\in C} M_{[c],t}^{\theta_{t,[c]}-1} \propto \prod_{c\in C} M_{[c],t-1}^{(\theta+\mathbb{1}(x_t))_{[c]}-1}$$

($\mathbb{1}(x_t)$ is the *indicator function*: zero everywhere except for index x_t, which is 1)

$$\theta_t = \theta_{t-1} + \mathbb{1}(x_t)$$

UNIVERSITY OF BATH

Model drift

- Model may change with time, e.g financial markets change.

Model drift

- Model may change with time, e.g financial markets change.
- Common approach is exponential falloff:

$$w_i = e^{-\lambda t_t}$$

w_i = weight of exemplar i, used for training.
t_i = is age of exemplar i.
λ is the decay constant; given h, the half life, $\lambda = \frac{\ln 2}{h}$.

- Exponential falloff is equivalent to repeated multiplication: It's incremental.

Model drift

- Model may change with time, e.g financial markets change.
- Common approach is exponential falloff:

$$w_i = e^{-\lambda t_t}$$

w_i = weight of exemplar i, used for training.
t_i = is age of exemplar i.
λ is the decay constant; given h, the half life, $\lambda = \frac{\ln 2}{h}$.

- Exponential falloff is equivalent to repeated multiplication: It's incremental.
- Approaches:
  - Can use weighted reservoir sampling and a model that takes weighted data.
  - Reduce n for incremental mean etc.
  - Can sometimes adjust parameters of conjugate priors to hallucinate less data.

Incremental Gaussian mixture model I

(There are multiple approaches)

Incremental Gaussian mixture model I

(There are multiple approaches)

- Kernel density estimate: Each new exemplar adds a Gaussian mixture component.

Incremental Gaussian mixture model I

(There are multiple approaches)

- Kernel density estimate: Each new exemplar adds a Gaussian mixture component.
- Mixture limit: User set cap, n, on number of mixture components.
- When limit is passed two mixtures are merged to bring it back to the limit.

UNIVERSITY OF BATH

Incremental Gaussian mixture model I

(There are multiple approaches)

- Kernel density estimate: Each new exemplar adds a Gaussian mixture component.
- Mixture limit: User set cap, n, on number of mixture components.
- When limit is passed two mixtures are merged to bring it back to the limit.
- Merging pair selected to minimise the Kullback-Leiber divergence.

Incremental Gaussian mixture model II

- Each mixture component, i, has:
  - w_i = Total weight assigned to it (unnormalised).
  - μ_i = It's mean.
  - Σ_i = It's covariance.

Incremental Gaussian mixture model II

- Each mixture component, i, has:
  - w_i = Total weight assigned to it (unnormalised).
  - μ_i = It's mean.
  - Σ_i = It's covariance.

- Each exemplar, x_i, creates a component:
  - $w_i = 1$
  - $\mu_i = x_i$
  - Σ_i = hyper-parameter, usually identity matrix scaled by *bandwidth* (KDE).

UNIVERSITY OF BATH

Incremental Gaussian mixture model II

- Each mixture component, i, has:
  - w_i = Total weight assigned to it (unnormalised).
  - μ_i = It's mean.
  - Σ_i = It's covariance.
- Each exemplar, x_i, creates a component:
  - $w_i = 1$
  - $\mu_i = x_i$
  - Σ_i = hyper-parameter, usually identity matrix scaled by *bandwidth* (KDE).
- First n kept, then merging starts...

Incremental Gaussian mixture model II

- Merge i and j to get m:

$$w_m = w_i + w_j$$

$$\mu_m = \frac{w_i}{w_m}\mu_i + \frac{w_j}{w_m}\mu_j$$

$$\Sigma_m = \frac{w_i}{w_m}\left(\Sigma_i + (\mu_i - \mu_m)(\mu_i - \mu_m)^T\right) + \frac{w_j}{w_m}\left(\Sigma_j + (\mu_j - \mu_m)(\mu_j - \mu_m)^T\right)$$

- Kullback-Leiber divergence:

$$\mathtt{KL}(\{w,\mu,\Sigma\}_p||\{w,\mu,\Sigma\}_q) = \frac{1}{2}\left(\log\frac{|\Sigma_q|}{|\Sigma_p|} + \mathtt{Tr}\left(\Sigma_q^{-1}\Sigma_p\right) + (\mu_p - \mu_q)\Sigma_q^{-1}(\mu_p - \mu_q)^T - d\right)$$

(d is the number of dimensions)

- Cost of merging i and j to get m (cache this):

$$\mathtt{cost}(\{w,\mu,\Sigma\}_i, \{w,\mu,\Sigma\}_j) = w_i\mathtt{KL}(\{w,\mu,\Sigma\}_i||\{w,\mu,\Sigma\}_m) + w_j\mathtt{KL}(\{w,\mu,\Sigma\}_j||\{w,\mu,\Sigma\}_m)$$

UNIVERSITY OF BATH

Recommendations

- If good enough: Conjugate priors or distribution fitting.
- Lots of storage: Reservoir sampling.
- Less storage: Incremental Gaussian mixture model.

- None of these good enough: Read or do research!

Random forest

- Can be modified to be incremental, though not robust to lots of data.
- Requires combining lots of tricks:

Random forest

- Can be modified to be incremental, though not robust to lots of data.
- Requires combining lots of tricks:
  1. Poisson bootstrap: Repeat count, r, of each item for each tree is $r \sim \texttt{Poisson}(1)$. Makes bootstrap incremental.
  2. Update leaves with statistics of data that passes down them.
  3. Extend leaves when they have enough data.
  4. Train new trees (replace tree with worst out of bag error; necessary as new data may require new tree structures).

UNIVERSITY OF BATH

Summary

- Incremental learning is about data overload.
- Filtering approach.
- Summarising approaches.
- Conjugate priors.
- Algorithm specific approaches.

UNIVERSITY OF BATH

Further reading

- Incremental Gaussian mixture model discussed above:
  "*Incremental One-Class learning with Bounded Computational Complexity*",
  by R. R. Sillito and R. B. Fisher (2007).

- Comparison of lots of incremental learning approaches:
  "*Choosing the Best Algorithm for an Incremental On-line Learning Task*",
  by V. Losing, B. Hammer and H. Wersing (2016).