COMP 790-124: Goals for today

- Semisupervised learning
  - Self-training
  - Latent-SVM
  - Co-training
- Active Learning
  - Query strategies
  - Parameter view

Semi-supervised learning tasks

We are given training feature vectors $\{\mathbf{x}_t : t = 1, \dots T\}$ and an incomplete set of target variables $\{y_t : t \in 1, \dots, L < T\}$.

We want to learn a rule for predicting target variables $y \in \mathcal{Y}$ given their corresponding feature vector $\mathbf{x} \in \mathcal{X}$.

Naively, we could just ignore unlabeled data and use set $\{(y_t, \mathbf{x}_t) : t = 1, \dots, L\}$ throwing away unlabeled feature vectors.

Semi-supervised methods take advantage of *both* labeled and unlabeled parts of the datasets as-is.

Active learning methods choose which labels to uncover/buy.

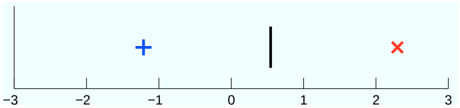
Unlabeled data is cheaper, but is it useful

The unlabeled data is cheaper to obtain.

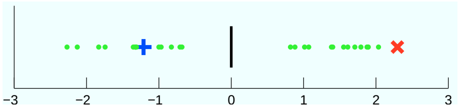
Labeling/measurement can be costly.[1]

What could we possibly learn from unlabeled data?

Supervised (only labeled data)



Semisupervised (both labeled and unlabeled data)



[1]Crowdsourcing can be cheap, e.g. Amazon's Mechanical Turk, but sadly an average Turker doesn't seem to have a PhD in Bio

How to go about semisupervised learning?

In generative models easy

$$L(\theta) = \underbrace{\prod_{t=1}^{L} p(\mathbf{x}_t, y_t|\theta)}_{\text{labeled}} \underbrace{\prod_{t=L+1}^{T} \sum_{y_t} p(\mathbf{x}_t, y_t|\theta)}_{\text{unlabeled}}$$

note that $\{y_t : t = L + 1 : T\}$ are hidden so we use EM.

$$E : q(y_t) \quad \propto \quad p(\mathbf{x}_t, y_t|\theta), \qquad t > L$$

$$M : \theta^{\text{new}} \quad = \quad \operatorname*{argmax}_{\theta} \underbrace{\sum_{t=1}^{L} \log p(\mathbf{x}_t, y_t|\theta)}_{\text{labeled}} + \underbrace{\sum_{t=L+1}^{T} \sum_{y_t} q(y_t) \log p(\mathbf{x}_t, y_t|\theta)}_{\text{unlabeled}}$$

In a generative model, semi-supervised learning is not a special case.

Some variables are hidden, others are observed – what else is new?

How to go about semisupervised learning?

Things go less smoothly with conditional likelihood optimization: the naive plug-in fails

$$CL(\theta) = \underbrace{\prod_{t=1}^{L} p(y_t|\mathbf{x}_t\theta)}_{\text{labeled}} \underbrace{\prod_{t=L+1}^{T} \overbrace{\sum_{y_t} p(y_t|\mathbf{x}_t, \theta)}^{=1}}_{\text{unlabeled}}$$

The unlabeled data is ignored.

Self-training

We can use a different objective

$$CL(\theta) = \underbrace{\prod_{t=1}^{L} p(y_t|\mathbf{x}_t,\theta)}_{\text{labeled}} \underbrace{\prod_{t=L+1}^{T} p(y_t^*|\mathbf{x}_t,\theta)}_{\text{unlabeled}}$$

where

$$y_t^* = \operatorname*{argmax}_y p(y|\mathbf{x}_t,\theta), t > L$$

is the MAP assignment for the label.

The algorithm is simple

- assign MAP labels to unlabeled instances $(t > L)$
- refit the conditional model $p(y|\mathbf{x},\theta)$ using the now complete set of labels

In general

The self-training algorithm can be used in general

- let $y_t = f(\mathbf{x}^t)$, $t > L$
- refit the f to minimize some loss on the completed set of labels

Convergence of the self-training algorithm in general

To show convergence of the algorithm you would recognize that it is a coodinate descent on the objective

$$\operatorname*{argmin}_{q, f \in \mathcal{F}} \sum_{t=1}^{L} \mathrm{Loss}(y_t, f(\mathbf{x}_t)) + \sum_{t=L+1}^{T} \sum_{y_t} q(y_t) \mathrm{Loss}(y_t, f(\mathbf{x}_t))$$

where

$$q(y_t) = [y_t = v]$$

if the loss is convex then so is expectation with respect to a distribution (q).

You can convince yourselves that the algorithm as shown on the previous slide indeed corresponds to coordinate descent in q and f on this objective.

Latent SVMs

Unsurprisingly, the SVM community arrives at this algorithm differently and refers to the general framework of dealing with the hidden variables in SVMs as latent SVMs.

