Active Learning Notes

Nicholas Denis

August 21, 2019

Contents

1 Two faces of active learning

Sanjoy Dasgupta.

We are in the business of binary classification, where we do not want to make any errors. That is, for a given hypothesis in our hypothesis space, $h \in \mathcal{H}$, given an underlying distribution generating the data P over $\mathcal{X} \times \mathcal{Y}$, we have that

$$err(h) = P[h(x) \neq y]$$

The data may be separable (e.g. $err(h^*) = 0$), or not (e.g. $err(h^*) > 0$).

A popular approach to active learning is to maintain the current *version space*, $V \subset \mathcal{H}$, is the set of hypotheses that are all consistent with respect to the data seen thus far. That is, given a sequence of data $\{x_i, y_i)_{i \in \mathcal{D}}$,

$$V = \{h \in \mathcal{H} | h(x_i) = y_i, \ \forall i \in \mathcal{D}\}$$

The intuition is to query data points that will split the version space in half at each step. An active learning algorithm is *aggressive* if it can select any $x \in \mathcal{X}$. A *mellow* algorithm is one where it receives a sequence of inputs, and can each decide to ask for a label, or not. One example is CAL (Cohn, Atlas, Ladner), which at each time step maintains the current version space V_t. Upon receive x_t, if there is a disagreement in V_t about the label of x_t, then the label is queried, and V_{t+1} is updated. Maintaining the entire version space is impossible, however approximations and algorithms that follow this intuition have been developed.

1.1 Label Complexity

This is simply a generalization of sample complexity, where now we measure the number of labels received, rather than samples,in order to achieve a classification accuracy or error rate of within ϵ of optimal. For supervised learning, given $VCdim(\mathcal{H}) = d < \infty$, then the sample (label) complexity is $\mathcal{O}(\frac{d}{\epsilon^2})$. PAC learning deals with the sample complexity $\mathcal{L}(\epsilon, \delta)$ associated to returning a hypothesis h such that $\exists T_0$, where for any sample of size $T \geq T_0$, $P[err(h) > \epsilon] \leq \delta$.

Theorem 1. *Let $\mathcal{H}$ have finite VC dimension d, and the learning problem is separable with disagreement coefficient θ. Then,*

$$\mathcal{L}_{CAL}(\epsilon) \leq \hat{\mathcal{O}}\Big(\theta d log \frac{1}{\epsilon}\Big).$$

1.2 Disagreement Coefficient

The error of a hypothesis can be used to induce a pseudo-metric on $\mathcal{H}$.

Definition 2. (Disagreement Coefficient) $\forall h,\ h' \in \mathcal{H}$, define

$$d(h, h') = P[h(x) \neq h'(x)].$$

The disagreement region of version space V is,

$$DIS(V) = \{x \in \mathcal{X} |\ \exists h,\ h' \in V, where\ h(x) \neq h'(x)\}.$$

Then, denoting $B(h, r)$ to be the ball of radius r around hypothesis h using the pseudo-metric defined above, we define the disagreement coefficient, θ:

$$\theta = \sup_{r>0} \frac{P[DIS(B(h^*, r))]}{r}$$

When θ is bounded, CAL and other known active learning algorithms have better label complexity than supervised learning, however there do exist hypothesis classes with arbtriarily large disagreement coefficient.

2 A general agnostic active learning algorithm

Sanjoy Dasgupta, Daniel Hsu, Claire Monteleoni

This algorithm extends the CAL algorithm, which must maintain the current version space explicitly, as well as the region of uncertainty. The algorithm uses selecting sampling approach. The idea is that data obtained thus far, $\mathcal{D}$ is split into two sets:

$$\mathcal{D} = \hat{S} \cup T$$

where $\hat{S}$ is the data that has not been queried for labels thus far. These points are assigned the labels that the algorithm thinks they are; and T is the set of data for which labels have been requested.

The idea is that $\hat{S}$ will remain consistent with the *best* separator of the classes, while counter-intuitively, the labels of T could be inconsistent with the best separator. To decide whether the algorithm is uncertain about the label of a point x, two new hypotheses h_+ and h_- are learned, where:

$h_+ =$ is consistent with all the labels in $\hat{S} \cup \{(x, +1)\}$ and has minimal empirical error on T, while

$h_- =$ is consistent with all the labels in $\hat{S} \cup \{(x, -1)\}$ and has minimal empirical error on T.

The idea being, if the *true* error of h_+ is much larger than that of h_-, then we may be confident that the best separator must also label x with -1. If the error is modest, then we request the label of x. Generalization bounds for an i.i.id sample allows for the use of empirical errors on $\hat{S} \cup T$ as a proxy. The algorithm is as follows:

