Active learning and semi-supervised learning

How to learn (or not) from unlabelled data

Brooks Paige

Supervised learning with limited labels

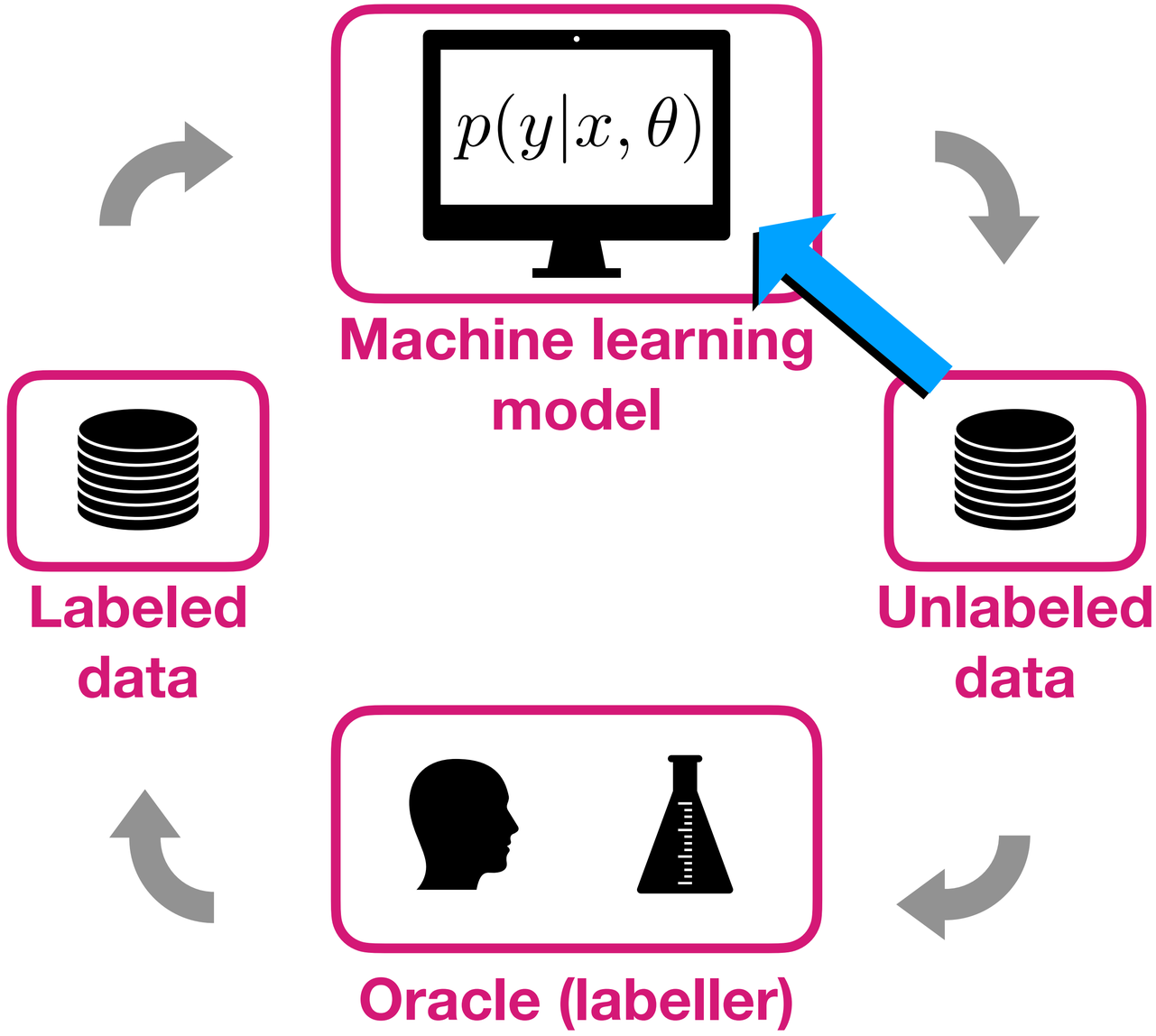
- Setting: lots of inputs x, not a lot of labels y (common real-world scenario)
- Going to focus on classification case: learn p(y | x)
- **Labeled data:** $\mathcal{D}_0 = (x_1, y_1), \ldots, (x_M, y_M)$
- **Unlabeled "pool":** $\mathcal{D}_p = x_1, \ldots, x_N$
- N >> M (not many labels!)

Problem setting

Active learning

- Have a pool of labeled pairs
- Have a pool of unlabeled inputs
- Fit a model to the labeled data
- Use the model to decide which unlabeled point(s) to label next
- Assumes there is a labeler in-the-loop, and a fixed label budget which we want to use efficiently

Question: **What about that unlabeled data??**



Common algorithms for active learning

General principals

- **Diversity**: select inputs which are far away from the labeled points
- **Uncertainty**: select inputs for which the classifier is very uncertain
- Quite a bit of current interest in scaling:
  - … to large “batch” sizes (i.e. selecting multiple next points to test at once)
  - … to deep learning models

Example of positive results

Citovsky et al., NeurIPS 2021

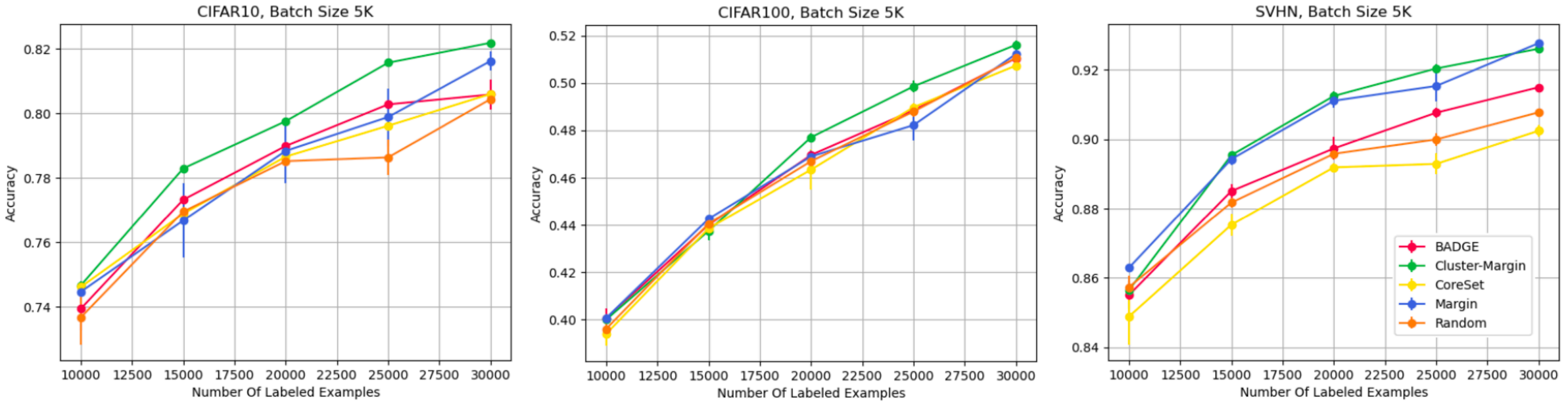
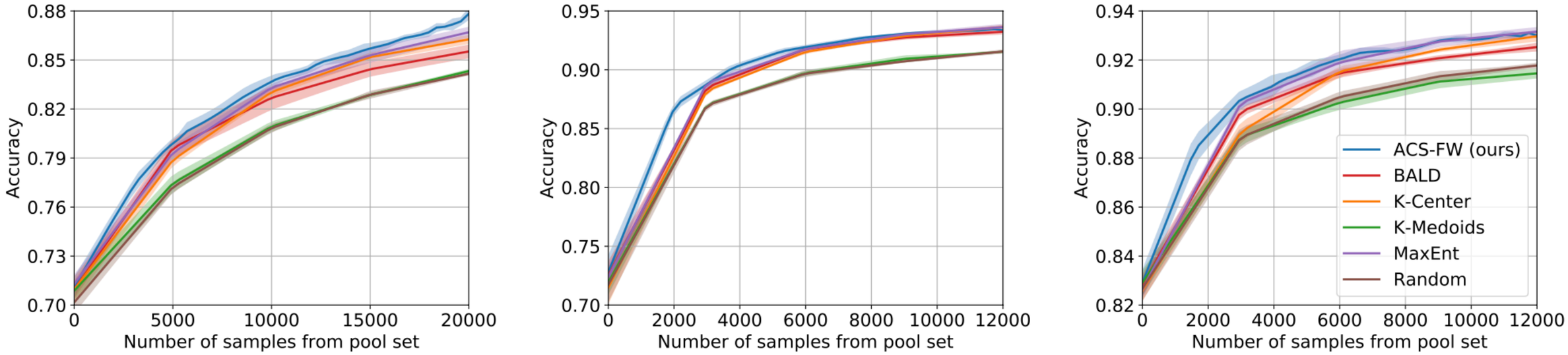


