Lessons learned
Building a highly-specialized Image Recognition from scratch

Antoine Labatie
Lead Computer Vision Engineer @ PEAT

Berlin AI 8th gathering – 23.05.2018

I. Context

II. Imperfect sampling

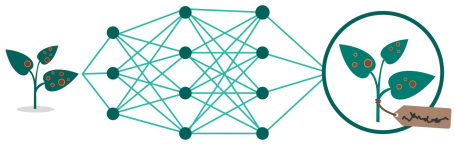
III. Data bottleneck

Starting point: recognizing crop diseases

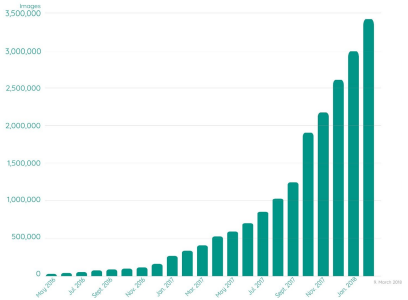
- Important challenge for humanity:
  - Food production must increase by 70% by 2050 to feed 9 billion people
  - Currently, FAO states 15-30% of crop harvests are lost due to diseases and pests each year

- PEAT contribution:
  - Plantix App is an AI system currently able to recognize over 290 diseases on crops
  - It offers Image Recognition and guidance to small-scale farmers

plantix
grow smart.



- Current statistics:
  - 350k monthly active users
  - Database > 4M images (unfortunately not fully labeled)



Machine Learning formulation

- At first sight, it's a **classification** problem:
  - Each disease corresponds to a particular class
  - We add an additional class "Healthy"

- However:
  - A lot of times, crops suffer from multiple infections
  - Some diseases are nearly indistinguishable

- Solution with **multi-label classification**:
  - Output (.., 0, 1, 0, .., 0, 1, 0, …) with multiple 1's when multiple diseases are present
  - Output (.., 0, p_1, 0, .., 0, p_2, 0, …) with probabilities $0< p_i<1$ and $\sum p_i = 1$, when a disease belongs to an indistinguishable group


Powdery Mildew


Leaf-Miner Flies


Pear Rust


Shothole Disease


Leaf Miner Fly & Citrus Canker

Building a highly-specialized Image Recognition from scratch

- Challenge of starting from scratch:
  - How to get initial data?
  - How to collect images most cost-effectively?
  - How to label images most cost-effectively?
  - How to monitor current Image Recognition to guide data collection and labeling?

- Challenge of a highly-specialized and difficult task:
  - Experts are hard to train
  - Labeling is a long process: it requires several steps or several experts and majority voting
  - Labeling cannot be crowdsourced

I. Context

II. Imperfect sampling

III. Data bottleneck

Imperfect sampling

- Usual assumption of ML problems: *samples are **i.i.d = independent identically distributed***
  - Independent: it makes sense to average over samples
  - Identically distributed: it makes sense to use the same loss for all samples

- Good approximation when:
  - data comes from a wide range of context (e.g. from a large number of users)
  - each context only provides a limited number of samples
  - sampling is done in the same way at data collection and at deploy time

- But what if **sampling is imperfect**?
  - Samples are not independent
  - There are distortions in the data distribution (e.g. training data is slightly different from deploy time)

- Which are the guidelines in this case?
  - …hard to find
  - In the litterature, the i.i.d. hypothesis is taken the vast majority of time

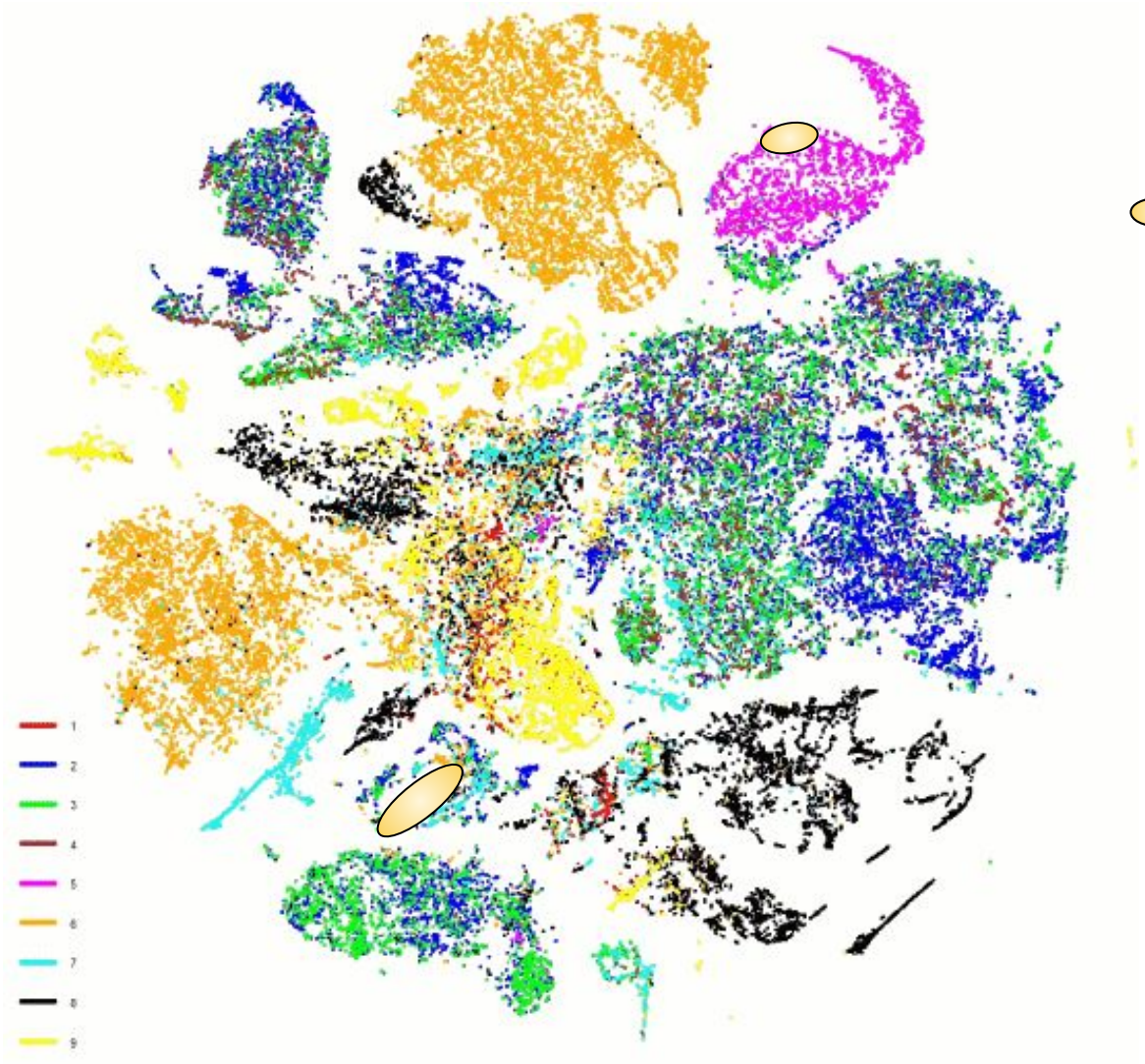
Imperfect sampling at PEAT

- PEAT employs “picture hunters” to gather pictures of particular crops and diseases
  - Data comes pre-labeled by picture hunters
  - It is much easier for experts to label

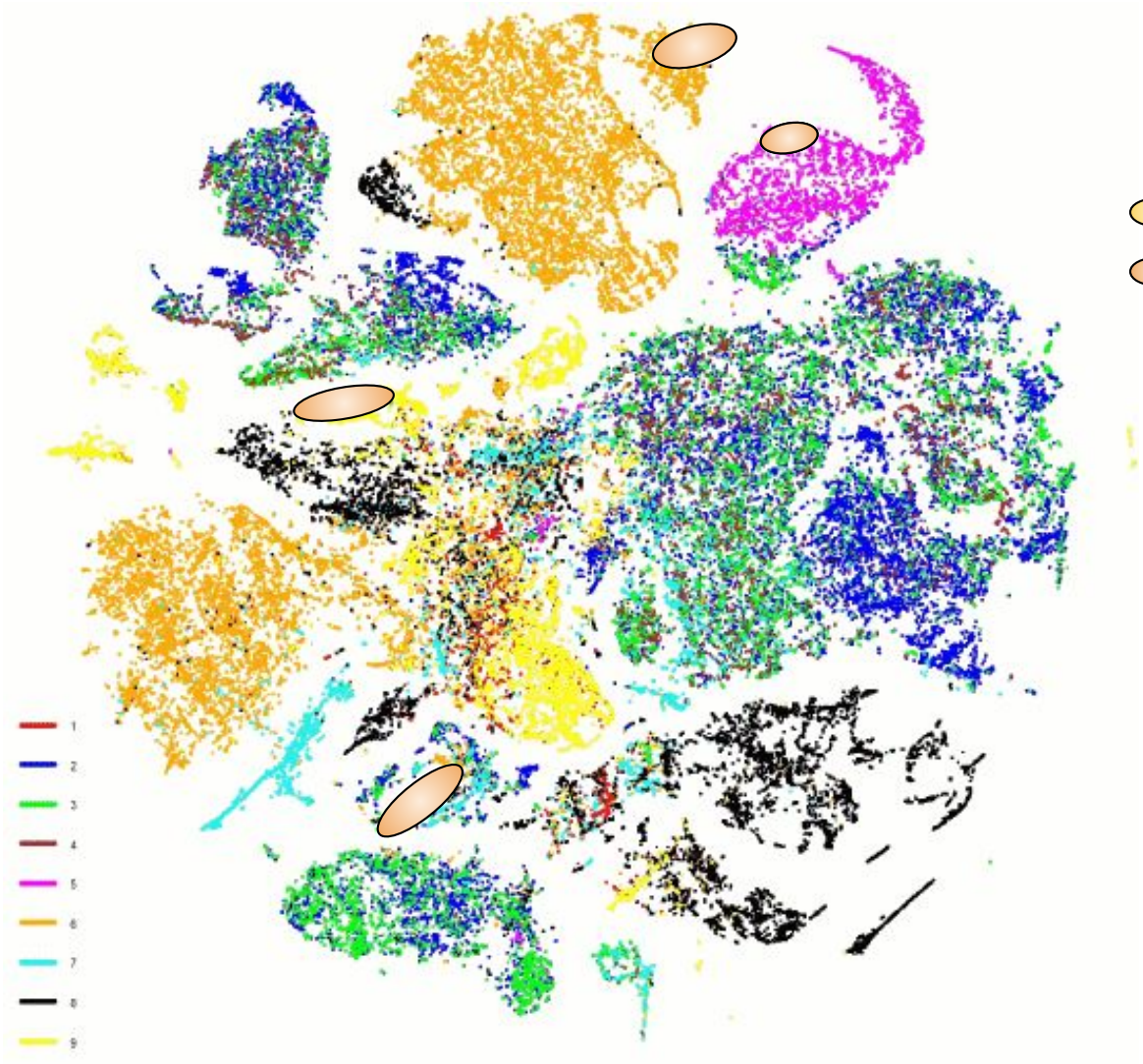
- “Picture hunting” is a difficult task:
  - Requires to know where to find given crops and diseases
  - Requires to reach these places physically
  - Places of interests might be far from each other
  - Cropping season and period of diseases are limited

- In summary:
  - Each picture hunter is limited:
    - in time
    - in places possible to reach
    - in the uncertainty of diseases apparition
  - Each picture hunter only has access to a **limited area in the whole distribution space**