- Note: $A, B \subset \mathcal{X} \times \{-1, +1\}$ let LEARN(A,B) denote the supervised learner that returns a hypthesis $h \in \mathcal{H}$ consistent with A, and with minimum error on B. If no such h exists, this is reported.
- Given $(x_1, x_2, ...)$ i.i.d. from $\mathbb{P}_{\mathcal{X}}$
- Initialize $\hat{S}_0, T_0 = \emptyset$
- for $n = 1, 2, ..., m$ steps
  - $\forall \hat{y} \in \{-1, +1\}$, $h_{\hat{y}} = \text{LEARN}(\hat{S}_{n-1} \cup \{(x_n, \hat{y})\}, T_{n-1})$.
  - If $\text{err}(h_{-1}, \hat{S}_{n-1} \cup T_{n-1})$ - $\text{err}(h_{+1}, \hat{S}_{n-1} \cup T_{n-1}) > \Delta_{n-1}$ then $\hat{S}_n := \hat{S}_{n-1} \cup \{(x_n, \hat{y})\}$ and $T_n := T_{n+1}$ (note: this is stated WLOG, however both directions of the inequality are checked and $\hat{y}$ is defined as one would expect).
  - Else, request label y_n; $\hat{S}_n := \hat{S}_{n-1}$ and $T_n = T_{n-1} \cup \{(x_n, y_n)\}$
- Return $\text{LEARN}(\hat{S}_m, T_m)$.

m, and Δ_n are chosen such that with high probability, if the error is larger than Δ_n, then it is highly probable that the prediction is correct, and thus $\hat{S}$ contains data instances that are highly likely to be consistent with h^*. Since h^* might make mistakes on some examples in T, we only require that LEARN minimizes the error on T, rather than be consistent, meanwhile h^* is consistent on $\hat{S}$.

The authors provide theoretical results that show that this agnostic active learning algorithm can perform no worse than an agnostic supervised learning algorithm. They also provide empirical results on synthetic data sets.

3 Improving generalization with active learning: CAL

David Cohn, Les Atlas, Richard Lander

A simple example where AL can help over passive learning is in determining the decision boundary on the unit interval in the realizable case. Here, AL can achieve label complexity of $\mathcal{O}\left(ln\frac{1}{\epsilon}\right)$, whereas a supervised learning algorithm requires $\mathcal{O}\left(\frac{1}{\epsilon}ln\frac{1}{\epsilon}\right)$, resulting in an exponential speedup in learning. Some definitions: For a target concept h^*, a training example isa pair $(x, h^*(x))$, drawn from some distribution $\mathbb{P}$. A concept c is consistent with an example if $h^*(x) = c(x)$. The error, $\text{err}(c, h^*, \mathbb{P})$ is defined as:

$$\text{err}(c, h^*, \mathbb{P}) = \mathbb{E}_{\mathbb{P}}[c(x) \neq h^*(x)].$$

3.1 Region of Uncertainty

We will consider a concept class C, and a set of m training examples S^m. We are interested in the subset of C that are consistent on S^m, as well as where these consistem hypothesis may differ. The region of uncertainty is defined as

$$\mathcal{R}(S^m) := \{x \mid \exists c_1, c_2 \in C,\ c_1, c_2 \text{ are consistent with all } s \in S^m, \text{ and } c_1(x) \neq c_2(x)\}.$$

For an arbitrary distribution P we can define the size of this region as

$$\alpha = P[x \in \mathcal{R}(S^m)].$$

The region of uncertainty acts as an envelope for the consistent hypothesis/concepts, and any disagreement between these hypotheses must lie in $\mathcal{R}(S^m)$. As well, α, the measure on $\mathcal{R}(S^m)$ acts as an uper bound on the potential error of any consistent hypothesis given m data points, and also can be used to bound the probability of a newly labelled point in reducing the error.

The authors assume the following setup: data streams in $(x_1, x_2, x_3, ...)$ and with each data point the option to query an oracle for $h^*(x_n)$ is made. Note that this induces a sampling bias on the labels (selective sampling). The authors note that explicitly representing the region of uncertainty can be infeasible. Hence, the goal is to approximate $\mathcal{R}(S^m)$ somehow. The authors note that, suppose one is able to maintain a superset $\mathcal{R}(S^m) \subset \mathcal{R}^+(S^m)$. If we can do so, and sample from $\mathcal{R}^+$ then we can be assured that no points within $\mathcal{R}(S^m)$ will be excluded from the sampling procedure. The only issue here is the inefficiency of sampling points that may not be useful. This inefficiency could be measured as:

$$\frac{P[x \in \mathcal{R}(S^m)]}{P[x \in \mathcal{R}^+(S^m)]}$$

Moreover, we may maintain a similarly defined subset $\mathcal{R}^-(S^m) \subseteq \mathcal{R}(S^m)$.

The authors implement a selective sampling algorithm that uses a neural network, described as the SG-net, that uses concepts of version-spaces and region of uncertainty. They first note that a naive approach would be to sample "uncertain" regions by using simple rules: if the network predicts greater than 0.9, or less than 0.1, the prediction is certain; else, it is uncertain and so this should be used to query an oracle. A problem here is that it measures uncertainty of this specific network (e.g. $h \in \mathcal{H}$) and not the actual region of uncertainty of the hypothesis class (version space). A *version space*:

$$V_{S^m} = \{h \in \mathcal{H} \mid h \text{ is consistent with all } x \in S^m\}.$$

To bound the hypotheses in the version space, the algorithm maintains two subsets, $S, G \subseteq V_{S^m}$. S is the set of all "most specific" consistent hypotheses (e.g. $S = \{h \in V_{S^m} \mid \nexists h' \in V_{S^m},\ h'^{-1}(\{+1\}) \subsetneq h^{-1}(\{+1\})\}$. G is the "most

general" concepts, (e.g. $G = \{h \in V_{S^m} \,|\nexists h' \in V_{S^m}, h^{-1}(\{+1\}) \subsetneq h'^{-1}(\{+1\})\}$). In some sense, S is the tightest version space consistent on S^m, and G is the largest version space consistent on S^m.