Figure 3: Accuracy of various active learning method as a function of the number of labeled examples, using active learning batch sizes of 5K. The mean and standard error as computed across 10 trials is shown.

Example of positive results

Pinsler et al., NeurIPS 2019



(a) cifar10 (b) SVHN (c) Fashion MNIST

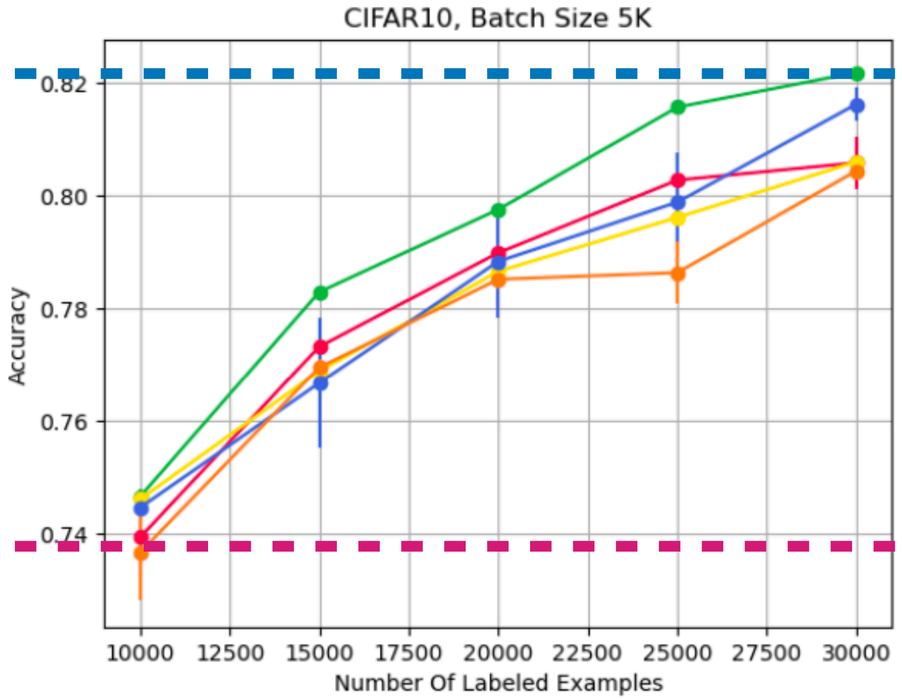
Figure 4: Test accuracy on classification tasks over 5 seeds. Error bars denote two standard errors.

What do these all have in common?

- Start with some initial number of labelled images
- Algorithmically select next images to label
- Observe that randomly selecting which images to label does not give as much improvement in test accuracy as using the proposed algorithm
- Both “random” and “active” have the same initial performance (they start with the same labels) and eventually converge (if you label the entire pool), but with large benefits particularly in the first handful of labels collected

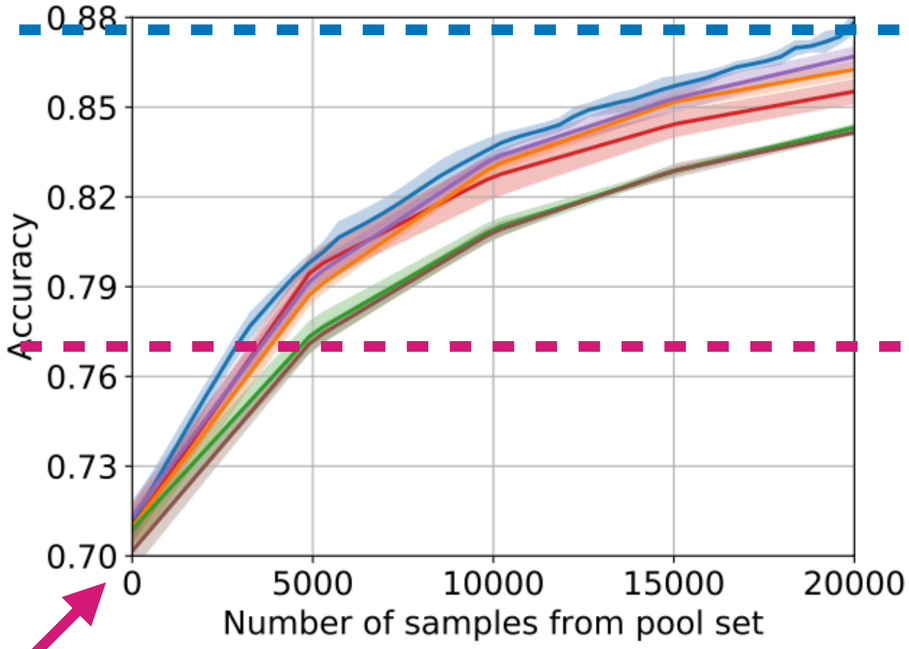
How much data are we actually using?

How “good” is CIFAR10 performance?



+20k AL:
83%–87%

10k, random:
74%–77%



Citovsky et al., 2021

Initial pool size:
5k images

Pinsler et al., 2019

[Note: architectures are not the same]

Two generally “unrelated” problems

Active learning

- Have a pool of labeled pairs
- Have a pool of unlabeled inputs
- Fit a model to the labeled data
- Use the model to decide which unlabeled point(s) to label next
- Assumes there is a labeler in-the-loop, and a fixed label budget which we want to use efficiently

Semi-supervised learning

- Have a pool of labeled pairs
- Have a pool of unlabeled inputs
- Fit a model to both the labeled and unlabeled data
- … Done!
- Does not assume that collecting any additional labels is possible

General approaches for semi-supervised learning

- Basically: loss = “labeled data loss” + “unsupervised regularizer”
- Pseudo-label methods (e.g. Sohn et al. 2020): for points where our classifier is very “confident” about the labels, add the label to the training set
  - Regularizer takes the form of a classification loss on predicted labels, for the subset where we are already highly confident
- Generative approach (e.g. Kingma et al. 2014): learn the joint distribution of p(data, label) and perform inference over the missing labels
  - Regularizer is the ELBO of a variational autoencoder

That sounds useful — does it work?

Sohn et al., NeurIPS 2020

Figure 2: FixMatch reaches 78% CIFAR-10 accuracy using only above 10 labeled images.