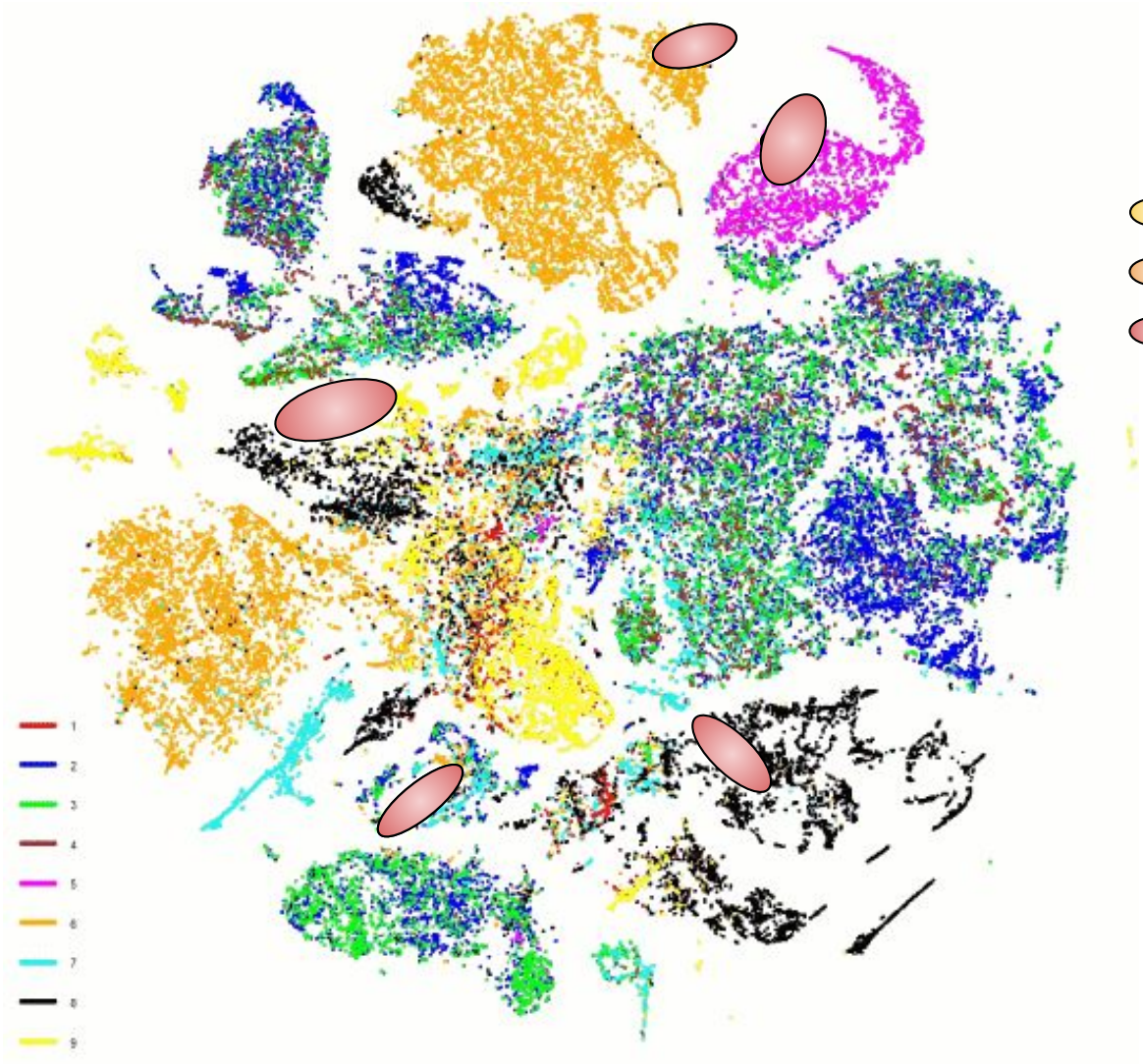
Imperfect sampling – toy illustration



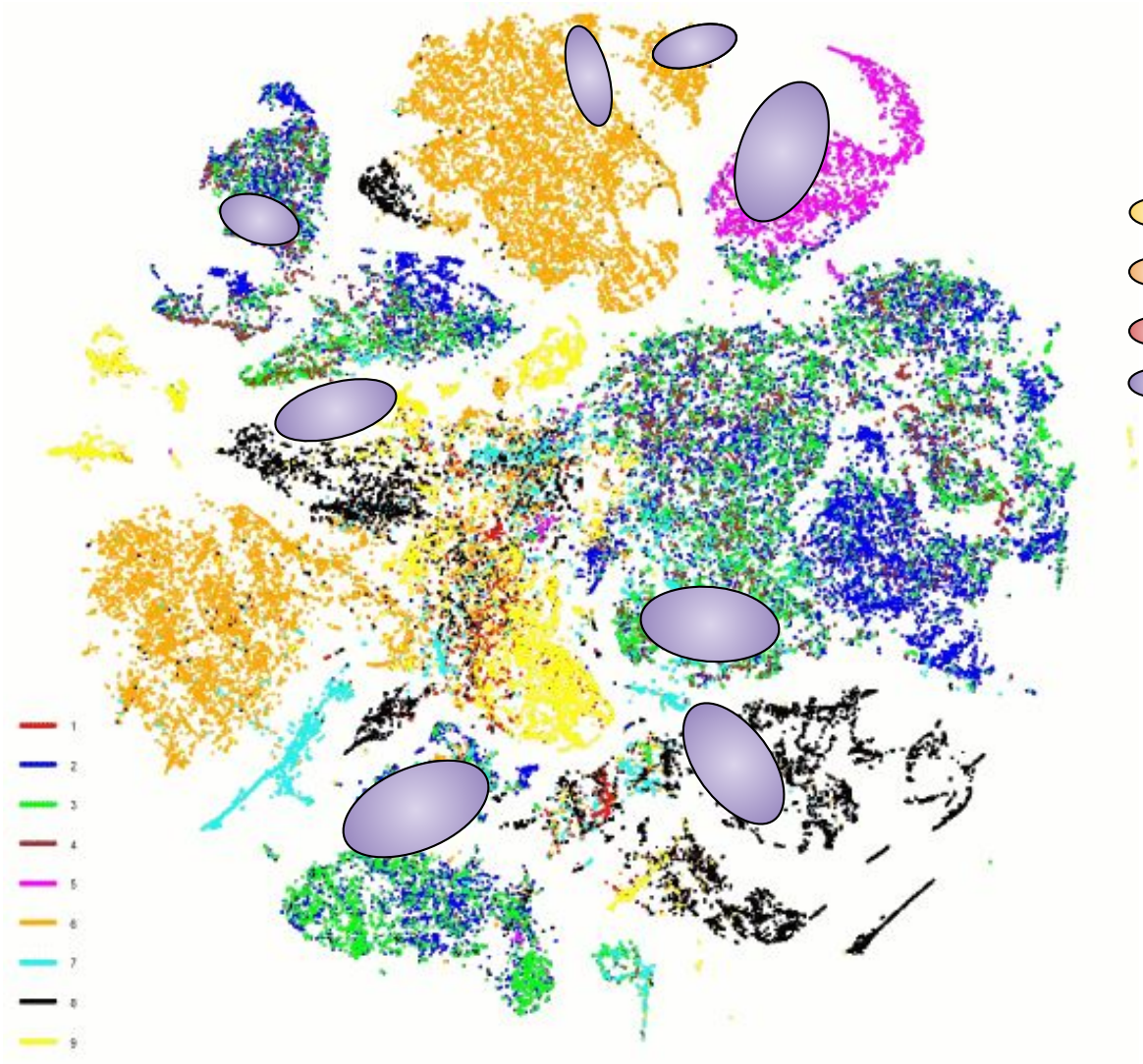
Imperfect sampling – toy illustration



Imperfect sampling – toy illustration



Imperfect sampling – toy illustration

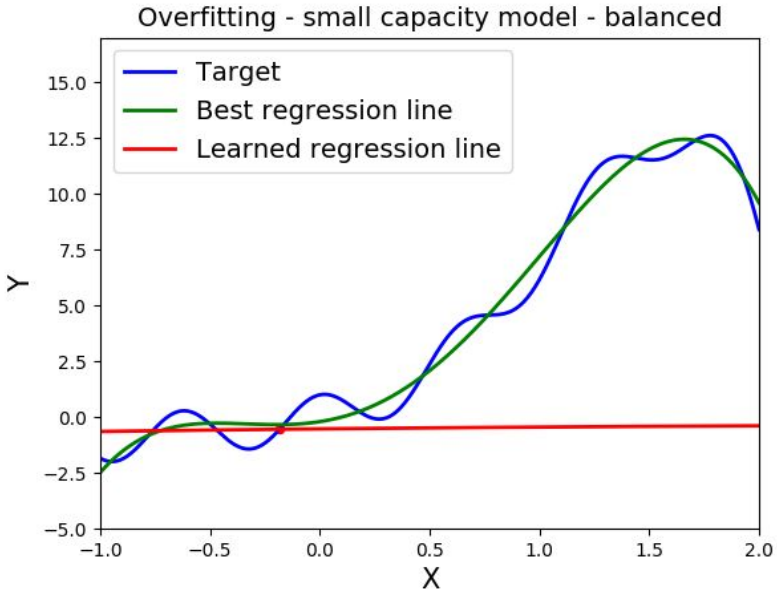


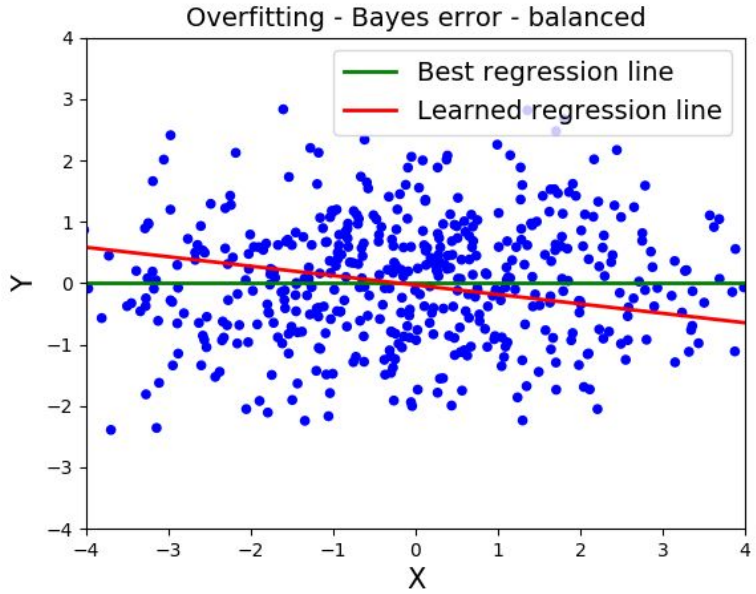
Imbalance in database

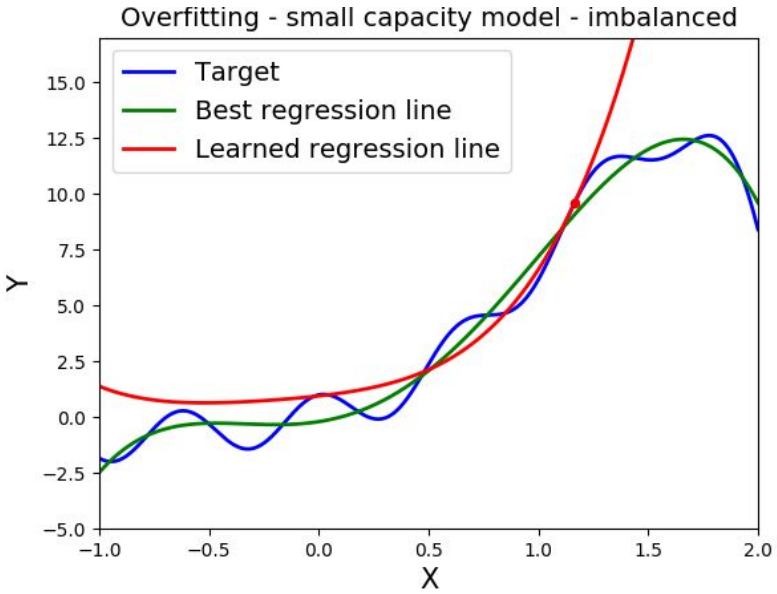
- Imperfect sampling:
  - Only part of the global distribution is sampled
  - At the same time, we want to sample as much as possible from what is accessible: e.g. in a field with a particular disease, picture hunters will gather a lot of pictures at the same place
  - Some picture hunters might contribute with several thousands of images in PEAT's database

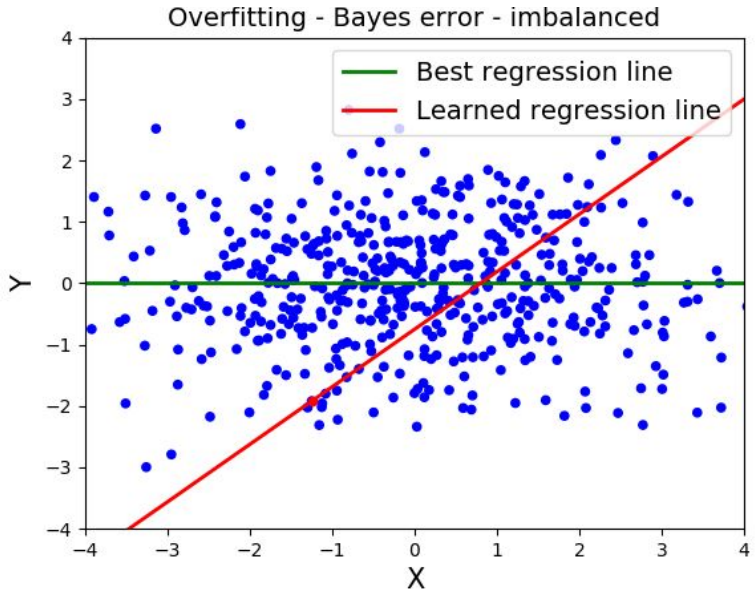
- This creates imbalance in database:
  - Some regions are highly sampled
  - Other regions are sparsely sampled, or not sampled at all