One can do active learning with the version space by finding data points that fall in the symmetric difference of S,G. If $x \in S\Delta G$ and $h^*(x) = +1$, then some $s \in S$ will have to generalize to accommodate the new information (e.g. S will grow in measure); if $h^*(x) = -1$, then some of the hypotheses in G will have to excluded (e.g. G will shrink in measure). Either way, the version space is reduced with every query of points from $S\Delta G$.

3.2 Implementing an active version space search

The authors define an ordering on the version space. They say a hypothesis h_1 is more general than h_2 if and only if for a random point $x \sim \mathbb{P}$, $\mathbb{P}[x \in h_1^{-1}(\{+1\})] > \mathbb{P}[x \in h_2^{-1}(\{+1\})]$. Under this ordering, we may find a single most general hypothesis g and a single most specific hypothesis s. Now, $s\Delta g$ will be a subset of $S\Delta G$, and when we sample labels from points in $s\Delta g$, again, either we will invalidate s or g, and hence can iteratively approximate a step-by-step traversal of S and G.

3.3 SG-net: a neural network version space search algorithm

The authors describe how they approximate learning hypothesis s, g using neural networks. Given a set of data S^m the most specific network s is one that is consistent on S^m, in the sense that it correctly classifies all the positive examples in S^m, and classifies as negative everything else (e.g. simply memorizes the positive examples). This is equivalent to finding h consistent with S^m that minimizes $\mathbb{P}[x \in h^{-1}(\{+1\})]$. The authors try to enforce this through altering the inductive bias (adding regularization) to penalize the network from classifying positive examples. One example is randomly sampling unlabelled data from the dataset and labelling them as negative examples. This will prevent the network from generalizing beyond positive examples, and simply memorize the positive examples it has seen. In their particular implementation, they used one learning rate for positive examples and another learning rate for these "negatively sampled" examples. I am not sure if this is necessary, and personally would use a generative model to produce examples of positive data points, then give them negative labels. To generate the g general network, the reverse procedure is done (for negative examples).

3.4 Active learning

Once s, g are learned, simply query a point and if $s(x) \neq g(x)$ then the label is queried and added to the training set. This can be done one at a time, or in a batched pool. One could simply continue training these networks with the

new data instances, or re-start from scratch. They re-train from scratch in this paper, and show positive results on synthetic data sets.

3.5 Comment

I believe this is an interesting approach, especially if coupled with a generative model such as a conditional VAE. Moreover, the true classifier does not need to be a neural network. For example, one could use SVM, random forest, etc, as your model, but rather take advantage of the powerful overfitting properties of neural networks to do the active learning approach (build the s, g hypotheses in the version space), simply to determine which data points to query an oracle to improve your actual model (SVM, random forest, etc).

4 Selecting Sampling sing the Query by Committee Algorithm

Yoav Freund, H.Sebastian Seung, Eli Shamir and Naftali Tishby

The authors note that there are two general measures for "progress" of an active learning algorithm: one is the reduction in the entropy of the posterior distribution that is induced by the answer of a query; the other is the accuracy of future predictions. The authors assume a noise-free realizable setting. The authors assume a Bayesian PAC learning setting, wherein there is a prior distribution over $\mathcal{H}$. They assume the algorithm has access to two oracles: Sample and Label.

The initial version space, $V_0 := \mathcal{H}$, and V_i is the version space after seeing i training examples (with labels). They define the *instantaneous information gain* from the i^{th} labeled example to be: $-log\frac{P(V_i)}{P(V_{i-1})}$, where the probability is taken with respect to the prior distribution. They also define the *cumulative information gain*:

$$\mathcal{I}(\langle x_1, h^*(x_1)\rangle, ..., \langle x_m, h^*(x_m)\rangle) := -logP(V_m).$$

The authors are also interested in knowing the *expected* instantaneous information gain taken with respect to the probability that each of the two labels occurs. This expected value is equivalent to the Shannon information, $\mathcal{H}$, content of the binary random variables whose probability of having label 1 is p. That is:

$$\begin{aligned}\mathbb{E}\Big[-\ \log\frac{P(V_i)}{P(V_{i-1})}\Big|x_i\Big] &= -p\ \log\frac{P(V^1)}{P(V_{i-1})} - (1-p)\log\frac{P(V^0)}{P(V^1_{i-1})}\\ &= -p\ \log(p) - (1-p)\log(1-p)\\ &= \mathcal{H}(p)\end{aligned}$$

The authors define the Gibbs preduction rule to predict the label of a new instance x by picking a hypothesis from the current version space, $h \in V_i$, and labeling x according to h. This sampling is made according to the prior distribution on $\mathcal{H}$, restricted to the version space. They assume they have access to a Gibbs oracle.

Determining how to select examples for labelling is addressed. The first approach considered is that of selecting samples by their expected information gain. They show that in some settings this is sufficient, however in some concept classes, this does not lead to improvement in learning, and so expected information gain is not a sufficient criterion for constructing good queries.

The QBC algorithm at each step randomly samples two hypothesis from the current version space, and if they disagree on their predictions, the label is queried from an oracle. The probability of two hypotheses from the current version space decreases over time, and leads to faster learning over passive learning. The algorithm proposed here calls Sample to receive a new x, then calls Gibbs twice, and compares the two predictions for the label of x. This is similar to the original QBC algorithm, except this paper deals with batch sampling, rather than purely sequential (online).