- Reminder! Before we had:

  10k random labels: 74%–77% accuracy

  25k-30k labels, 20k selected via active learning: 83%–87% accuracy

Semi-supervised learning performance

Sohn et al., NeurIPS 2020

	CIFAR-10			CIFAR-100			SVHN		
Method	40 labels	250 labels	4000 labels	400 labels	2500 labels	10000 labels	40 labels	250 labels	1000 labels
Π-Model	-	54.26±3.97	14.01±0.38	-	57.25±0.48	37.88±0.11	-	18.96±1.92	7.54±0.36
Pseudo-Labeling	-	49.78±0.43	16.09±0.28	-	57.38±0.46	36.21±0.19	-	20.21±1.09	9.94±0.61
Mean Teacher	-	32.32±2.30	9.19±0.19	-	53.91±0.57	35.83±0.24	-	3.57±0.11	3.42±0.07
MixMatch	47.54±11.50	11.05±0.86	6.42±0.10	67.61±1.32	39.94±0.37	28.31±0.33	42.55±14.53	3.98±0.23	3.50±0.28
UDA	29.05±5.93	8.82±1.08	4.88±0.18	59.28±0.88	33.13±0.22	24.50±0.25	52.63±20.51	5.69±2.76	**2.46**±0.24
ReMixMatch	**19.10**±9.64	**5.44**±0.05	4.72±0.13	**44.28**±2.06	**27.43**±0.31	**23.03**±0.56	**3.34**±0.20	**2.92**±0.48	2.65±0.08
FixMatch (RA)	**13.81**±3.37	**5.07**±0.65	**4.26**±0.05	48.85±1.75	28.29±0.11	**22.60**±0.12	**3.96**±2.17	**2.48**±0.38	**2.28**±0.11
FixMatch (CTA)	**11.39**±3.35	**5.07**±0.33	**4.31**±0.15	49.95±3.01	28.64±0.24	23.18±0.11	7.65±7.65	**2.64**±0.64	**2.36**±0.19

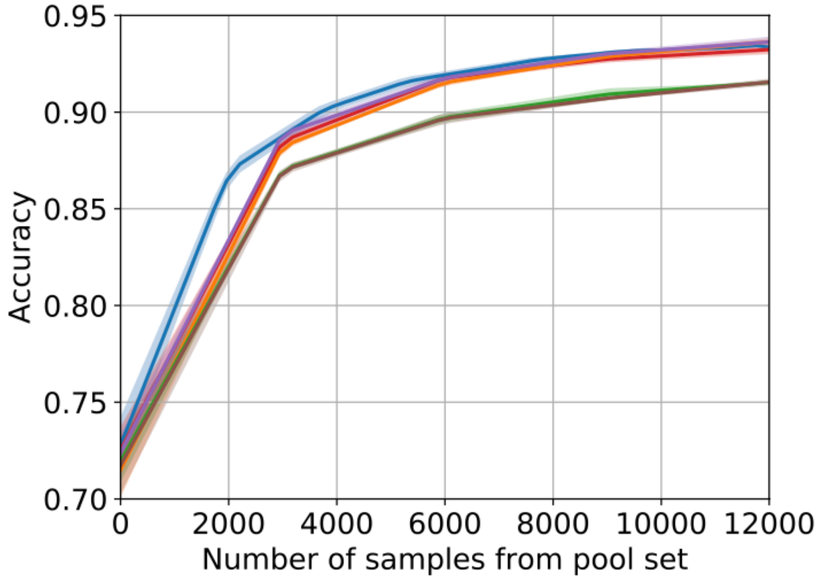
Will come back to the algorithm later!

Point for now: this makes very efficient use of unlabeled data…

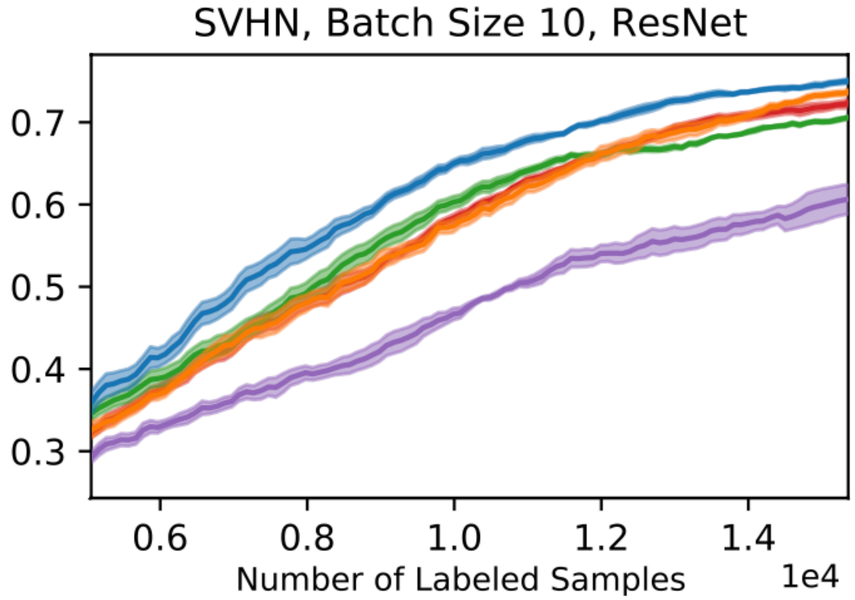
Active learning performance (+SSL baseline)

Tried to compare these on SVHN…

- Matched architectures (resnet18); No data augmentation on labeled set
- Got **>96% accuracy on SVHN with 1000 initial labels**



**Pinsler et al.,
NeurIPS 2019**



**Ash et al.,
NeurIPS 2019**

What gives?

- Huge performance increases possible from semi-supervised learning … moderate gains from using active learning?
- If you have lots of unlabeled data, does it change whether it is worth "paying" for any new labels?
- Can these just trivially be combined? Or does knowledge of where the unlabeled points of located, fundamentally affect which label we should query next?
- Not much work looking at solving these problems jointly! (Exception: Rottmann et al., 2018)