Imbalance in database - toy illustration

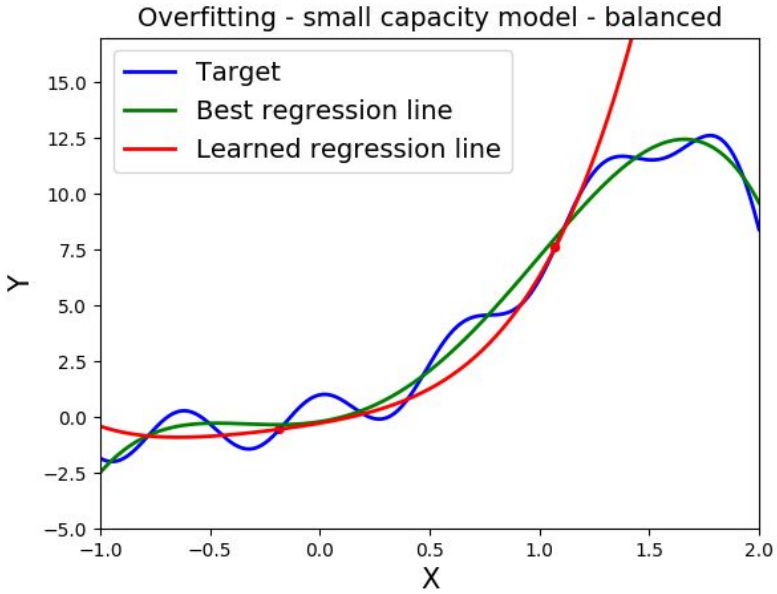


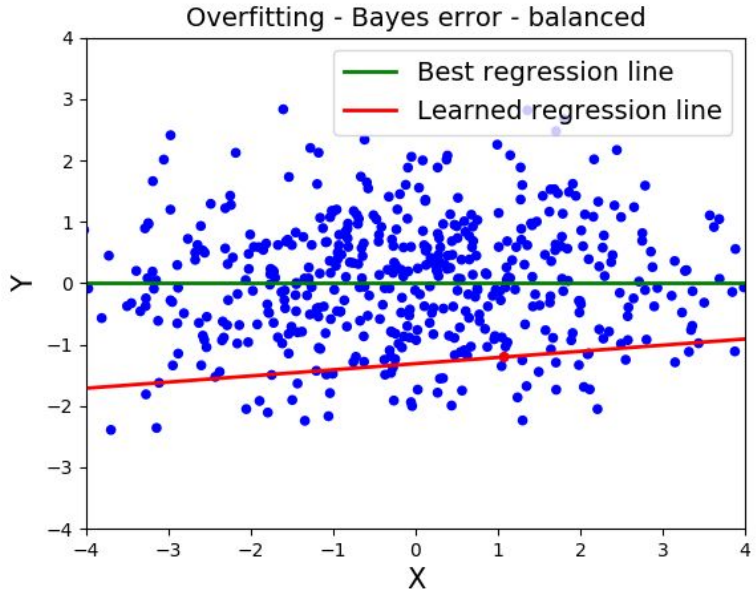


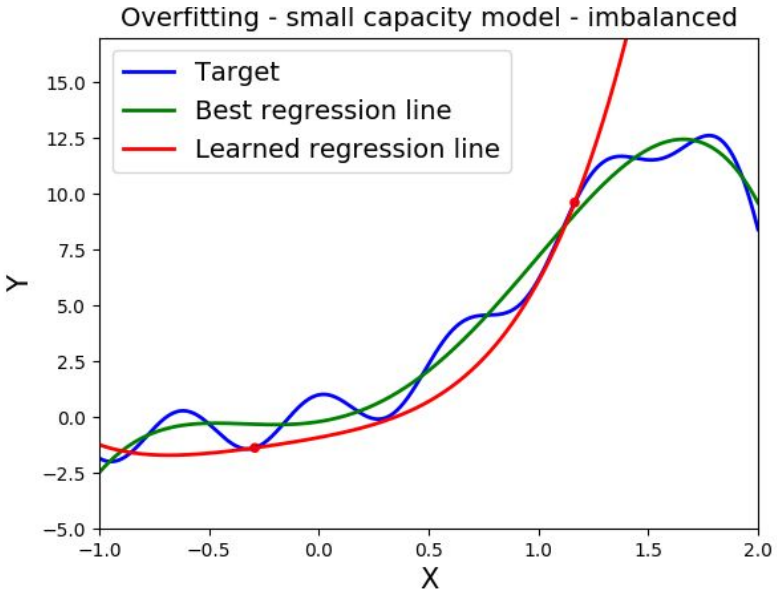


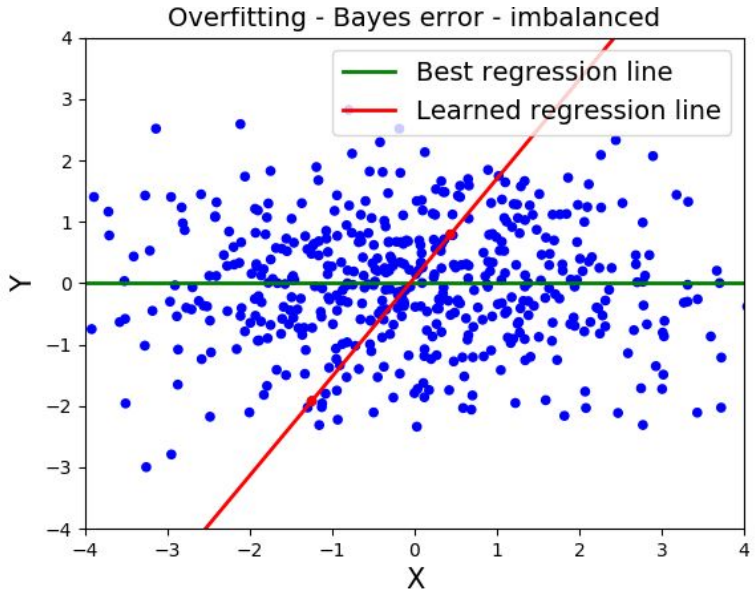


Imbalance in database - toy illustration

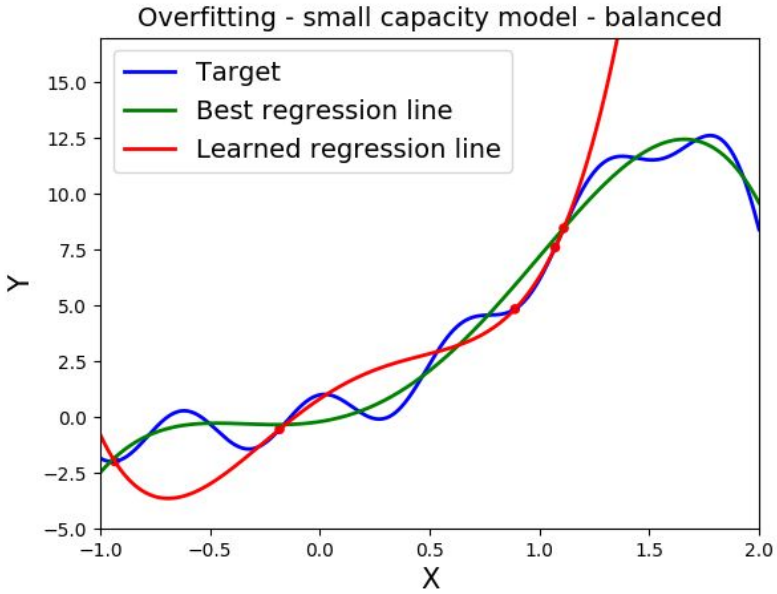


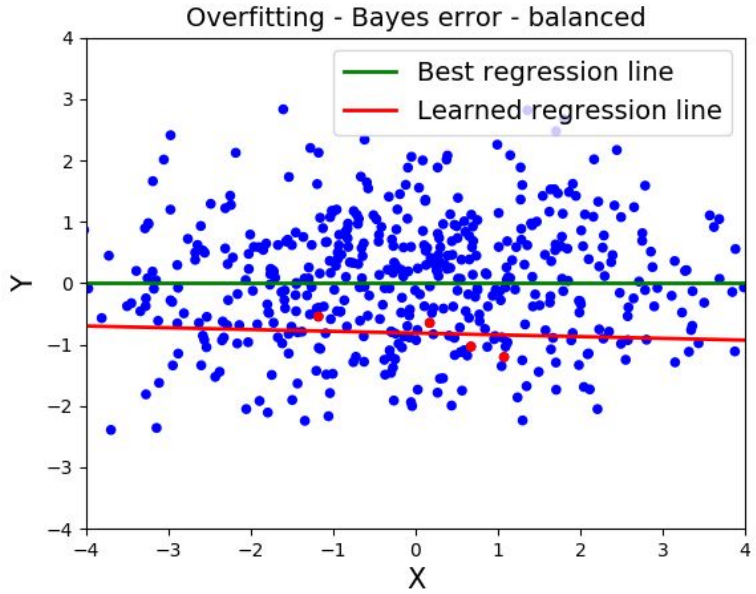


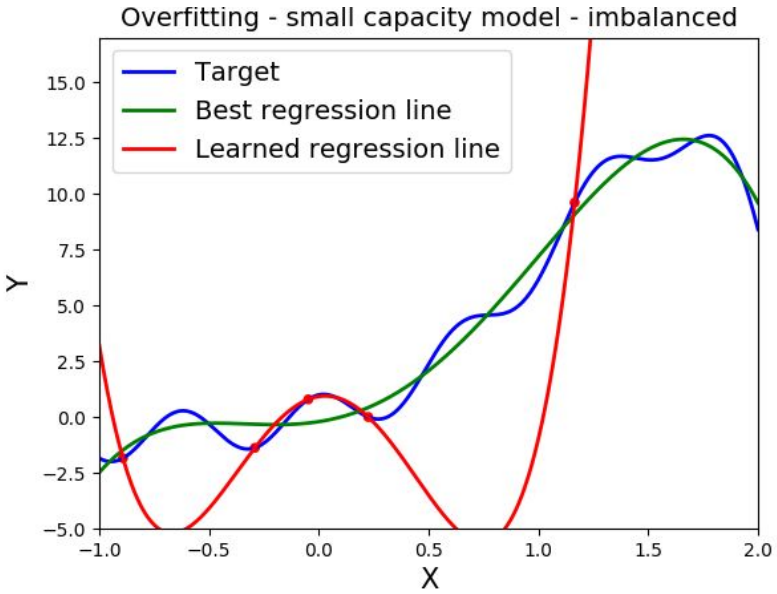


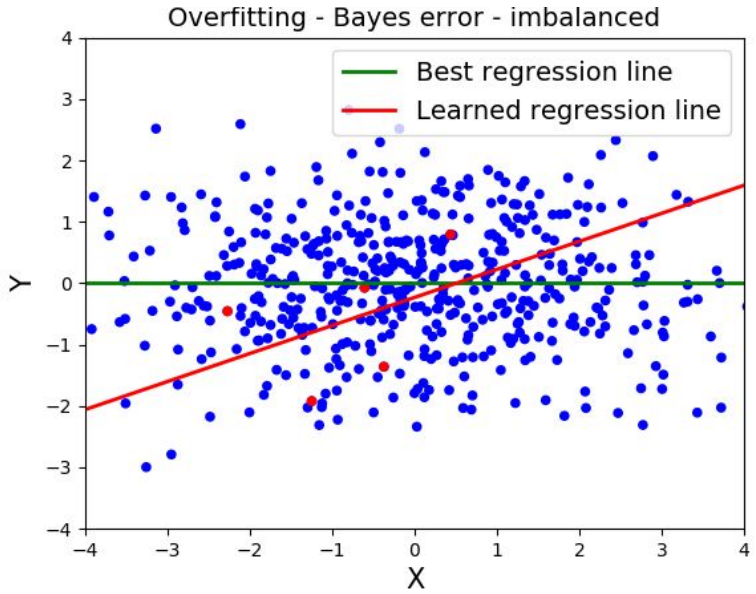


Imbalance in database - toy illustration

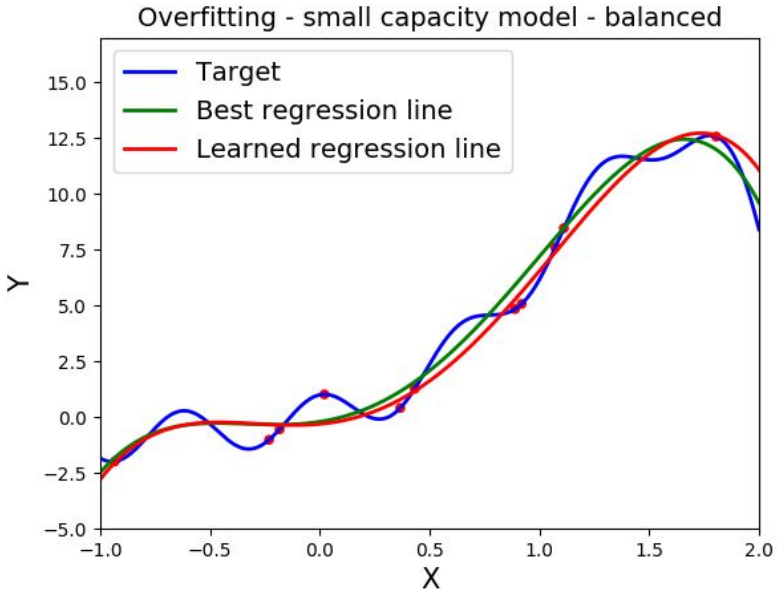