The remainder of the paper is quite theoretical. They devise a stopping criteria to terminate the algorithm, and prove familiar PAC bounds on their algorithm for paticular concept classes. They provide theoretical results in terms of the number of calls made to the oracles, as well as how long the algorithm will run before terminating. A theoretical paper, however an interesting approach that could be approximated in practice (though expensive).

5 Query by Committee Made Real

Ran Gilad-Bachrach, Amir Navot, Naftali Tishby

This paper introduces Kernel QBC (KQBC). Rather than sampling from the version space, the authors devise a way to sample from a lower dimensional projection of the version space, and make use of linear classifiers. Since they can be kernelized, the approach is broad in scope. The algorithm requires sampling from convex bodies, and suggest a *hit and run* random walk for this. Matlab code is provided.

The authors define the version space:

$$V = \{w : ||w|| \leq 1 \text{ and } \forall i \; y_i \langle w, x_i \rangle > 0\}$$

The authors assume a prior distribution ν over $\mathcal{H}$. The sample S induces a posterior over the class of linear classifiers which is the restriction of ν to V

$(\nu|V)$. They state that the probability that QBC will query for the label of an instance x is exactly: $2P_{w\sim\nu|V}[\langle w,x\rangle > 0]P_{w\sim\nu|V}[\langle w,x\rangle < 0]$. The authors note that rather than sampling two random hypothesis in QBC, it is sufficient to cook up some sampling method that has the exact same probability distribution. They come up with a sampling approach that return $\hat{y}$ with the same probabilities as specified above.

Let T denote the span of $x_1, x_2, ..., x_k$ and of x, where $x_1, ..., x_k$ have labels (previously queried) while x is a new data instance. Rather than sampling from V, the algorithm samples from the intersection of V and T. T is lower dimensional and at most of dimension $k+1$, and this intersection is also a convex body. The input dimension plays a minor role in the sampling algorithm. This approach is also open to kernels. The authors provide a theorem that if ν is uniform, then sampling from the intersection of V, T is equivalent to the formulation mentioned above, satisfying the same probabilities.

Sampling with Kernels. Since any hypothesis $w \in V \cap T$ is of the form:

$$w = \alpha_0 x + \sum_{i=1}^{k} \alpha_i x_i$$

, the hypothesis w assigns the label to a new instance x' via:

$$\begin{aligned}\langle w, x'\rangle &= \Big\langle \big(\alpha_0 x + \sum_{i=1}^{k} \alpha_i x_i\big), x' \Big\rangle \\ &= \alpha_0\langle x, x'\rangle + \sum_{i=1}^{k} \alpha_i \langle x_i, x'\rangle.\end{aligned}$$

Hence, sampling w is the same as sampling $\alpha_0, ..., \alpha_k$. The authors note that sampling from these $\alpha' s$ uniformly is not the same as sampling w's uniformly, as the x_i do not form an orthonormal basis. Hence, they suggest a method for computing the orthonormal basis, thus guaranteeing a way to uniformly sample hypotheses.

5.1 Hit and Run Convex Body Sampling

Given an inital point z in a convex body, C, the algorithm samples a random point u from the unit sphere, and samples a point uniformly from $l \cap C$, where l is the line passing through z and $z + u$. The stationary distribution of this algorithm is the uniform distribution; it mixes fast and does not require a burn in period. There are no parameter tunin other than the number of random steps to take. The authors use this approach on synthetic data sets, as well as classifying images. They compare to SVM, KQBC, as well as using an SVM with samples selected via LQBC. Their results were quite positive. I thought this was a good paper.

6 Generative Adversarial Active Learning

Jia-Jie Zhu, Jose Bento

Interesting paper. They use GANs to synthesize training instances close to an SVM hyper-plane. The closest synthetic instances to the decision boundary are queried by an oracle. The biggest take home from this is that a popular SVM_{active} algorithm outperforms their approach, and so should be looked into. Their approach trains a GAN on unlabeled data. They initialize their labeled set with a small amount of labeled data, then iteratively produces synthetic instances, select a batch of those that are closest to the decision boundary, get labels for these, then add the labeled data to the training set to update the decision boundary (SVM). They note that including diversity of the samples used for labeling would probably be better (rather than biasing and focusing around the decision boundary).

The experiments include binary classification of 7 vs 5 MNIST images, and horse vs automobile CIFAR10 images. They compare their proposed GAAL approach, using a regular GAN, using SVM_{active}, using passive random sampling, using a passive supervised learning and self-taught learning. For the MNIST dataset, they also do out-of-distribution testing on USPS dataset. I believe they did this *after* first testing on MNIST, hoping that the GAN would do better on a different dataset.

For the first set of experiments testing on USPS, their approach was not significantly different than SVM_{active}, nor using "simple GAN" control. When testing on MNIST and on the CIFAR10 dataset, SVM_{active} performed the best. One downside of SVM_{active} is that it sweeps through the entire data set each time it samples data points to be labelled. Finally, they re-did their MNIST test experiments where "every now and then" random samples were chosen to be labelled, rather than those produced by the GAN that are closest to the boundary. This approach out performed GAL. Not that impressive for results, however the idea was interesting.