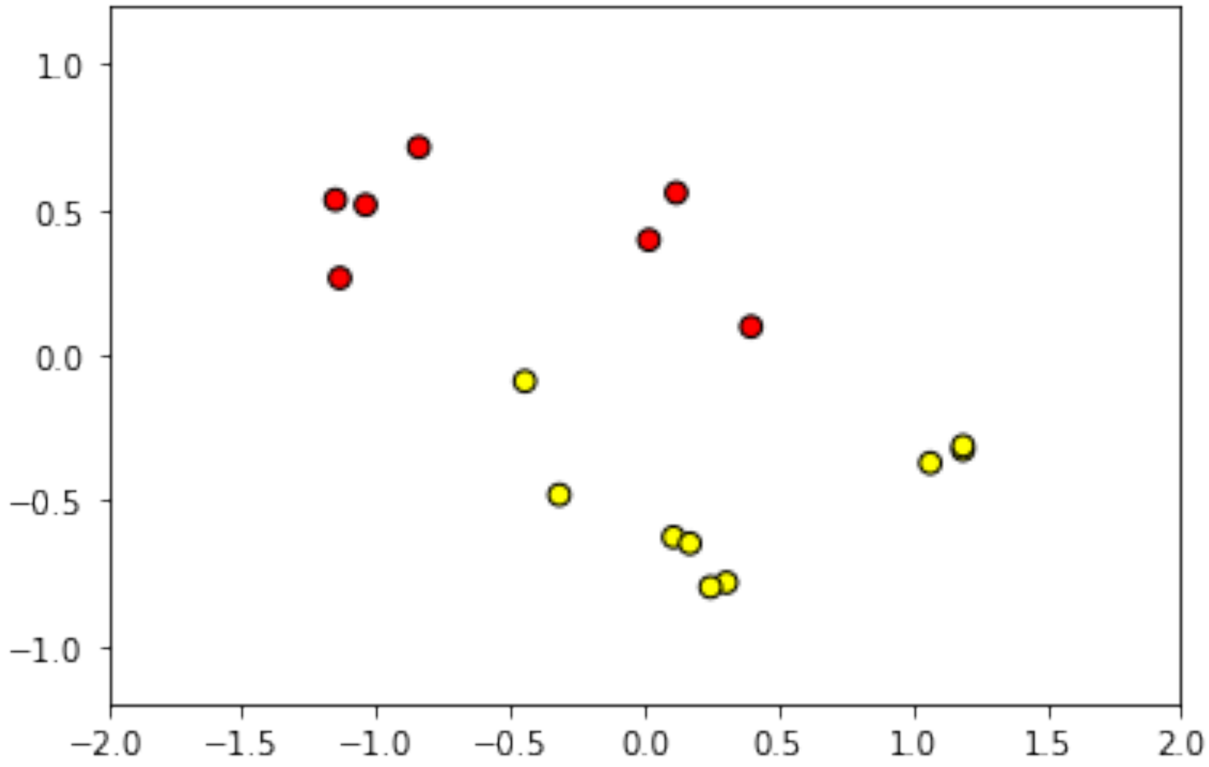
Cold-start problem

Tran et al., ICML 2019

with ResNet18 (He et al., 2016a) and ResNet18pa (He et al., 2016b), which have shown to produce competitive classification results in several tasks. The sample acquisition setup for each data set is: 1) the number of samples in the initial training set is 1,000 for MNIST, 5,000 for CIFAR-10, 15,000 for CIFAR-100, and 10,000 for SVHN (the initial data set percentage was empirically set – with values below these amounts, we could not make the training process converge); 2) the number of acquisition iterations is 150 (50 for SVHN), where at each iteration 100 (500 for SVHN)

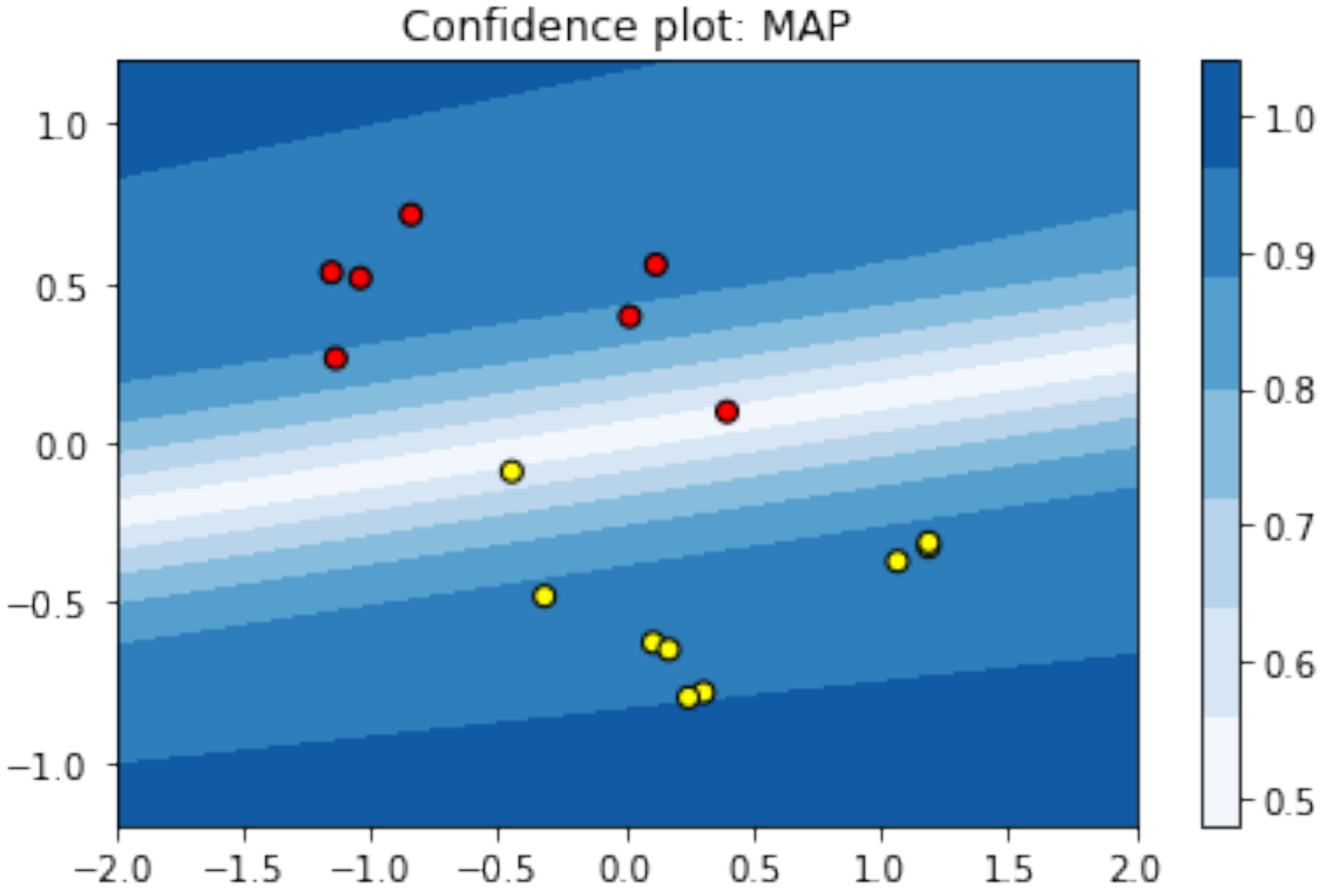
Does this matter?

Does using a semi-supervised classifier change the decision?



Fitting a classifier

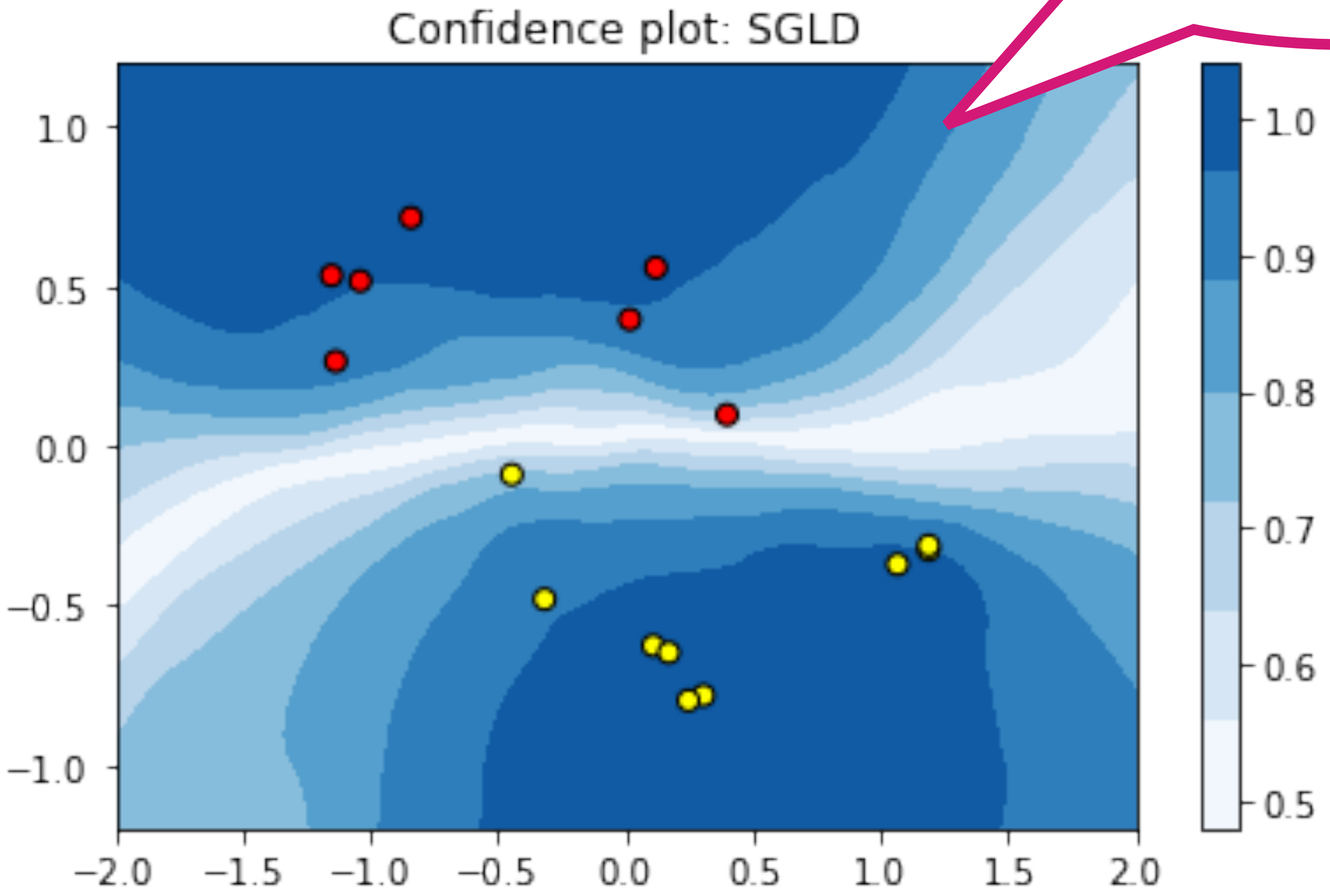
Where are we uncertain?