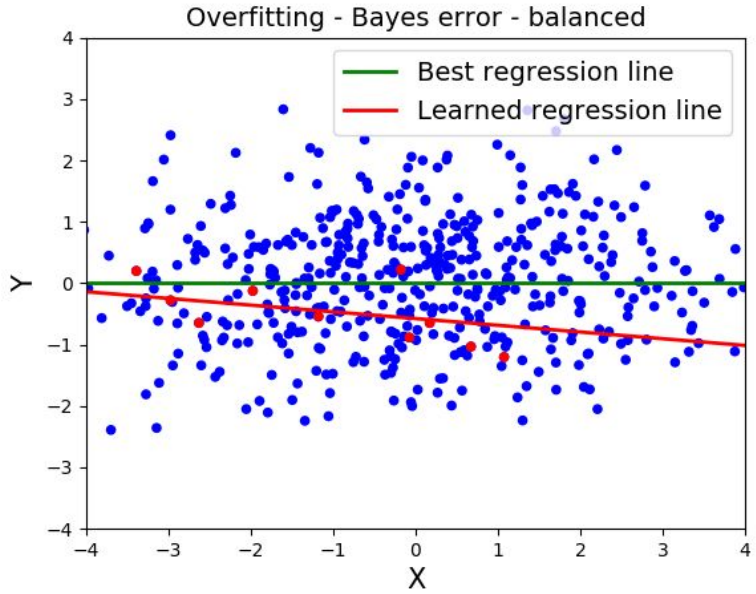


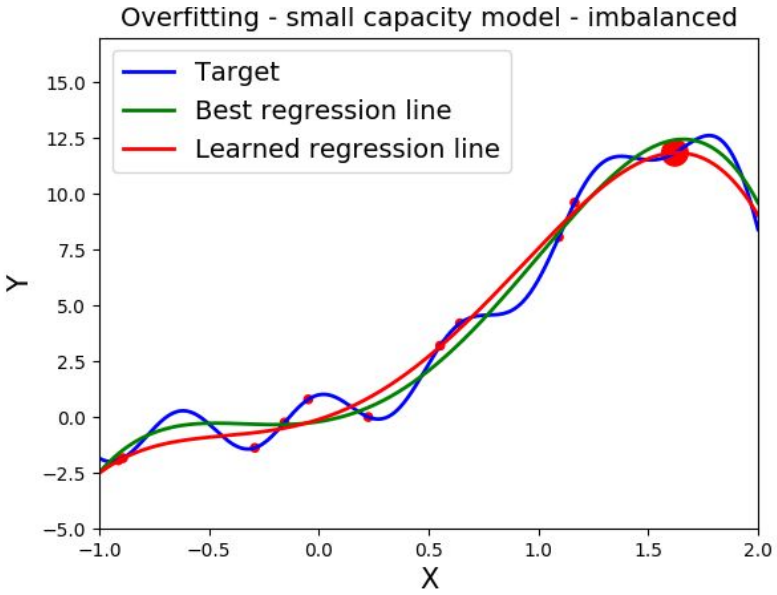


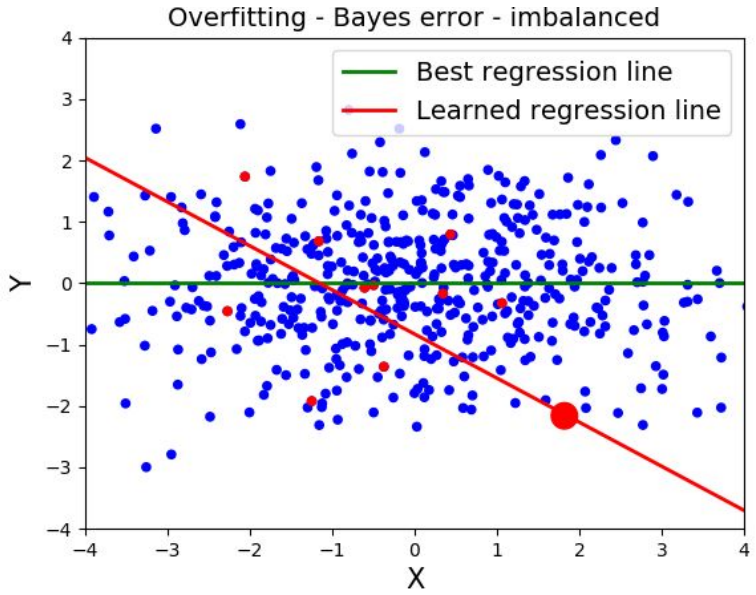


Imbalance in database - toy illustration

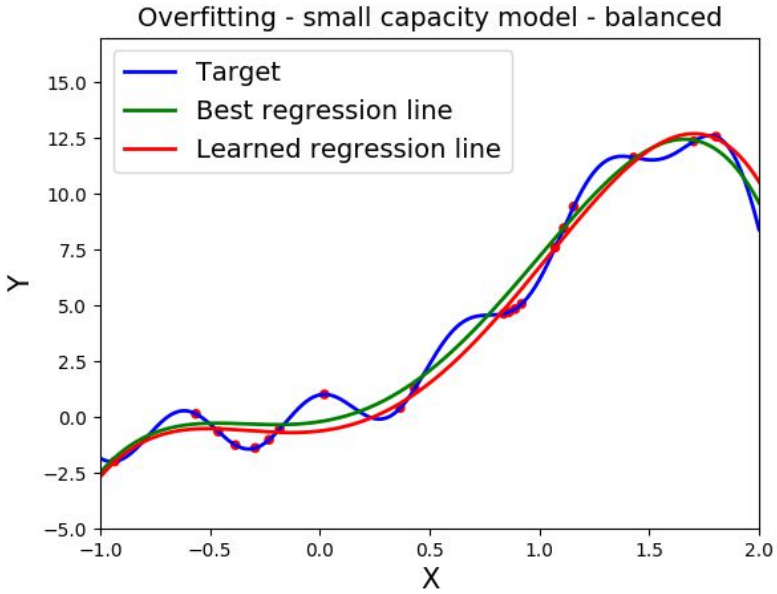


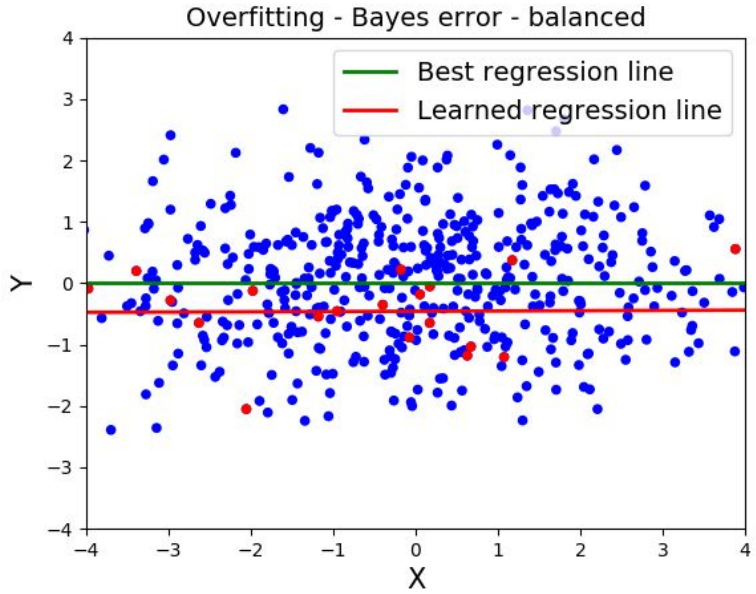


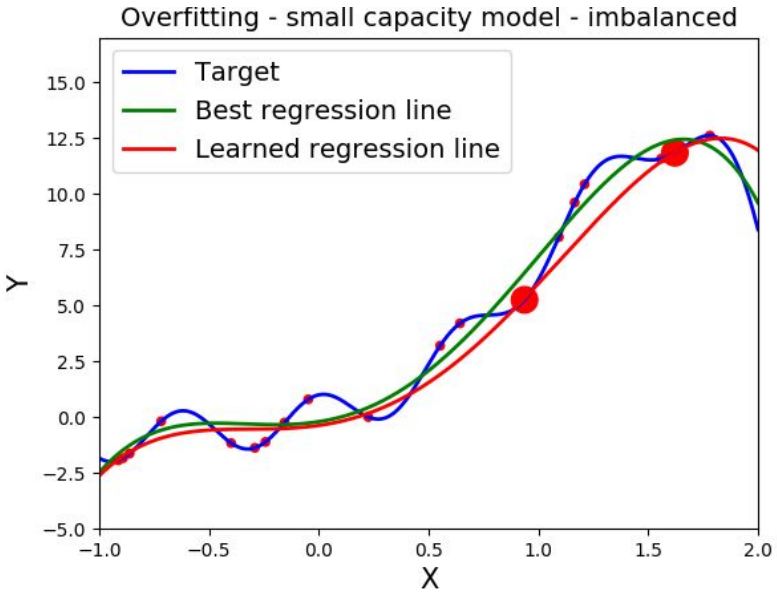


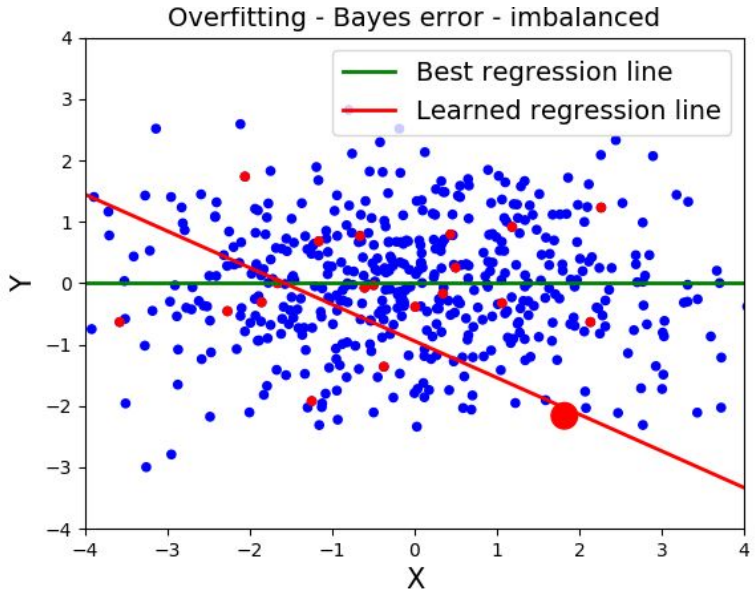


Imbalance in database - toy illustration

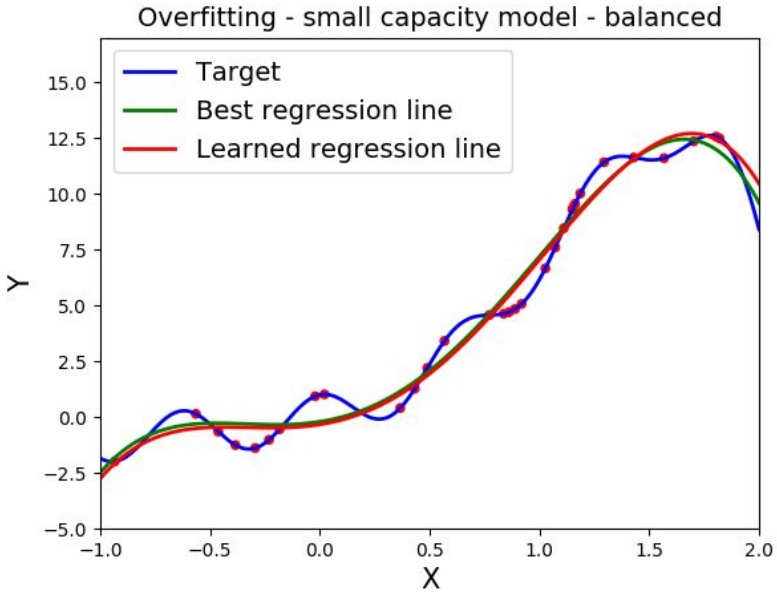


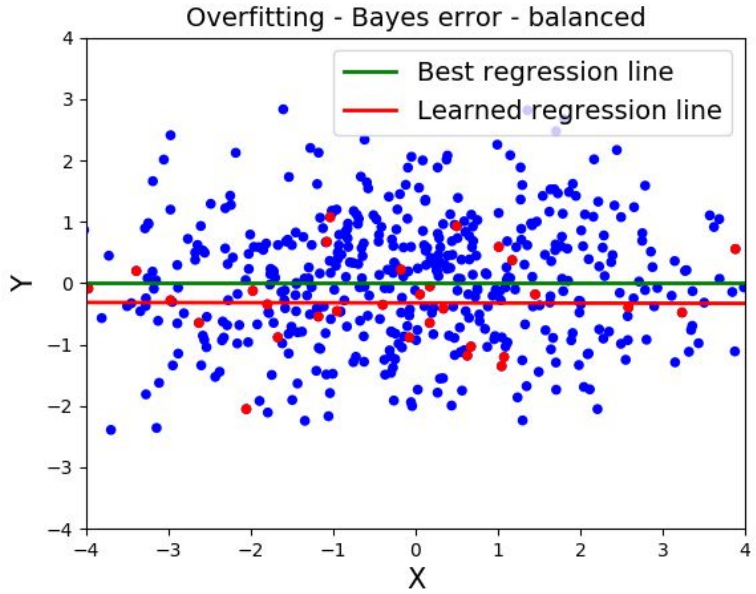