7 Support Vetor Machine Active Learning with Applications to Text Classification

Simon Tong, Daphne Koller

This is the "SVM_{active}" algorithm mentioned in the GAAL paper. The authors provide theoretical motivations for different SVM-based AL algorithms that aim at reducing the version space. Assumptions are on linearly separable data in the *feature space*, $\mathcal{F}$ (e.g. the reproducing kernel Hilbert space). The authors are interested in doing document classification.

7.1 SVM: Review

Recall that from Mercer's theorem, using a kernel operator K one can map data from the input space $\mathcal{X}$ to some (possibly infinite, although almost surely higher) dimensional RKHS $\mathcal{F}$. The set of classifiers (hypotheses) can thus be represented as

$$f(x) = \Big(\sum_{i=1}^{n} \alpha_i K(x_i, x) \Big)$$

for some finite set of labeled data points $\{x_i\}_{i=1}^n$. Moreover, Mercer's theorem states that $K(u, v) = \langle \Phi(u), \Phi(v) \rangle$, for $\Phi : \mathcal{X} \to \mathcal{F}$. SVM computes the appropriate α's corresponding to the maximal margin hyperplane in $\mathcal{F}$. By using appropriate kernels (e.g. polynomial, radial basis, etc), one can obtain arbitrarily complex separating hyperplanes. The authors assume that the data is normalized, in the sense that $\forall i$, $||\Phi(x_i)|| = \lambda$, for some fixed constant λ. WLOG we can assume $\lambda = 1$.

7.2 Version Space

The authors note that the hypothesis space

$$\mathcal{H} = \Big\{ f \ | f(x) = \frac{\langle w, \Phi(x) \rangle}{||w||} \text{ where } w \in \mathcal{W} \Big\}$$

and the parameter space $\mathcal{W}$ is simply $\mathcal{F}$. The version space, V is defined as:

$$V = \{ f \in \mathcal{H} | \ \forall i \in \{1, 2, ..., n\} \ y_i f(x_i) > 0 \}.$$

The authors note that since $\mathcal{H}$ is a set of hyperplanes that is in bijection between the unit vectors $w \in \mathcal{W}$, they note that

$$V = \{ w \in \mathcal{W} | \ ||w|| = 1, \ y_i f(x_i) > 0 \ \forall i \in \{1, 2, ..., n \ \}.$$

7.3 Duality

The authors note that there is a duality between points in $\mathcal{F}$ and hyperplanes in $\mathcal{W}$, and vice versa. They note that by definition, a point in $\mathcal{W}$ corresponds to a hyperplane in $\mathcal{F}$. They exaplin the converse by observing that a training instance x_i in the feature space restricts the set of possible separating hyperplanes to ones that classify x_i correctly. Any separating hyperplane was satisfy $y_i \langle w, \Phi(x_i) \rangle > 0$. Instead of viewing w as the normal vector of a hyperplane in $\mathcal{F}$, they view $\Phi(x_i)$ as being the normal vector ofa hyperplane in $\mathcal{W}$. Hence, $y_i \langle w, \Phi(x_i) \rangle > 0$ defines a separating hyperplane in $\mathcal{W}$, and the innerproduct is zero at hyperplanes in $\mathcal{W}$ that acts as a boundary on the version space V. Since the version space is defined with respect to unit vectors, it is the surface of some hypersphere, and is connected, in parameter space $\mathcal{W}$.

Recall the SVM optimization procedure follows:

$$\begin{array}{ll} \text{maximize}_w & \min_i\{y_i\langle w, \Phi(x_i)\rangle\} \\ \text{subject to :} & ||w|| = 1; \quad y_i\langle w, \Phi(x_i)\rangle > 0 \ \forall \ i \end{array}$$

Hence, the solution of this is always in V. Given the constraints on $||\Phi(x_i)|| = \lambda$, then each $\frac{\Phi(x_i)}{\lambda}$ is a unit normal vector of a hyperplane in parameter space. Under the constraints mentioned, each of these hyperplanes delimit (e.g. construct the boundary) of the version space. Under this duality interpretation, $y_i\langle w, \Phi(x_i)\rangle > 0$ can be interpreted as a constant multiple (λ) of the distance between the point w and the hyperplane with normal vector $\Phi(x_i)$. Again, under the dual representation, finding w^* in the version space that maximizes the minimum distance to any of the delineating hyperplanes becomes the goal. Equivalently, the SVM finds the center of the largest radius hypersphere whose center can be placed within the version space and whose surface does not interesect with the hyperplanes corresponding to labeled instances (which incidentally are a result of the support vectors).

The radius of the largest sphere that can fit inside V that touches one of the hyperplanes is given by $y_i\frac{\langle w^*, \Phi(x_i)\rangle}{\lambda}$, where $\Phi(x_i)$ is the support vector. Now returning to the primal representation, the distance is $\frac{1}{\lambda}$ times the distance between the support vector and the hypoerplane with normal vector w^*, e.g. the margin of the SVM divided by λ. Hence, the radius of the sphere is proportional to the margin of the SVM.