Fitting a classifier

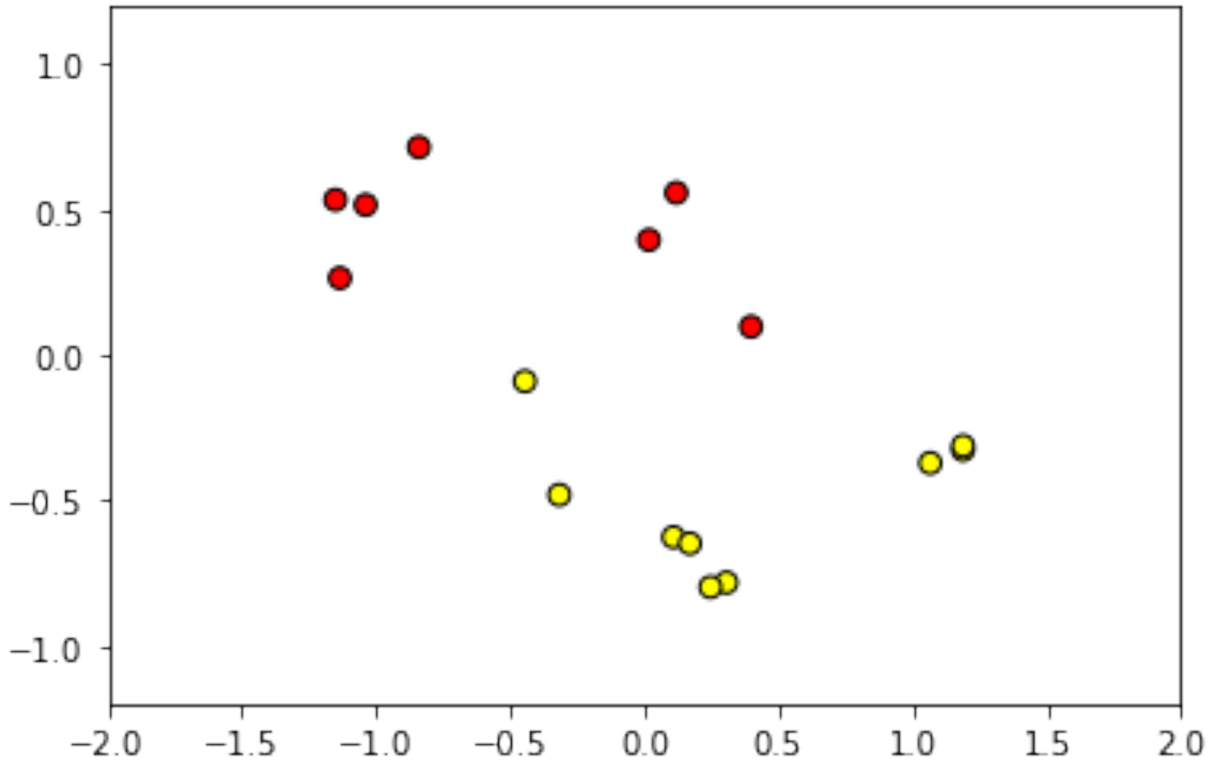
Where are we uncertain?

Looks good!
Time for active learning!

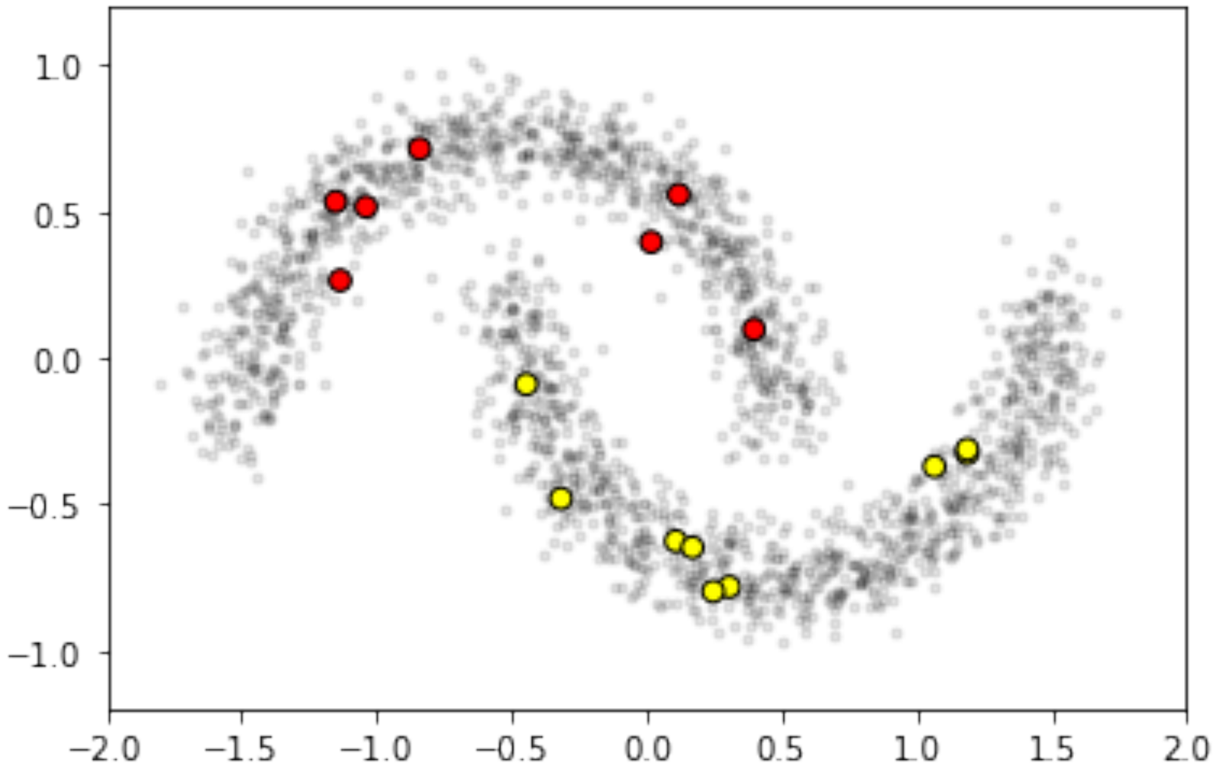


Time for active learning

Note: I haven’t shown you the pool of unlabeled data yet!