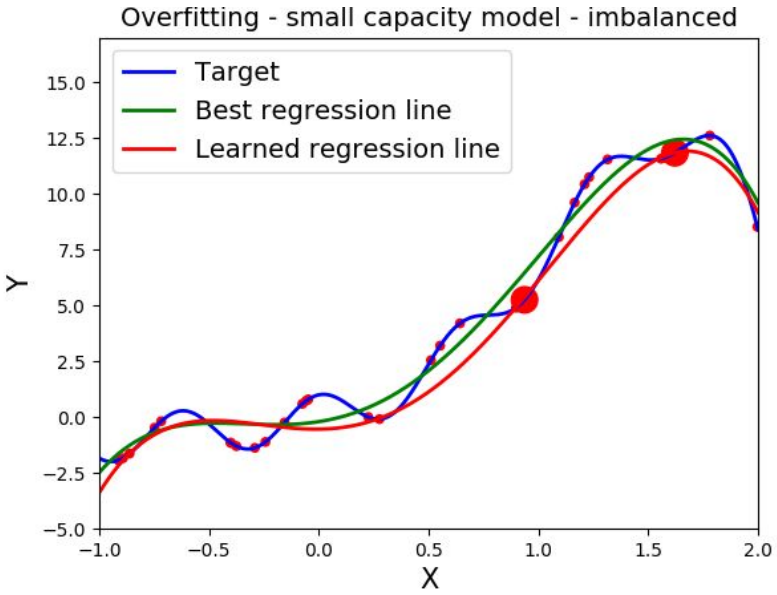


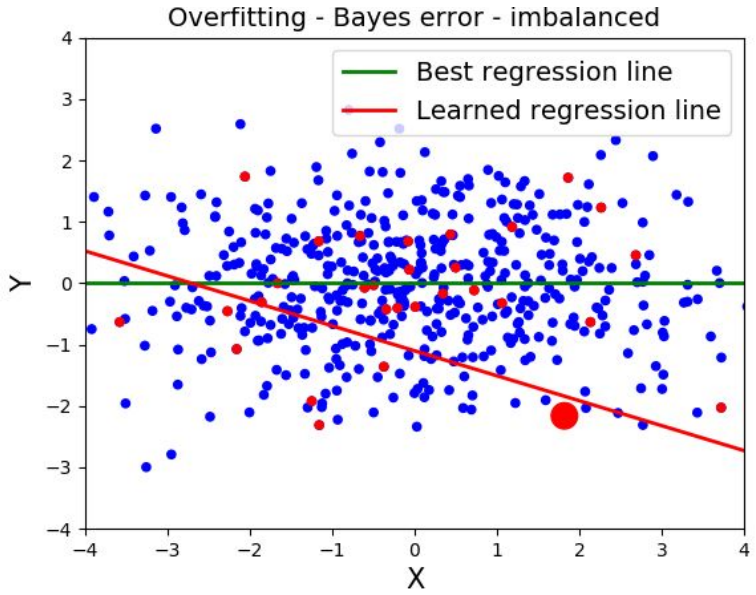


Imbalance in database - toy illustration

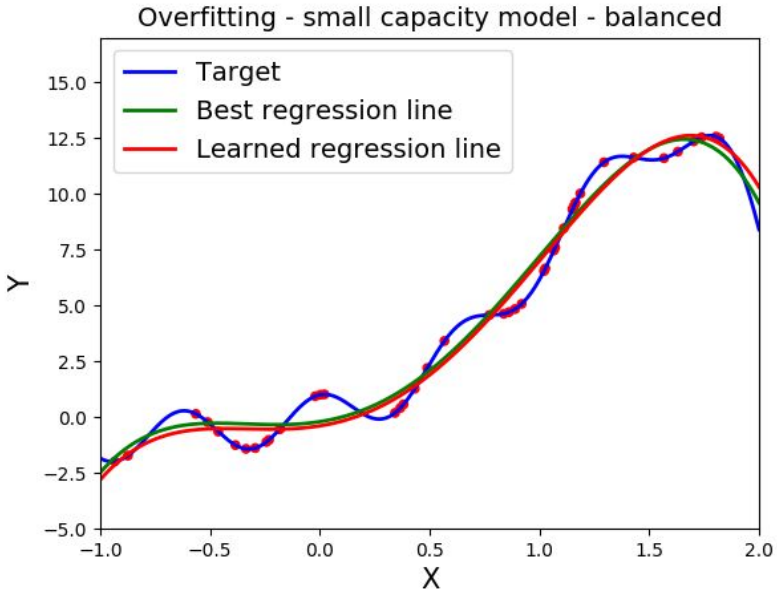


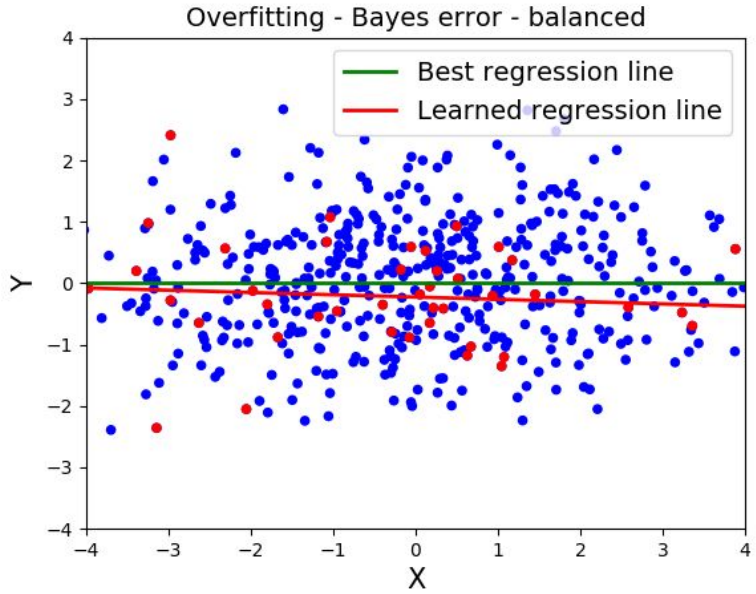


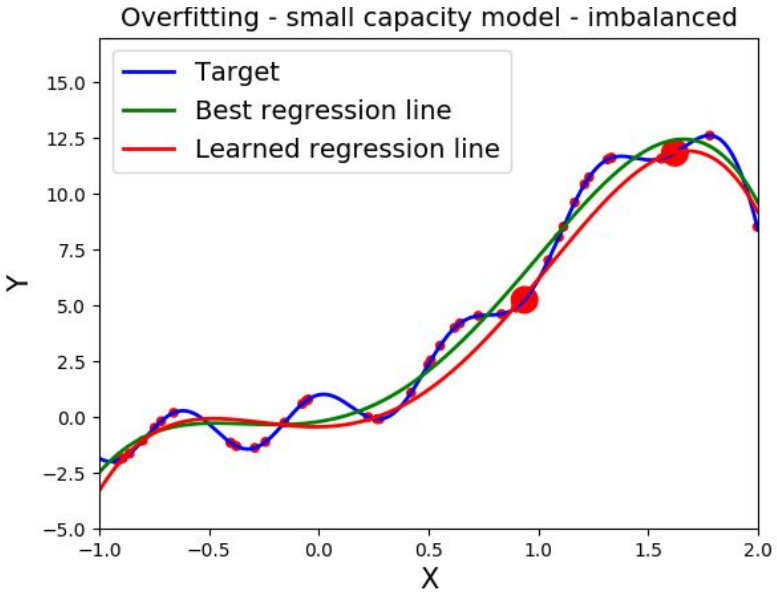


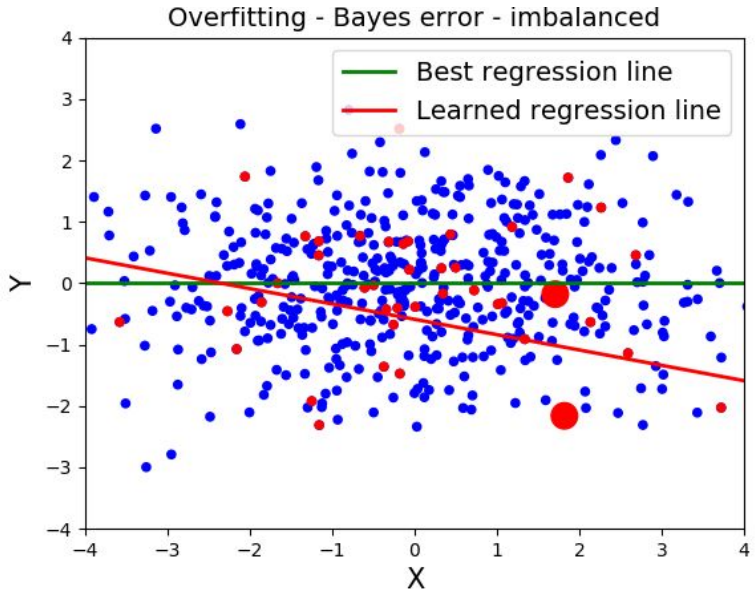


Imbalance in database - toy illustration

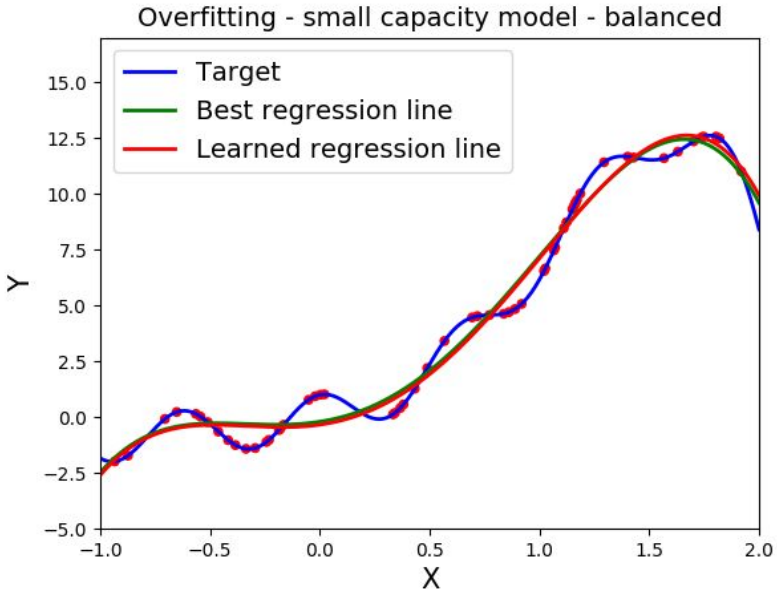


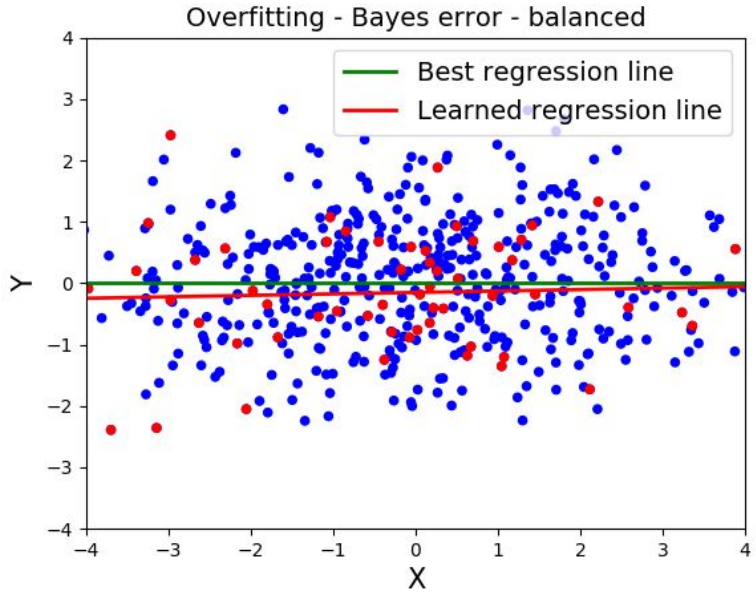


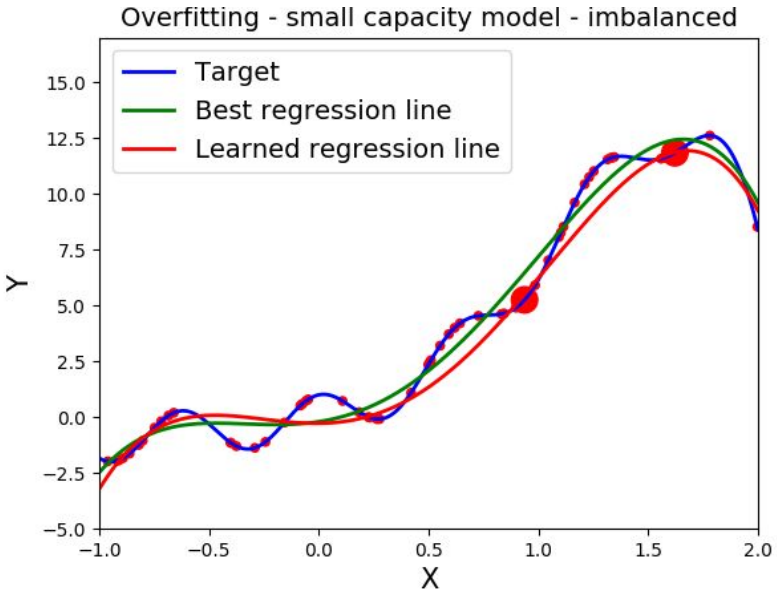


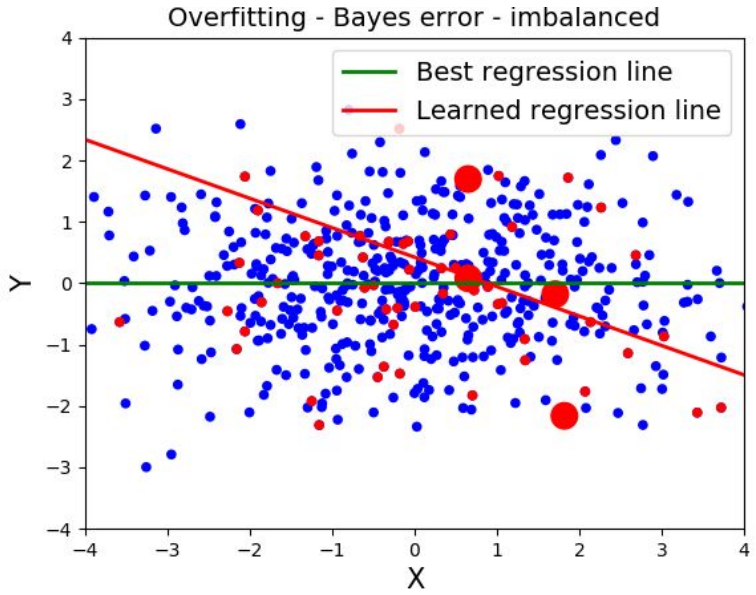