7.4 Active Learning

The goal of the active learning procedure is to reduce the version space as much as possible. We have seen that this is related to the size of the sphere that can be inscribed within the version space, which is related to the margin of the SVM. The authors define $Area(V)$ as the surface area that the version space V occupies on the hypersphere with $||w|| = 1$. The authors denote V_i the version space after an active learner has made i queries, and upon the $(i+1)$th query, of x_{i+1}, define:

$$\begin{aligned} V_i^- &= V_i \cap \{w \in \mathcal{W}| \ -\langle w, \Phi(x_{i+1})\rangle > 0\} \\ V_i^+ &= V_i \cap \{w \in \mathcal{W}| \ \langle w, \Phi(x_{i+1})\rangle > 0\} \end{aligned}$$

which are the resulting version spaces based on the possible future labelings of x_{i+1}. If one can halve the version space at each iteration, under separability and no noise in the labels, this results in an exponential speed up in learning over passive learning. Though computing V_i^-, V_i^+ is not feasible, the authors introduce three approaches that aim at splitting the version space into equal parts, as much as possible.

Simple Margin. The intuition is that any time, the current w_i is as centrally located within the current version space as possible. By looking at the entire unlabeled dataset one can see how close each data point x_{i+1} the corresponding hyperplanes in $\mathcal{W}$ come to the centrally located w_i. To compute this, it si simply how close $\Phi(x_{i+1}$ is to w_i, which is easily computed by $|\langle w_i, \Phi(x_{i+1})\rangle|$. In effect, this query strategy is the one that simply decides to label the data instance that is closest to the separating margin hyperplane.

MaxMin Margin. Simple margin assumes that the version space is fairly symmetric and that w_i is centrally located, which may not be true. Given i labelled examples, w_i is the center of the largest hypersphere that can fit inside the current version space V_i, with radius m_i, which is proportional to the size of the margin w_i. Using the radius m_i as an indication of the size of the version space, one can take a query instance x and look at the resulting margin if x is given label -1 (m^-), as well as if it is given the label $+1$, (m^+). The goal is to split the version space so that $Area(V^-) \approx Area(V^+)$. Considering $min(Area(V^-) \approx Area(V^+))$: it will be small if the two are quite different. Using $min(m^-, m^+)$ as an approximation, then the following decision rule is made:

$$\max_{x \in Unlabeled} \; min(m^-, m^+)$$

Ratio Margin. This is similar in spirit to MaxMin, however trying to take into account the fact that the current version space may have a non-symmetric shape. In doing so, they look at relative sizes of m^- and m^+, and choose that x that maximizes: $min(\frac{m^-}{m^+}, \frac{m^+}{m^-})$.

Experimental Results. The authors test on Reuters data set as well as Newsgroups dataset. They describe feature representation of documents, processing steps, and note that each document is represented as a roughly 10,000 dimensional vector. They had thousands of test documents. They compare to SVM trained on entire data set, and SVM with randomly sampled subsets (active learning with random selection). They show a huge increase in performance using their approaches over randomly sampling (passive learning). On Reuteres dataset their three approaches were more or less equivalent. Their approaches saved upwards of over 100-fold data to be labeled over random sampling. They noticed that their active learning strategies tended to be balanced in their queries, in the sense that roughly half of the queries were of positive, and half of negative examples for the given classification task. To test that this was important, they augmented the "random" sampling approach to constrain it such that half of its samples were positive instances, half negative, and this showed that the "random balanced" approach, though not as good as their active learning approach, was quite competitive. They argue that this is evidence that their active learning approach is well balanced and samples nicely across both class types. Their experiments each started with a small number of initially

labeled data instances, and from a given pool of unlabeled instances, selected a fixed number to be labeled, and this procedure was repeated iteratively. They next showed that when you increase the pool size (e.g. the number of unlabeled data instances that the SVM active learning algorithm can consider) you see an increase in performance. The authors note, though, that this process is expensive as you must train roughly 2 SVM's for each unlabeled data instance considered. For the newsgroup dataset they saw that the MaxMin margin and Ratio Margin approaches were the best, and the Simple approach did not do as well, though still better than random sampling.

8 Multi-class active learning for image classificaiton

A.J. Joshi, F. Porikli, N. Papanikolopoulos

The authors deal with image classification in a multi-clas setting. They still use SVM's for their approach, and can even handle instances where the number of classes grows and is not known *a priori*. Their approach follows Platt's method of estimating probabilistic outputs from an SVM.

8.1 Probabilistic SVM Output

For a binary classification task, we have

$$p(y = 1|x) = \frac{1}{1 + \exp(Af(x) + B)}$$

where A, B are parameters that are estimed via MLE, where they are the arg mins of the following:

$$-\sum_{i=1}^{\ell}(t_i \log(p_i) + (1 - t_i)\log(1 - p_i)), \text{and}$$

$$p_i = \frac{1}{1 + \exp(Af(x) + B)}$$

and $t_i = \frac{N_p+1}{N_p+2}$ if $y_i = 1$ and $t_i = \frac{1}{N_n+2}$ if $y_i = -1$. Where N_p is the number of positive examples, and N_n defined similarly. The authors use one-vs-one approach to SVM multi-class classification. They assume that the binary class probabilities can be obtained as above, then assume that r_{ij} are the binary proabilities estimating $P(y = i|i \text{ or } j, x)$. Then the probability of class i is p_i and
$\min_p \frac{1}{2} \sum_{i=1}^{k} \sum_{j,j \neq i} (r_{ji} p_i - r_{ii} p_j)^2$
subject to $\sum_i p_i = 1$, $p_i \geq 0$, $\forall i$.