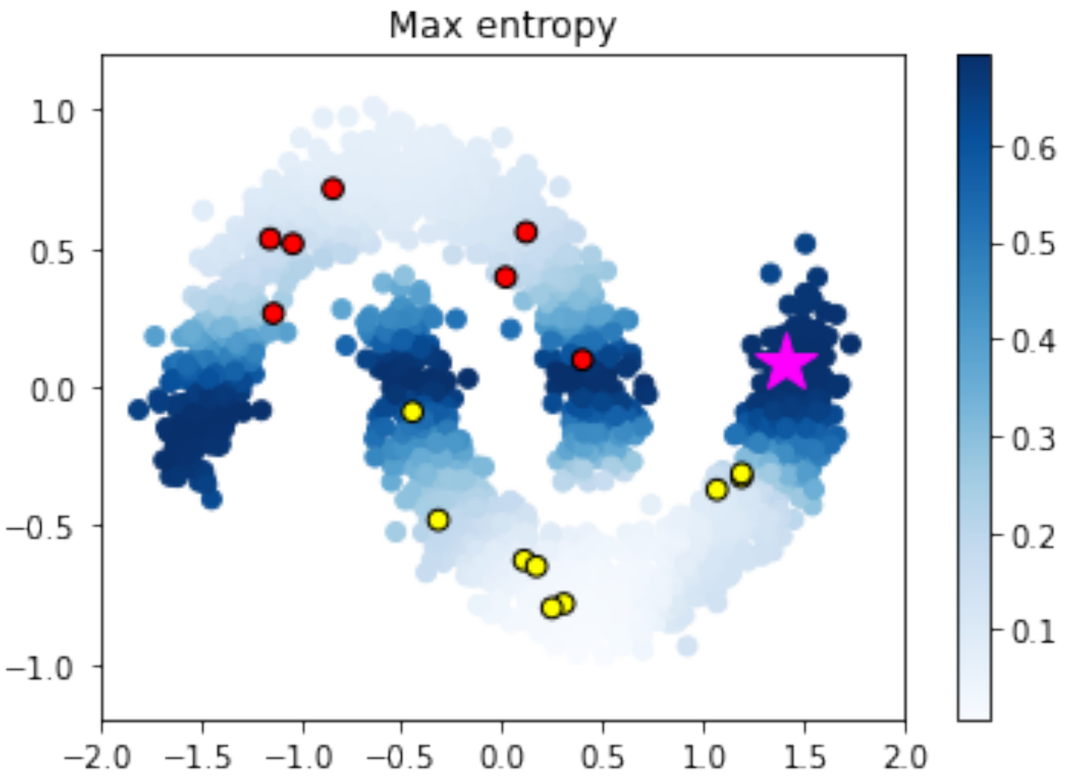


Does this still seem reasonable?

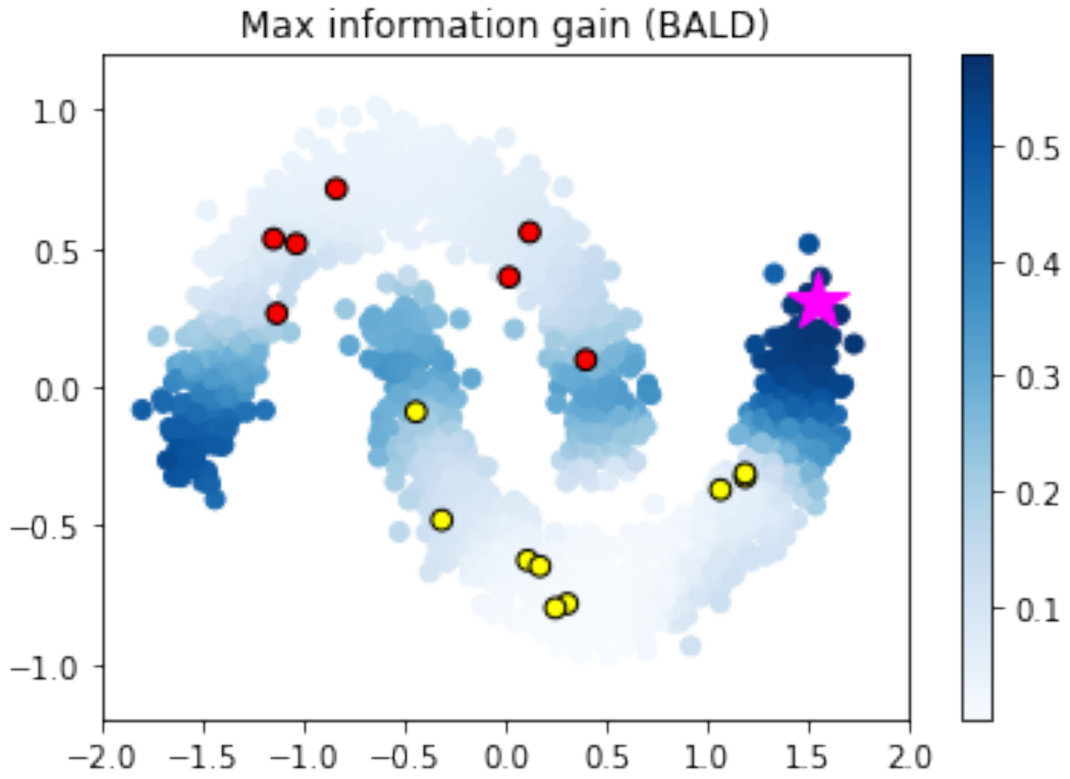


What points would active learning select?