Objective

$$\begin{aligned}
\underset{w,\xi,\{y_t : t>L\}}{\text{minimize}}\ & \|w\|_2^2 + C\sum_t \xi_t \\
\text{subject to } & y_t(w^T\mathbf{x}_t + b) - 1 + \xi_t \geq 0 && \forall t \in \text{Train} \\
& \xi_t \geq 0 && \forall t \in \text{Train} \\
& y_t \in \{-1, +1\} && t > L
\end{aligned}$$

This is a mixed-integer program.

The self-training algorithm we've shown earlier solves a dual of this problem.

Multi-view and Co-training

The underlying assumption is that there exist two sets of features that are conditionally independent given a class.

These are called **views**.

For example

- View 1: chemical properties of a drug (what it is)
- View 2: effects that administration of a drug has on a variety of molecular measurements (what it does)

If we are interested in predicting label drug-cures-cancer we can *co-train* two classifiers on these two views.

Co-training

```
Split features $\mathbf{x}$ into views $\mathbf{x}^A$ and $\mathbf{x}^B$
$\mathrm{CoLabeled} = \{1, \ldots \mathrm{L}\}$
while not all co-labeled do
    train classifier $f^A$ on $\{(y_t, \mathbf{x}_t^A) : t \in \mathrm{CoLabeled}\}$
    train classifier $f^B$ on $\{(y_t, \mathbf{x}_t^B) : t \in \mathrm{CoLabeled}\}$
    for $f^A$ choose an unlabeled instance $s^A$ on which it is most confident
    for $f^B$ choose an unlabeled instance $s^B$ on which it is most confident
    update $y_{s^A} = f^A(\mathbf{x}_{s_A}^A)$ and $y_{s^B} = f^B(\mathbf{x}_{s_B}^B)$
    $\mathrm{CoLabeled} = \mathrm{CoLabeled} \cup \{s_A, s_B\}$
end
for $u$=$L$+1, ..., $T$ do
    $y_u = \mathrm{argmax}_y \, p_A(y|\mathbf{x}_u^A) p_B(y|\mathbf{x}_u^B)$
end
```

Semi-supervised flavors and related methods

Self-training very close to EM and a de-facto standard for discriminative models with hidden variables.

Co-training work wonders if the features are really independent given the class, i.e. you have two views that are equally informative. It can underperform if the assumption is violated.

There are hybrid methods between EM and co-training and semi-supervised learning is quite active.

Curriculum learning is an opposite idea: ignore the labeled data that is “hard” and learn from easy examples.

Active learning

So far, passive learning – here's a dataset, do your best.

Suppose that instead of inferring a label we were able to query it – at a price.

Which labels should we query and when are we done?

The cost of labeling a new data point may be high but acceptable for a small number of queries (testing a representative drug against a cell line).

Closely related to sequential experiment design.

Examples

In a cancer classification problem[2] to achieve the same accuracy

- 174 labeled instances required by a passive learner
- 31 labeled instances required by an active learner

[2]Active Learning with Support Vector Machine Applied to Gene Expression Data for Cancer Classification, Liu Y., J. Chem. Inf. Comput. Sci. 2004

Active learning - query strategies

We will assume the same setting as in semi-supervised learning $\{\mathbf{x}_t : t = 1, \ldots T\}$ and $\{y_t : t \in 1, \ldots, L < T\}$.

Suppose we trained some $p(y|\mathbf{x})$ on the labeled set.

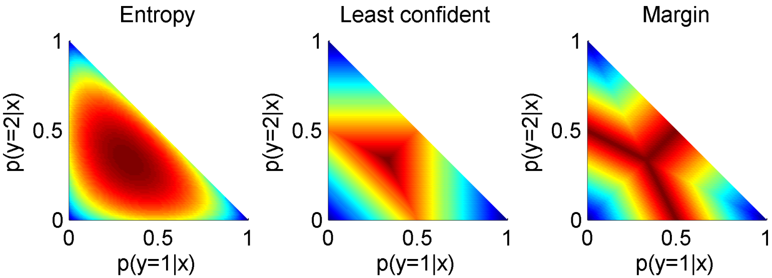
How should we choose the "best" unlabeled instance t^* to query?

- least confident instance $\mathrm{argmax}_t\, 1 - p(y_t|\mathbf{x}_t)$
- smallest margin $\mathrm{argmin}_t\, p(y_t = a|\mathbf{x}_t) - p(y_t = b|\mathbf{x}_t)$ where a is the best and b second best label for instance t
- largest entropy $\mathrm{argmax}_t - \sum_{y_t} p(y_t|\mathbf{x}_t) \log p(y_t|\mathbf{x}_t)$

Query score for different distributions over 3 labels

Each point in the triangle (simplex) corresponds to a distribution over 3 values.

$$p(y=3|\mathbf{x}) = 1 - p(y=2|\mathbf{x}) - p(y=1|\mathbf{x})$$



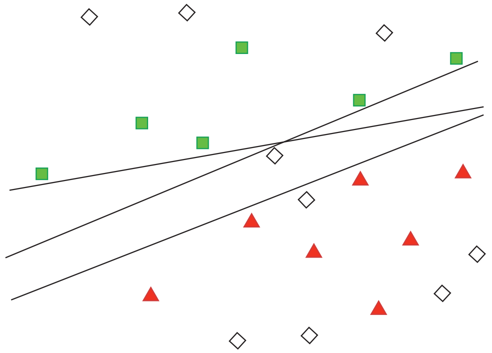
$-\sum p \log p$ $\quad$ $1 - \max p$ $\quad$ $p(y^1) - p(y^2)$

Query by committee

Query-by-committee relies on a set of classifiers $f_1, \ldots, f_C$

The committee members all vote on labeling of an unlabeled data point.