The above is a convex problem that can be solved using Gaussian elimination or a simple iterative algorithm. They use LIBSVM. The authors use two approaches to do AL. First is entropy measure: $H(Y) = -\sum_{i=1}^{k} p_i \log(p_i)$. However, they note that this places too much mass to "unimportant" classes. They also use the "Best versus Second Best" approach (margin).

The authors experiment n Caltech-101 dataset and another with 13 natural secene categories. (Pendigits, USPS, Letter datasets). The authors use linear, polynomial and RBF kernels. They find that BvSB approaches are best and can reduce up to 53 percent of the data required versus SL. They also visualize some examples of unlabeled examples selected vs not selected for AL (images) and show that the ones queried for AL were ambiguous.

9 Combining Active Learning and Semi-Supervised Learning using Gaussian Fields and Harmonic Functions

X. Zhu J. Lafferty, Zoubin Ghahramani

This paper uses the manifold regularized SSL approach. A kernel is used to measure distance of all the data, and labels are propagated on this Gaussian field. Weights of the graph are the distances computed between all data. Active learning is performed on top of the SSL approach by greedily selecting queries from the unlabeled data to minimize the estimated expected classification error. This can all be done efficiently using matrix methods.

9.1 Gaussian random fields and harmonic energy minimizing functions

Assume $(x_1, y_1), ..., (x_l, y_l)$ ℓ labeled data, and $\ell + 1, ..., \ell + u$ unlabeled data. For a total of $n = \ell + u$ data points. We use a metric (e.g. RBF) and compute weights $w_{ij} = \exp\Big(- \frac{1}{\sigma^2} ||x_i - x_j||_2 \Big)$. Scale parameter σ is learned or a hyperparameter. We maintain the labels for the labeled data $y \in \{0, 1\}$, and for the unlabeled data we predict a mapping $f : V \to [0, 1]$, where V is the vertex set of unlabeled data in our newly constructed weighted graph.

The energy $E(y) = \frac{1}{2} \sum_{i,j} w_{i,j}(y_i - y_j)^2$, and seek to minimize energy of our graph. Hence, we assume that nearby points on the graph should have the same labels. Let $D = \text{diag}(d_i)$, with $d_i = \sum_j w_{ij}$, and the combinatorial Laplacian is the $n \times n$ matrix $\Delta = D - W$, where W is the weight matrix. We can rewrite the energy function as $E(y) = y^t \Delta y$. The Gaussian random field is $p(y) = \frac{1}{Z_\beta} \exp(-\beta E(y))$.

Hence, we turn the binary classification problem into a regression over $[0, 1]$. The probability distribution over unlabeled nodes is Gaussian with covariance matrix $\frac{1}{\beta}\Delta_{uu}^{-1}$. Δ_{uu} is the submatrix of Δ corresponding to unlabeled data. As it turns out, the function f that minimizes the energy function $E(y)$ is harmonic and satisfies $\Delta f = 0$, and $f(j) = \frac{1}{d_j}\sum_{i\sim j} w_{ij} f(i)$, which means it is simply the mean of its neighbours.

Δ can be broken down into block matrix form by Δ_{ll}, Δ_{lu}, Δ_{ul}, Δuu, and $f = [f_l \;\; f_u]^t$, where f_l is simply the true labels on the labeled data set. It turns out that $f_u = -\Delta_{uu}^{-1}\Delta_{ul} f_l$, and $y_u \sim \mathcal{N}(f_u, \Delta_{uu}^{=1})$. Bayes classifier would predict the label of node i as class 1 if $f(i) > 0.5$ and 0 else.

9.2 Active Learning

The risk $\mathcal{R}(f) = \sum_{i=1}^{n}\sum_{y_i=0,1}[sgn(f_i) \neq y_i]p^*(y_i|L)$, where p^* is the true distribution. We do not have access to this, so the authors state that under this model, an appropriate approximation to the risk:

$$\begin{aligned}\hat{\mathcal{R}}(f) &= \sum_{i=1}^{n}[sgn(f_i) \neq 0](1 - f_i) + [sgn(f_i) \neq 1]f_i \\ &= \sum_{i=1}^{n} min(f_i, 1 - f_i)\end{aligned}$$

After doing a round of AL, we will get a label, re-compute the Harmonic field, denoted $f^{+(x_k,y_k)}$, and the estimated risk will also change:

$$\hat{\mathcal{R}}(f^{+(x_k,y_k)}) = \sum_{i=1}^{n} min(f_i^{+(x_k,y_k)}, 1 - f_i^{+(x_k,y_k)})$$

Since before we do a round of AL we do not know what label we receive, we can compute the expected risk:

$$\hat{\mathcal{R}}(f^{+x_k}) = (1 - f_k)\hat{\mathcal{R}}(f^{+(x_k,0)}) + f_k\hat{\mathcal{R}}(f^{+,(x_k,1)})$$

The AL algorithm selects the data point x_k that minimizes the expected Risk. In order to do this, $f^{+(x_k,y_k)}$ needs to be computed after adding the oracle labeled (x_k, y_k). This involves re-trainining the dataset, which is expensive. The authors note that this can be computed cheaply in closed form, recalling that $f_u = -\Delta_{uu}^{-1}\Delta_{ul} f_l$. They show in the appendix that $f_u^{+(x_k,y_k)}$ can be computed:

$$f_u^{+(x_k,y_k)} = f_u + (y_k - f_k)\frac{(\Delta_{uu}^{-1})_k}{(\Delta_{uu}^{-1})_{kk}}$$

where $(\Delta_{uu}^{-1})_k$ is the k-th column of the inverse Laplacian on unlabeled data, and $(\Delta_{uu}^{-1})_{kk}$ is the k-th diagonal element of the same matrix.