Two example acquisition functions



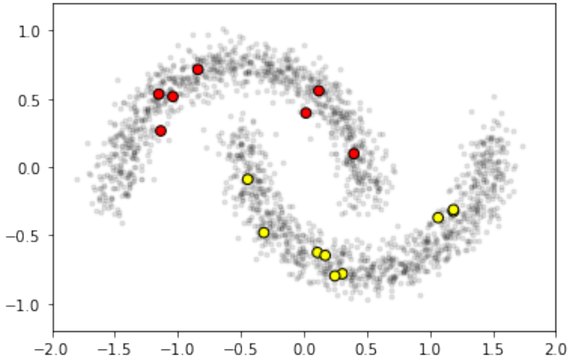
Darker color: higher entropy

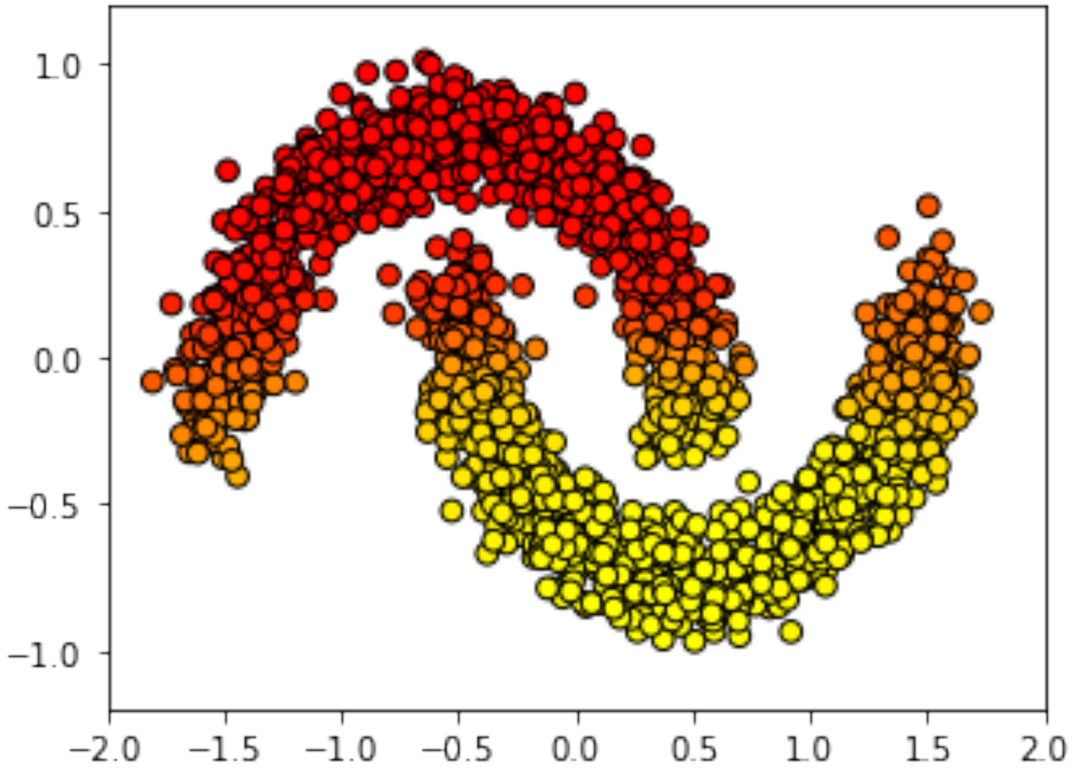


Darker color: higher BALD score

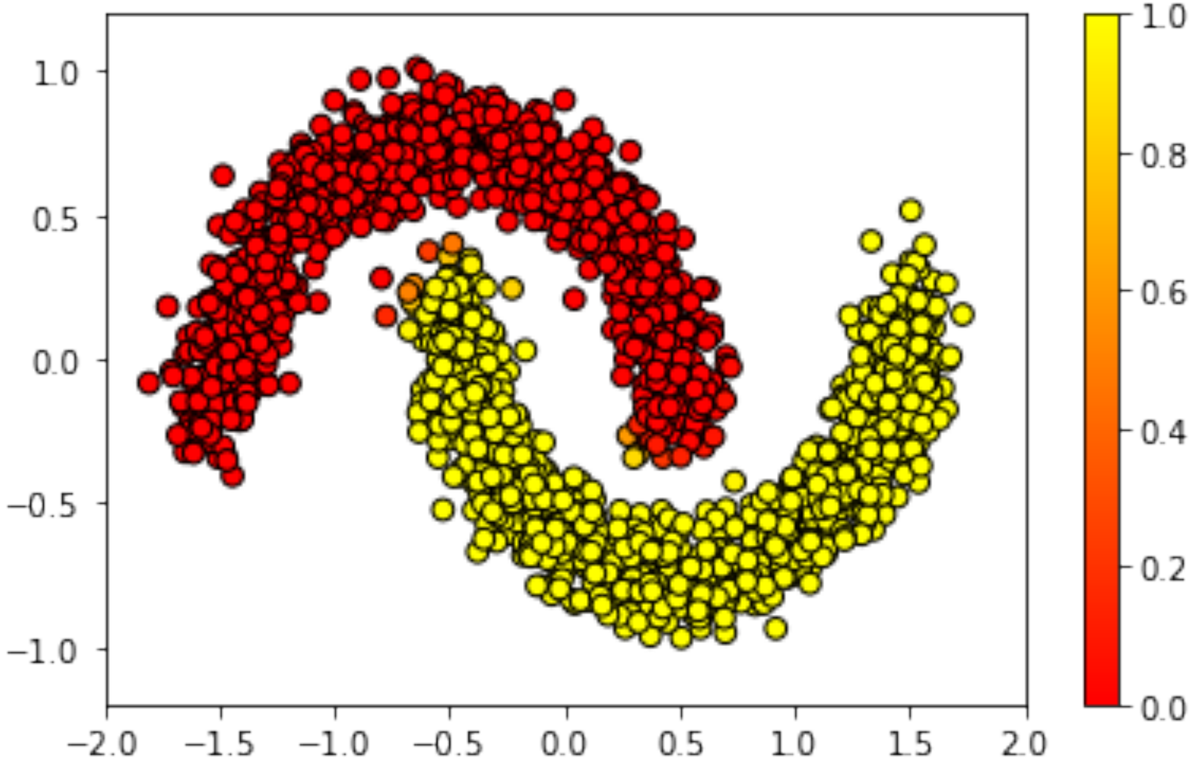
Semi-supervised learner

Generative approach, learning p(data | label)