Imbalance in database - toy illustration

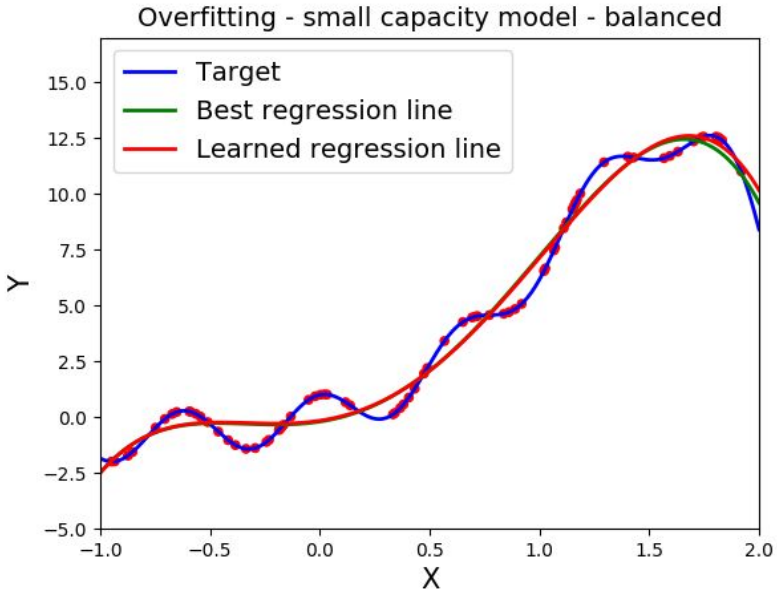


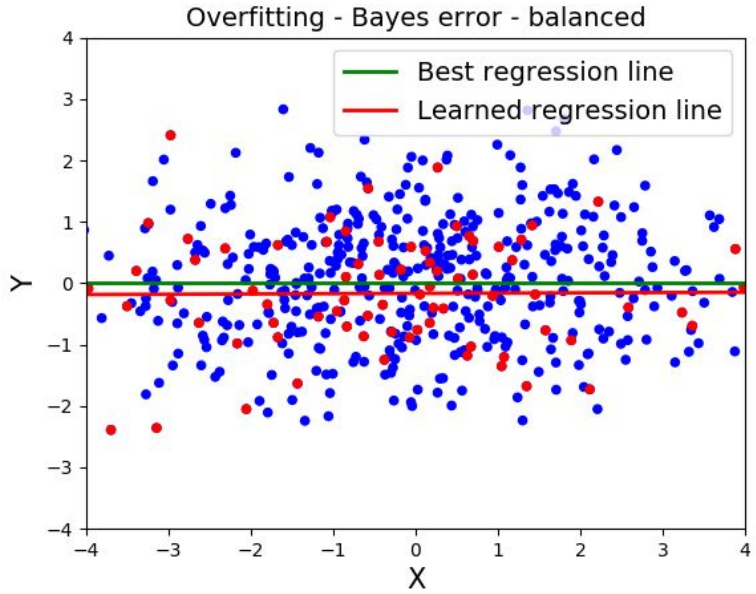


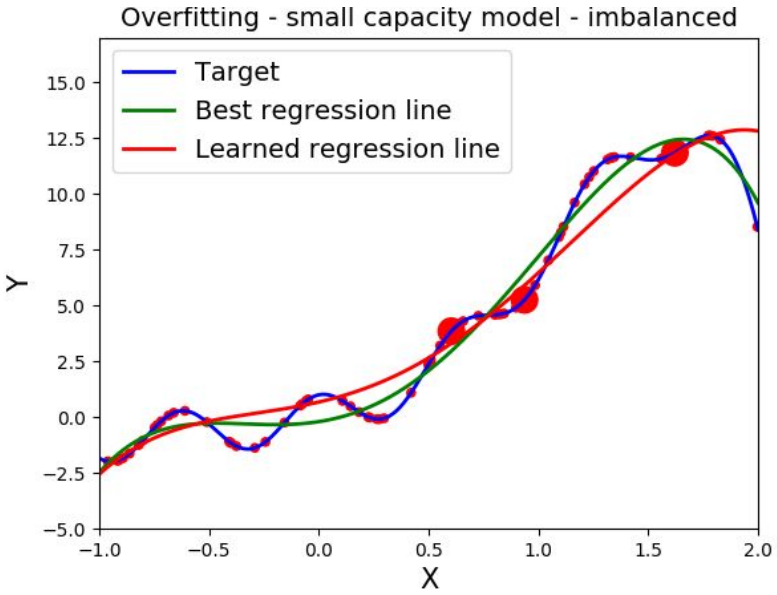


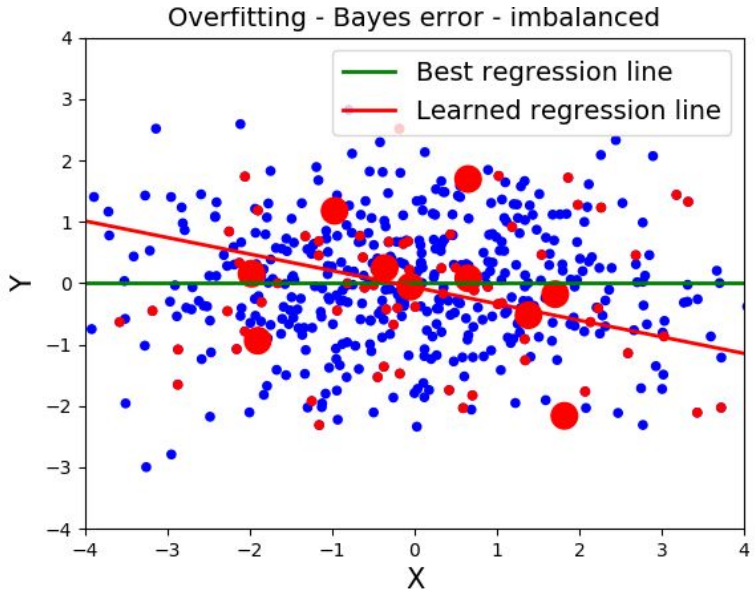


Imbalance in database - toy illustration

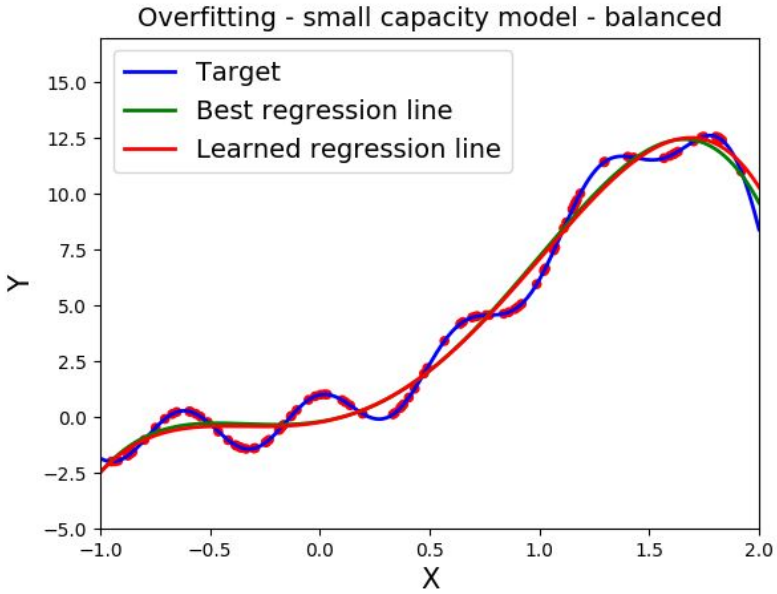


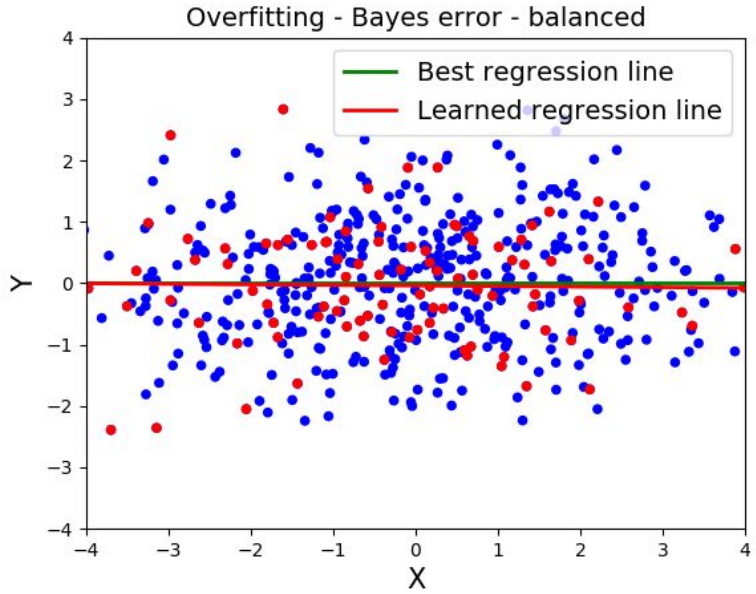


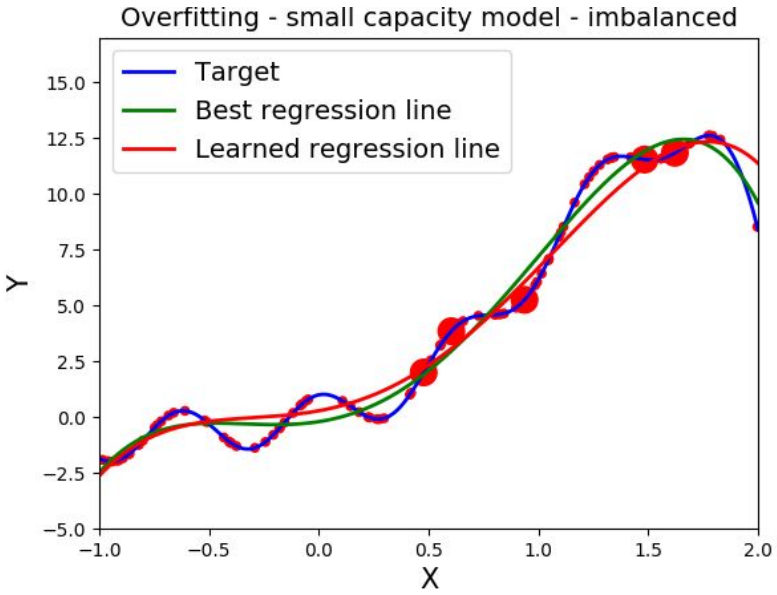


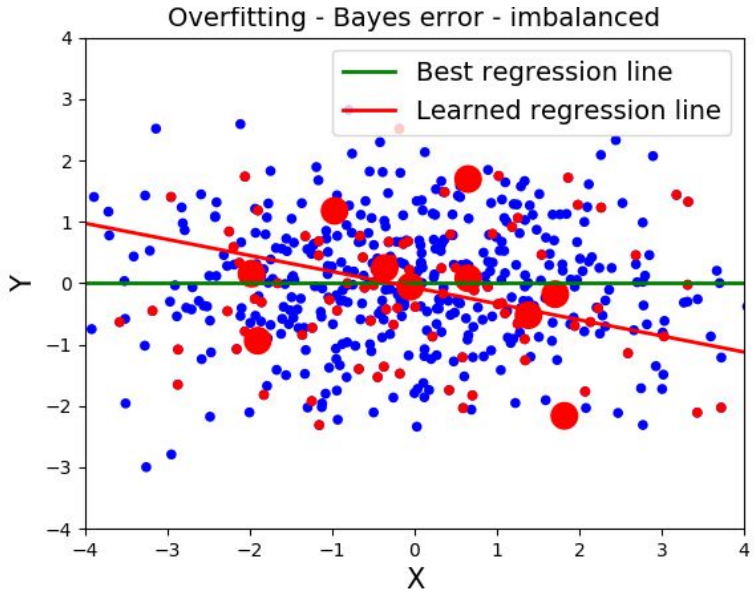


Imbalance in database - toy illustration