The authors do some experiments on synthetic data, and on 20 newsgroup (text) data. They explain how they generate features, and the graph (similarity metric). They compare their querying strategy with some other baselines. They also do experiments on handwritten digit dataset

10 Deep Bayesian Active Learning with Image Data

Yarin Gal, R. Islam, Zoubin Ghahramani

Gal combines the Bayesian NN work of his thesis and applies this to uncertainty measurements also presented in his thesis as mechanisms for data selection for labeling in AL. Specifically they use Bayesian CNN's, which simply apply Dropout after every convolutional layer. Training these networks is no different than training any network with dropout. The difference is at test time, dropout is still used, and T stochastic forward passes are made. These forward passes are then used to compute uncertainty measurements, and is called *MC Dropout*.

This paper, and prior work by Gal and Ghahramani introduce Dropout as approximately equivalent to variational inference in Bayesian settings (e.g. Bayesian neural nets, Gaussian Processes, etc). I leave the reader to find these works, and advise them to consult Gal's phd thesis, specifically chapter 3. Essentially, the random nature of the Bernoulli distribution random variable masks can be combined with weights to represent the weights as random variables. Hence, approximating the posterior can be done:

$$p(y = c|x, \mathcal{D}_{train}) \approx \frac{1}{T} \sum_{t=1}^{T} p(y = c|x, \hat{\omega}_t)$$

where ω is the random weights of the network, and $\hat{\omega}$ is the realized weights. For further details and a more thorough look at Bayesian NN and Bayesian CNN, I am preparing a separate document on modeling uncertainty in machine learning.

10.1 Active Learning

Gal uses 4 measures of uncertainty, plus random sampling to compare AL strategies. These are also covered in his PhD thesis.

Maximum Entropy. Here the point that maximizes
$\mathbb{H}[y|x, \mathcal{D}_{train}] = -\sum_c p(y = c|x, \mathcal{D}_{train}]\log p(y = c|x, \mathcal{D}_{train})$.

Mutual Information. Here the point that is expected to maximize the information gained about the model parameters (e.g. maximize the mutual information between predictions and model posterior):

$$
\begin{aligned}
&\mathcal{I}[y, \omega | x, \mathcal{D}_{train}] \\
&= \mathbb{H}[y|x, \mathcal{D}_{train}] - \mathbb{E}_{p(\omega|\mathcal{D}_{train})}[\mathbb{H}[y|x, \omega]] \\
&= -\sum_c p(y = c|x, \mathcal{D}_{train}] \text{log} p(y = c|x, \mathcal{D}_{train}) + \mathbb{E}_{p(\omega|\mathcal{D}_{train})}[\sum_c p(y = c|x, \omega) \text{log} p(y = c|x, \omega)] \\
&\approx -\sum_c \Big[\Big(\frac{1}{T} \sum_t p(y = c|x, \hat{\omega}_t) \Big) \text{log} \Big(\frac{1}{T} \sum_t p(y = c|x, \hat{\omega}_t) \Big) \Big] + \frac{1}{T} \sum_t \sum_c \Big(p(y = c|x, \hat{\omega}_t) \text{log} p(y = c|x, \hat{\omega}_t \Big)
\end{aligned}
$$

Maximize Variation Ratios. Variation ratio (VR):

$$
\begin{aligned}
\text{VR}(x) &= 1 - \text{max}_y p(y|x, \mathcal{D}_{train}) \\
&= 1 - \text{max}_y \frac{1}{T} \sum_{t=1}^{T} p(y = c|x, \hat{\omega}_t)
\end{aligned}
$$

Note that both Max Entropy and Max Variation Ratios lack confidence measures.

Maximize Mean Standard Deviation. Here the authors compute the standard deviation of the posterior predictions for each class, then take the mean across the C classes. Using the notation of $f^{\hat{\omega}_t}(x)$ to denote the stochastic output of a Bayesian NN for iteration $t \in [T]$ of the MC dropout forward passes, we have the estimated (unbiased estimator) posterior predictive second moment:

$$
\mathbb{E}\Big[(y^t)(y)] = \tau^{-1} I + \frac{1}{T} \sum_t f^{\hat{\omega}_t}(x)^T f^{\hat{\omega}_t}(x)
$$

where the superscript T is to denote the transpose, not the indexing, and

$$
\hat{Var}[y] = \tau^{-1} I + \frac{1}{T} \sum_t f^{\hat{\omega}_t}(x)^T f^{\hat{\omega}_t}(x) - \Big(\frac{1}{T} \sum_t f^{\hat{\omega}_t}(x) \Big)^T \Big(\frac{1}{T} \sum_t f^{\hat{\omega}_t}(x) \Big)
$$

where τ is a precision parameter of your model. They also consider random sampling. For MNIST experiments, they found that the first three approaches worked best, with Mean STD and random sampling being equal.. They also showed that computing the three approaches (Var Ratio, entropy, mutual information) with a Bayesian CNN vs. CNN was dramatically better for a Bayesian CNN. Very positive results.