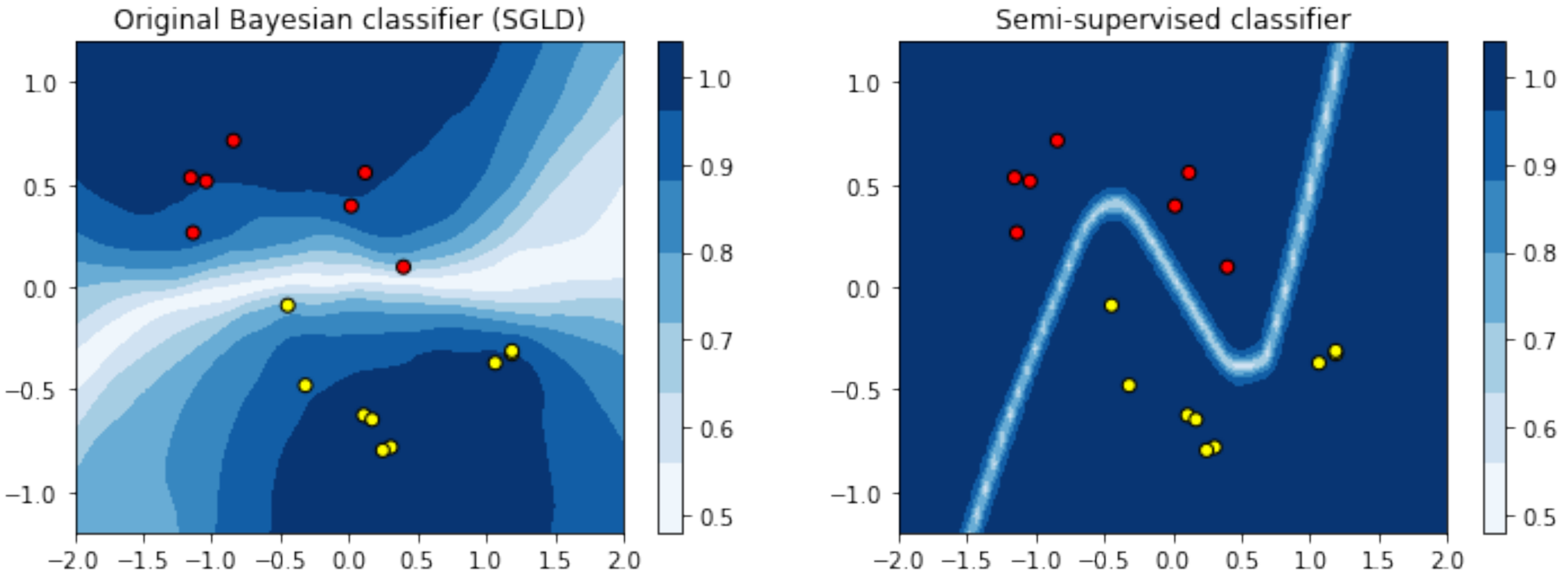
Original classifier:
86.7% accuracy



Semi-supervised classifier:
99.7% accuracy

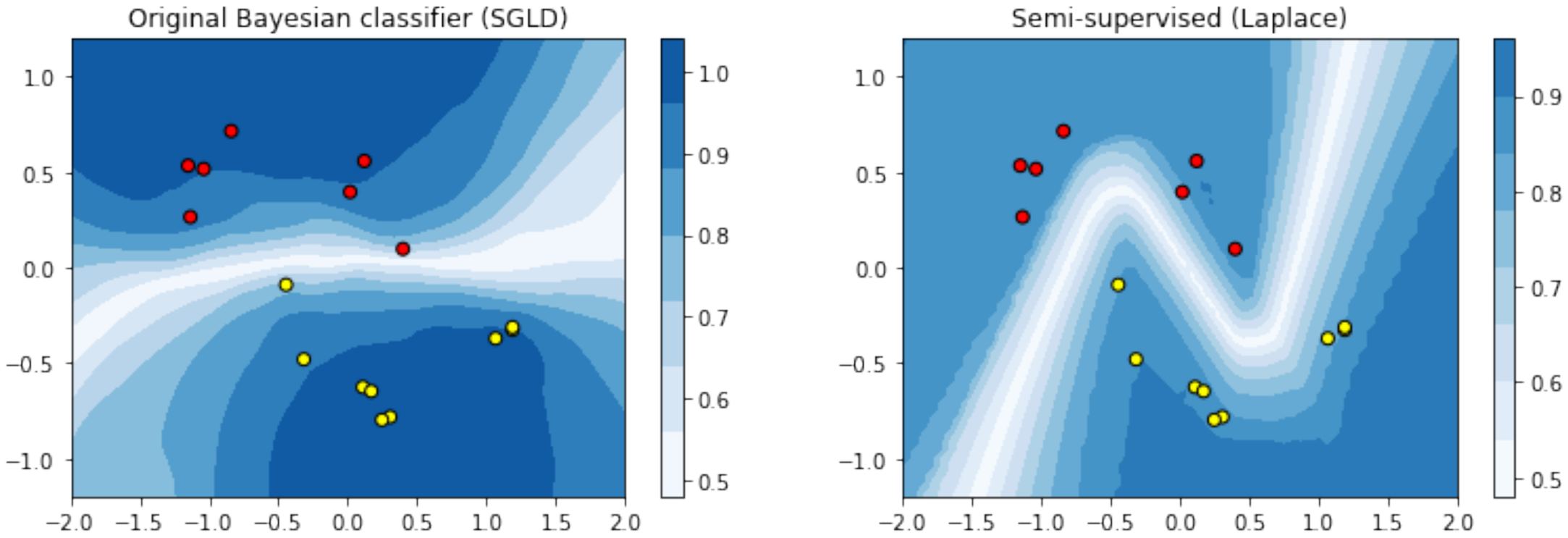
Semi-supervised learner

Note: not “Bayesian” (what would it mean?)



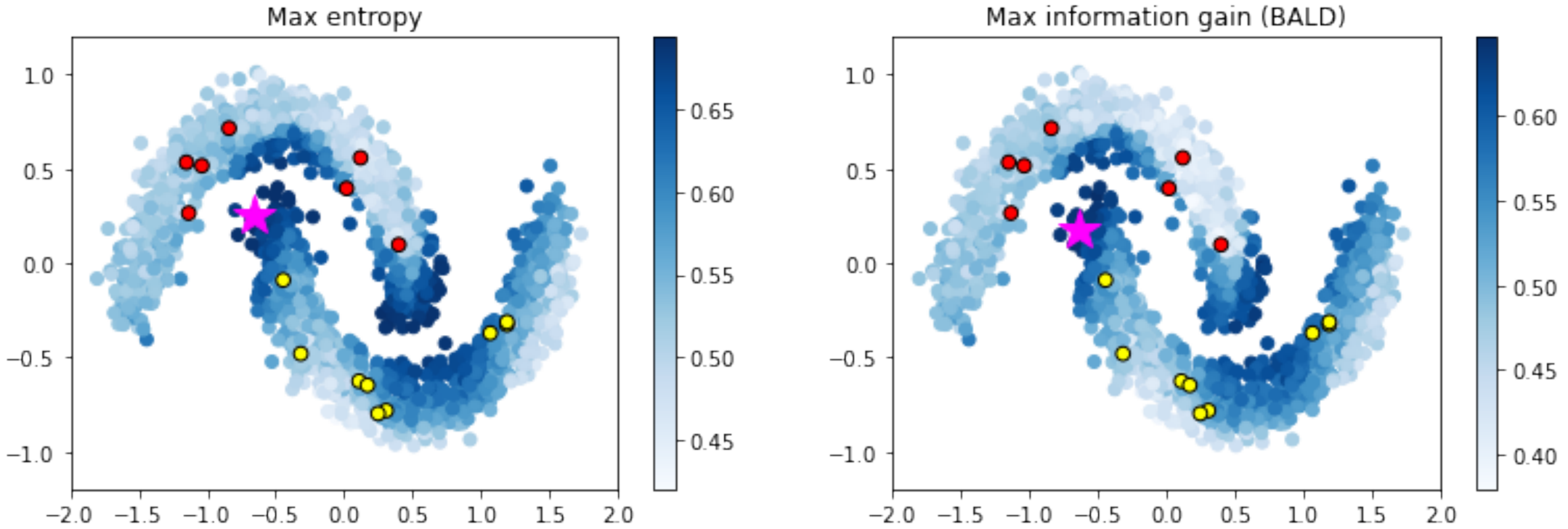
Semi-supervised learner

Quick hack for this presentation: Laplace



Semi-supervised active learning

“Important” points are where we expect



Interpreting uncertainty in SSL

Kingma et al., 2014

- Without going into too much detail, objective involves a classifier loss, as well as ELBOs for learning a deep generative model:

$$\sum_{x,y \in \mathcal{D}_0} \log p(y|x,\theta) + \lambda_1 \sum_{x,y \in \mathcal{D}_0} \mathcal{L}^{\text{sup}}(x,y,\phi) + \lambda_2 \sum_{x \in \mathcal{D}_p} \mathcal{L}^{\text{unsup}}(x,\theta,\phi)$$

- This involves “learning” the regularizer, parameterized by ϕ, which here is taken to mean all encoder and decoder parameters aside from the classifier parameters θ

- Can (or should) this be interpreted as a sort of regularizer on the classifier parameters?

Interpreting uncertainty in SSL

Sohn et al., 2020

- Objective function:

$$\sum_{x,y \in \mathcal{D}_0} \log p(y|x,\theta) + \lambda \sum_{x \in \mathcal{D}_p} \mathbb{I}[\max_y q(y|x,\theta) > \tau] \log q(y|x,\theta)$$

- Interpretation: include “pseudo-observation” if probability of predicted class is greater than τ (e.g. 0.95)

- The distribution $q(y \mid x, \theta)$ is defined in terms of perturbations on the inputs x, through integrating out data augmentations

$$q(y|x,\theta) = \int p(y|x',\theta)\kappa(x'|x)\mathrm{d}x'$$

Predictive uncertainty and data augmentation

Integrating out RandAugment: $q(y|x,\theta) = \int p(y|x',\theta)\kappa(x'|x)\mathrm{d}x'$

$\kappa(x'|x)$:

- Aleatoric uncertainty: irreducible noise in data or measurement process
- Epistemic uncertainty: reducible uncertainty about model & parameters
- What is this? Does “robustness to arbitrary transformations”, or integrating over noisy inputs, have anything to do with uncertainty?

Thoughts and questions

- What uncertainties do we need for deep active learning?
- It seems clear that our predictions, and probably also our uncertainty estimates, need to depend on **all** the data, including the unlabeled pool
- What is a sensible Bayesian interpretation of semi-supervised learning? How should we think about posterior uncertainties?
- Will we need new algorithms for semi-supervised active learning? Or do we just apply existing active learning algorithms to semi-supervised classifiers?
- Points selected by active learning will always depend on the model being used: is there any way to be sure we are “safe”?
- Are image datasets really where we should be benchmarking these algorithms…?

Side-note: Laplace didn’t work here…

Where are we uncertain?