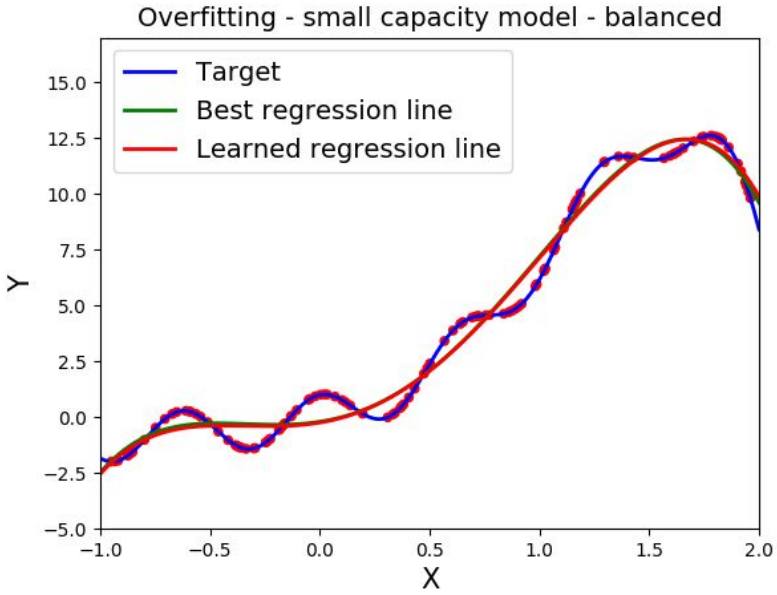


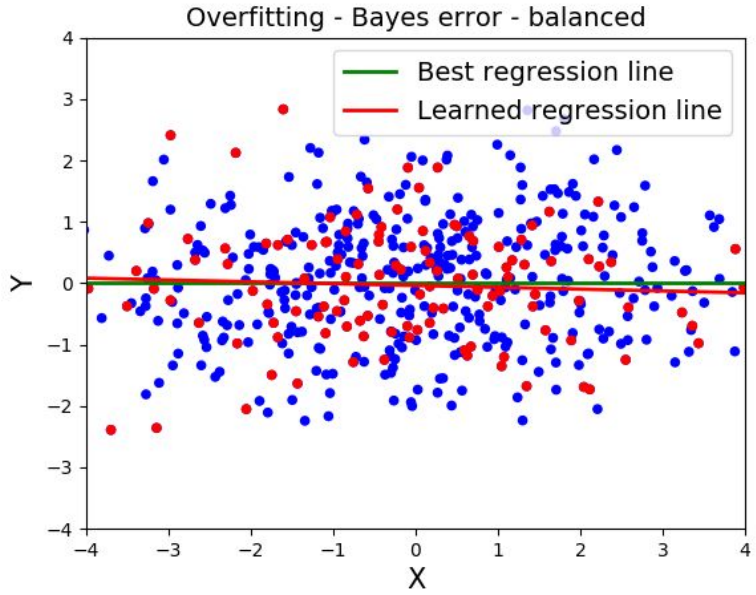


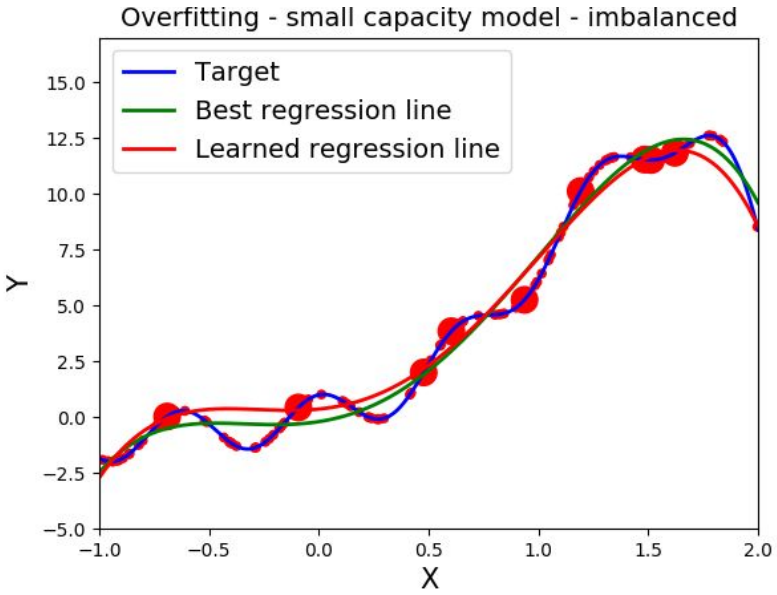


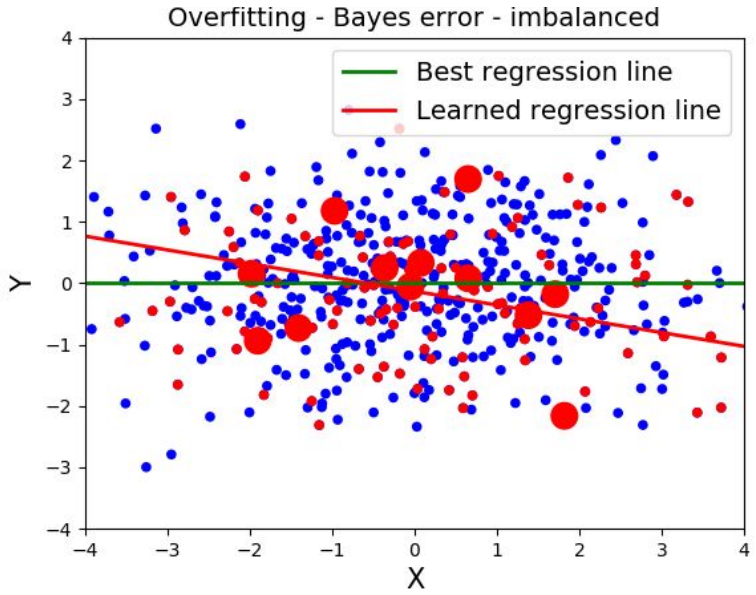


Imbalance in database - toy illustration









Training with imperfect sampling : inter-class balancing

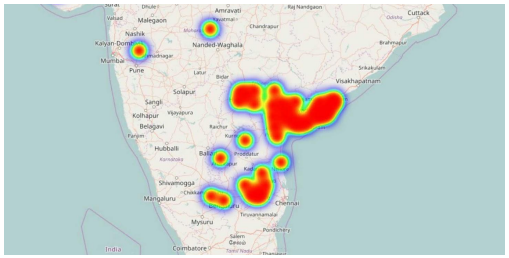
- Imperfect sampling distorts inter-class statistics:
  - Some diseases are over-represented
  - Other diseases are under-represented

- So how to set the training frequency of each class?