The most contested unlabeled data instance is queried to settle the committee's disagreement.



Measuring committee disagreement

Paralleling the construction of the most informative point to label, a number of committee disagreement measures have been put forward.

- vote entropy

$$t^* = \operatorname*{argmax}_t - \sum_{y_t} \frac{\sum_k [f_k(\mathbf{x}_t) = y_t]}{C} \log \frac{[f_k(\mathbf{x}_t) = y_t]}{C}$$

- KL-divergence

$$t^* = \operatorname*{argmax}_t \sum_c \mathrm{KL}\left(p_c(y|x_t) \middle\| \frac{1}{C} \sum_{i=1}^{c} p_i(y|x_t)\right)$$

Parameter view

So far we looked at the uncertainty of the predicted label and tried to decide what to query based on the reduction in that uncertainty.

The alternative view takes into account the underlying parameters.

If we were to label another data instance would we reduce *uncertainty on parameters*, e.g. separating hyperplane's normal w

In Comp Bio this is a more important view esp. if your model is parameterized in terms of physically measurable quantities (e.g. binding affinity).

Uncertainty in parameters – without a prior

If our model is

$$p(\mathbf{x}|y,\theta)p(y|\theta)$$

then the log likelihood is

$$\mathrm{LL}(\theta) = \sum_{t=1} \log \sum_{y} p(\mathbf{x}_t|y,\theta)p(y|\theta)$$

and its Hessian is called the observed Fisher information

$$[J_n(\theta)]_{ij} = -\sum_{t} \frac{\partial^2}{\partial\theta_i \partial\theta_j} \log \sum_{y} p(\mathbf{x}_t|y,\theta)$$

and $\left(\frac{1}{T} J_n\right)^{-1}$ quantifies our uncertainty about the parameters around the ML estimate of θ.

Uncertainty in parameters – with a prior

If our model also has a prior

$$p(\mathbf{x}|y,\theta)p(y|\theta)p(\theta)$$

we can construct a posterior on parameters

$$p(\theta|\mathbf{x}_{1:T}, y_{1:L}).$$

We can approximate this posterior with a Gaussian distribution (this is a Laplace approximation[3])

$$p(\theta|\mathbf{x}_{1:T}, y_{1:L}) \approx \mathcal{N}(\theta_{MAP}, \mathbf{diag}(\sigma_1^2, \ldots, \sigma_k^2))$$

then we can obtain estimates of our uncertainty in different parameters, e.g. σ_l^2 is variance of parameter θ_l.

[3]do not mistake for approximation with Laplacian distribution. The Laplace approximation uses Gaussian distribution

Parameter view

Equipped with approximate parameter covariance Σ we can choose the data-point that results in the biggest reduction in parameter uncertainty.

There are a number of ways of converting a matix Σ into a single scalar $U(\Sigma)$ measuring overall uncertainty, corresponding to different optimal designs.

Objective based on Determinant design

D-optimal design: $U(\Sigma) = \log \det \Sigma$

Entropy of a Gaussian distribution is:
$k/2(\log 2\pi + 1) + 1/2 \log \det \Sigma$

We can view D-design as minimizing the entropy of the normal approximation to the parameter distribution.

Geometrically: shrink the volume of the ellipsoid occupied by the likely parameters.

Objective based on Average design

A-optimal design: $U(\Sigma) = \mathbf{Tr}\,\Sigma = \sum_i \Sigma_{ii}$

Simply collect all of the diagonal entries, variances of corresponding parameters.

If Σ is diagonal, similar to D-design.

Geometrically: shrink the volume of the axis-aligned ellipsoid occupied by the likely parameters.

Objective based on Extreme design

E-optimal design: $U(\Sigma) = \lambda_{\max} = \max_x \frac{\|\Sigma x\|}{\|x\|}$

Find the direction of the largest diameter and shrink that one.

Geometrically: shrink the volume of a sphere containing the ellipsoid occupied by the likely parameters.

Algorithmics

Iterate while sufficient change in U

- for each instance compute average/expected change in U across all outcomes
- query the best instance

Robot scientist

King et al, **The Automation of Science**. Science 324 (5923)

We did ...

- Semisupervised learning
  - Self-training
  - Latent-SVM
  - Co-training
- Active Learning
  - Query strategies
  - Parameter view