	Same frequency as database	Same frequency as deploy time
Pro	Limits overfitting	Training distribution = deploy distribution
Con	Training distribution ≠ deploy distribution	Amplifies overfitting

⇨ frequency = hyperparameter to optimize

- But how do we know the deploy distribution = expected distribution of diseases?
  - Diseases are seasonal and sometimes regional
  - Disease spreading is hard to predict → long-term goal of PEAT



Fair testing with imperfect sampling

- If we're not careful, database correlation introduces a bias in testing results:
  - If test images are correlated with train images → overestimation bias
  - To be fair, test images must be as uncorrelated as possible with train images

- Ideally, each class is split between train and test:
  - with acceptable data proportions
  - with minimum correlation between train and test

- We train this **"split"** model and test its performance

- If we are satisfied with test results, we train a **"full"** model on our full database for deployment

Modeling database correlation

- Define the binary variable of right / wrong classification for each image I in test:
  - ξ[I] = 0 if top-1 prediction is wrong
  - ξ[I] = 1 if top-1 prediction is right

- $\mathbb{E}[\xi]$ = top-1 accuracy

- Correlation between any pair of pictures I_1 and I_2:
  - We can model correlation based on metadata (e.g. user / country / GPS coordinates / date):

$$\rho(\xi(I_1), \xi(I_2)) = \rho(meta_1, meta_2)$$

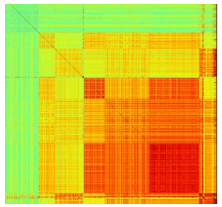
  - Can we also model correlation based on image content (e.g. using bottleneck features from a carefully crafted model)?

$$\rho(\xi(I_1), \xi(I_2)) = \rho(bottleneck_1, bottleneck_2)$$

Training with imperfect sampling : intra-class balancing

- If we know the correlation matrix between all image pairs in a given class:

$$\rho = \left(\rho[\xi(I_i), \xi(I_j)]\right)_{i,j}$$



- Suppose we want to keep using SGD with standard loss: to optimally weight training images, find positive weights minimizing variance of estimator:

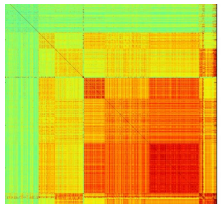
$$w_{train} = \underset{w \geq 0, \sum w_i = 1}{\arg\min} \left[\sum_{i,j} w_i \rho_{i,j} w_j \right] \qquad w_{train} = \underset{||w||_1 = 1}{\arg\min} \left[w^T \rho w \right]$$

- Toy example:
  - Suppose we have gathered N independent images, but some images have been duplicated in our database
  - Optimal weighting: remove duplicates to keep each independent image only once and weight all images equally
  - We recover this with w_{train}

Testing with imperfect sampling: intra-class balancing

- Again suppose we know the correlation matrix between all image pairs:

$$\rho = \left(\rho[\xi(I_i), \xi(I_j)]\right)_{i,j}$$



- Optimally weight test images in each class:

$$w_{test} = \underset{||w||_1=1}{\arg\min} \left[w^T \rho w\right]$$

- Closed-form solution is available when correlation matrix is non singular:

$$w_{test} = \frac{\rho^{-1}\mathbb{1}}{||\rho^{-1}\mathbb{1}||_1}$$

- Toy example:
  - Suppose we have gathered N independent images, but some images have been duplicated in our database
  - Optimal weighting: remove duplicates to keep each independent image only once and weight all images equally
  - We recover this with w_{test}

Monitoring database with imperfect sampling

- We want to monitor the status of different classes in the database to guide data collection and data labeling efforts:
  - How well these classes perform in the Image Recognition?
  - What are current limitations in the database?
  - How much data do we have?

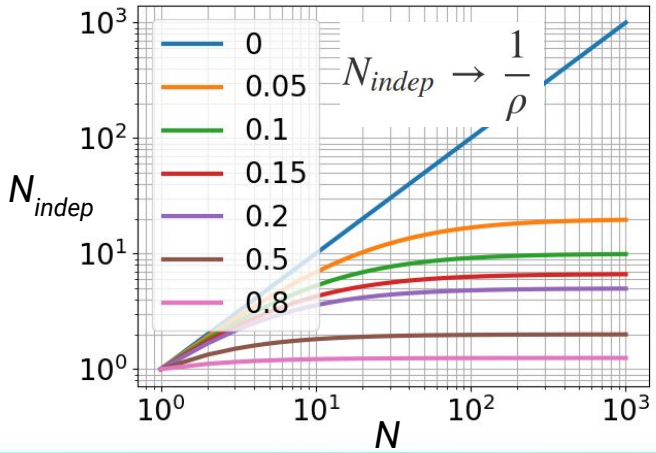
- Possible approaches:
  1. Consider the estimated accuracy of the Image Recognition on this particular class
  2. Have realistic indicators to monitor amount of data

- Simple realistic data monitoring:

$$w_{test} = \frac{\rho^{-1}\mathbb{1}}{||\rho^{-1}\mathbb{1}||_1} \qquad \sigma^2(\xi_{w_{test}}) = \frac{1}{N_{indep}}\sigma_0^2$$

- Toy example with fixed correlation:

$$\begin{bmatrix} 1 & \rho & \rho & \rho & \rho \\ \rho & 1 & \rho & \rho & \rho \\ \rho & \rho & 1 & \rho & \rho \\ \rho & \rho & \rho & 1 & \rho \\ \rho & \rho & \rho & \rho & 1 \end{bmatrix}$$



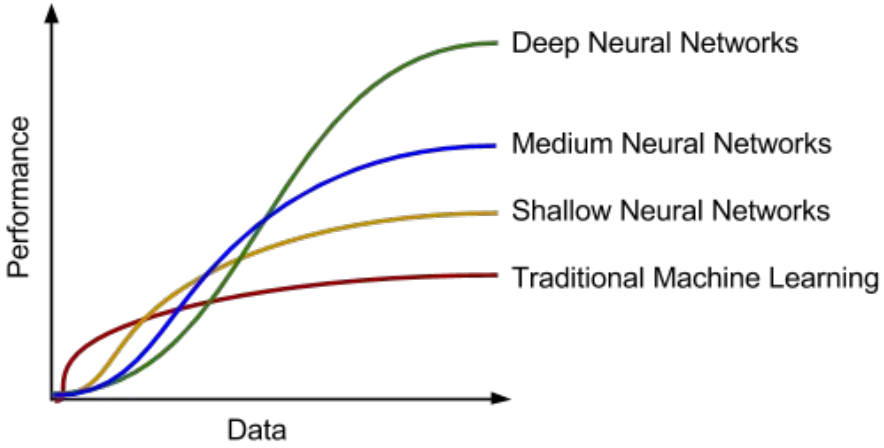
I. Context

II. Imperfect sampling

III. Data bottleneck

Data bottleneck

- Deep Learning enabled many A.I. successes recently, but it depends heavily on **data**
- Usually:
  - unlabeled data is quite cheap
  - labeled data is expensive
- Labeled data becomes the true **bottleneck** for improving performance



- Possible solutions:
  - crowdsourcing (mechanical turk)
  - active learning
  - weakly-supervised learning
  - semi-supervised learning

Data bottleneck at PEAT

- Images from Picture Hunters are prelabeled but they are subject to imperfect sampling
  - experts also need to label images from Plantix = direct exploration of the database
  - **problem**: it is nearly impossible for a given expert to know all crop diseases

- Realistically:
  - An expert knows a given list of diseases
  - For a given image, the expert is only able to say which diseases from the list are present in the image

- Plain database exploration:
  - requires expert work
  - requires time
  - only provides a **partial labeling**!

$$\begin{pmatrix} ? \\ ? \\ 0 \\ ? \\ 1 \\ 0 \\ ? \\ ? \end{pmatrix}$$

Overcoming data bottleneck - Image prefiltering

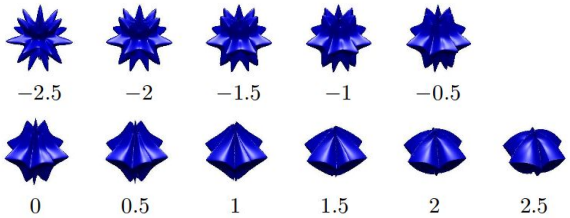
- A possible solution is to **prefilter** images before showing them to the experts:
  - Filter non interesting / non relevant images
  - Only select images with particular symptoms
  - Only select good candidates for a particular list of diseases
  - Ideally, use **active learning** to select images which add the most information

- **Pro**: it makes database labeling manageable for experts

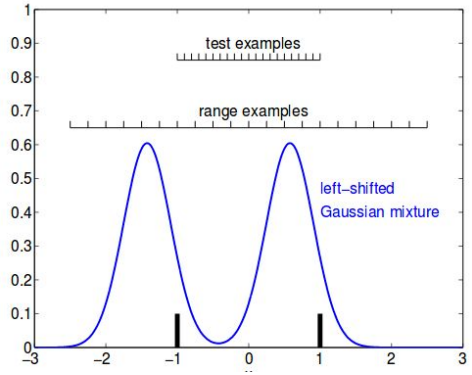
- **Con**: it creates distortion in the labeled distribution
  - Only images that pass prefiltering will be labeled
  - So we miss again parts of the distribution
  - We have to be careful with the “prefiltering threshold” and find a tradeoff between prefiltering efficiency and non-distortion in the distribution

Overcoming data bottleneck - Semi-supervised learning

- Experiments show that humans do perform semi-supervised learning (Humans Perform Semi-Supervised Classification Too Zhu et al. 2007)

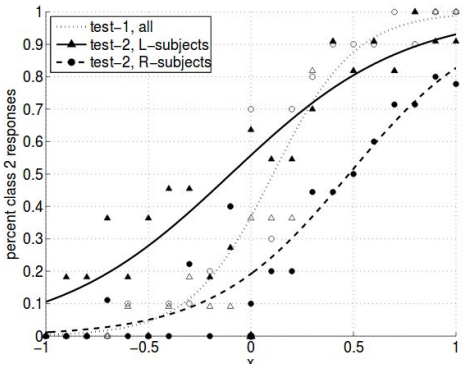


- Suppose:
  - 2 classes: N and B
  - 2 labeled points: (x=-1, N) and (x=1, B)
  - unlabeled data is sampled with mixture of 2 Gaussians, shifted left (L) or right (R) around labeled points



- Two different groups of subjects are shown:
  1. First, only the labeled data
  2. Then the unlabeled data with either shift L or shift R

- Conclusion:
  - Both groups give the same decision boundary with just labeled data (as expected)
  - Then the decision boundary is influenced by the direction of the shift in the sampling of unlabeled data
  - **Interpretation:** in the absence of labeled information, humans use the information coming from the discontinuity in unlabeled data distribution



Overcoming data bottleneck - Semi-supervised learning

- How to leverage unlabeled data with **Semi-Supervised Learning (SSL)?**

- Virtual Adversarial Training (VAT)
  - models should be robust to adversarial perturbations at all points of the distribution
    - models should have flat output at all points of the distribution
    - this supposes some form of discontinuity of data between classes (= low density boundary region)

- Π-model
  - models should be invariant to specific forms of stochasticity (e.g. specific data augmentation) at all points of the distribution

- Entropy Minimization:
  - models should be “confident” at all points of the distribution
    - this discourages the boundary condition to come close to any point in the data distribution
    - this also supposes some form of discontinuity of data between classes

- DCGAN
  - Features from generator and discriminator on unlabeled data are transferable to the classification task

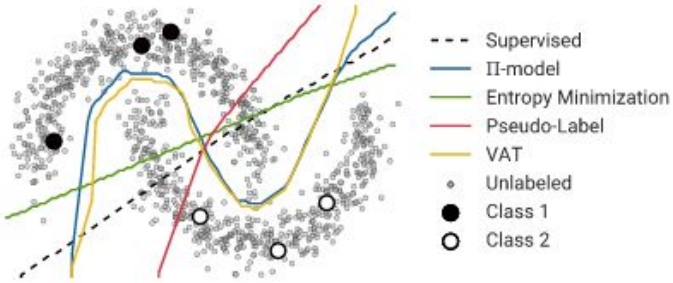
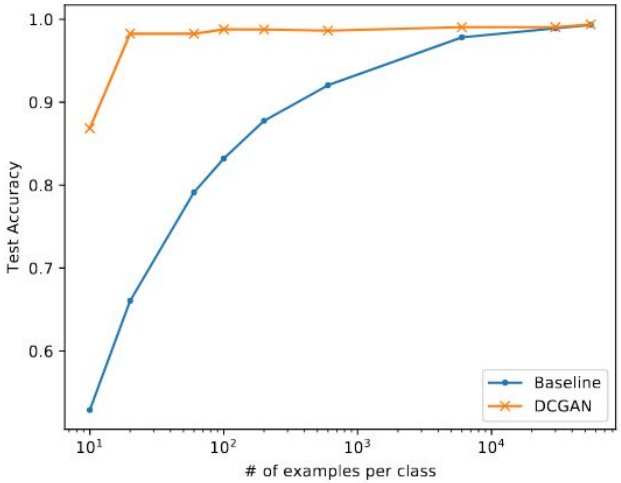


Figure from [Realistic Evaluation of Semi-Supervised Learning Algorithms](), Avital et al. 2018



Overcoming data bottleneck - Semi-supervised learning at PEAT

- Amount of data:
  - ~4M unlabeled images from Plantix
  - 1 order of magnitude more than labeled images

- 2 possible caveats of SSL described in Avital et al. 2018:
  - SSL can hurt when unlabeled data does not come from the same distribution as the test distribution of labeled data
  - transfer learning sometimes work better than any SSL approach

- In our case:
  - There is indeed a discrepancy between distributions of unlabeled data and labeled data
    - however the unlabeled data distribution (Plantix distribution) is the **one we're interested in**
    - on the other hand, labeled data is not exactly the distribution we're interested in (it's only an **imperfect proxy** for it)
    - so we should benefit even more from SS?

  - When each is used separately, transfer learning sometimes works better than SSL
    - but what if they are used together?
    - initialize with pre-trained weights, then use SSL as a regularizer during training

Thank you!