META LEARNING

Machine Learning for Autonomous Robots

Dr. Matias Valdenegro-Toro

Slides by Anett Seeland, Mario Michael Krell, Alexander Fabisch

Assistant Professor at University of Groningen

December 13, 2022 – Bremen, Deutschland

University of Bremen

German Research Center for Artificial Intelligence

1. Overview of Meta Learning
2. Introduction to Ensemble methods
3. Bias-Variance Trade-off
4. Bagging
5. Boosting
6. Modern Ensemble Methods
7. references

Overview of Meta Learning

What is Meta Learning?

Meta-learning in this lecture is different from the *Meta-Learning* in the current Deep Learning literature.

- Meta-Learning here refers to classic models of models (the meta part).
- Meta-Learning in Deep Learning is also models of models but more related to things like *learning to learn* and its use with neural network models.
- This is just a word of warning in case you perform a literature search and are a bit confused...

What is Meta Learning?

Meta-learning includes... (ordered by degree of automization)

- guidelines and heuristics for the application of algorithms
- **tools that enable users without experience in machine learning (ML) to solve tasks with ML, these tools include**
  - broad set of algorithms
  - evaluation methods
  - model selection
- meta-level learning schemes:
  - multi-task learning
  - transfer learning
  - lifelong learning
  - **ensemble learning**
- automatic machine learning *(active field of research)*

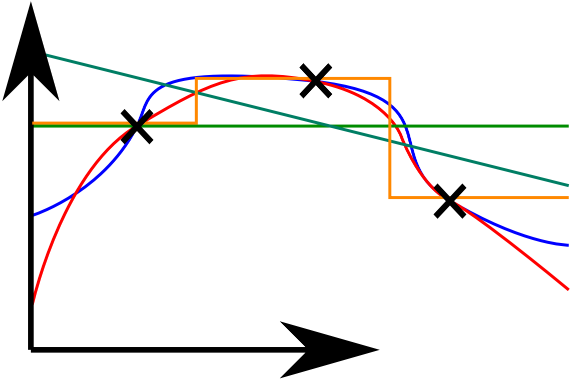
Why not just one algorithm that solves everything?

There is no free lunch.

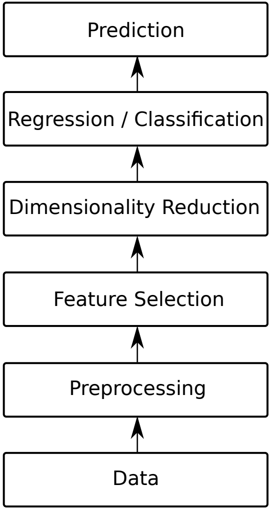
- Every learning algorithm has an **inductive bias**.
- No learning algorithm is better than any other learning algorithm **on average over all datasets.**
- No Free Lunch (NFL) theorem.

Inductive Bias

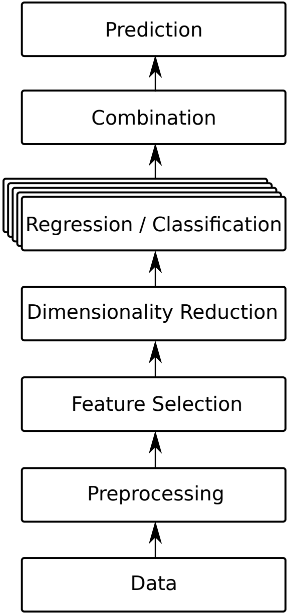
- An unbiased algorithm cannot generalize. Having no inductive bias means immediate overfitting.
- The bias of a learning algorithm is called **inductive bias**.
- Examples: min. CV error, maximum margin, Occam's razor



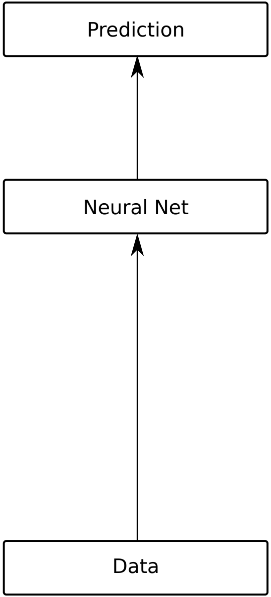
Standard Pipeline



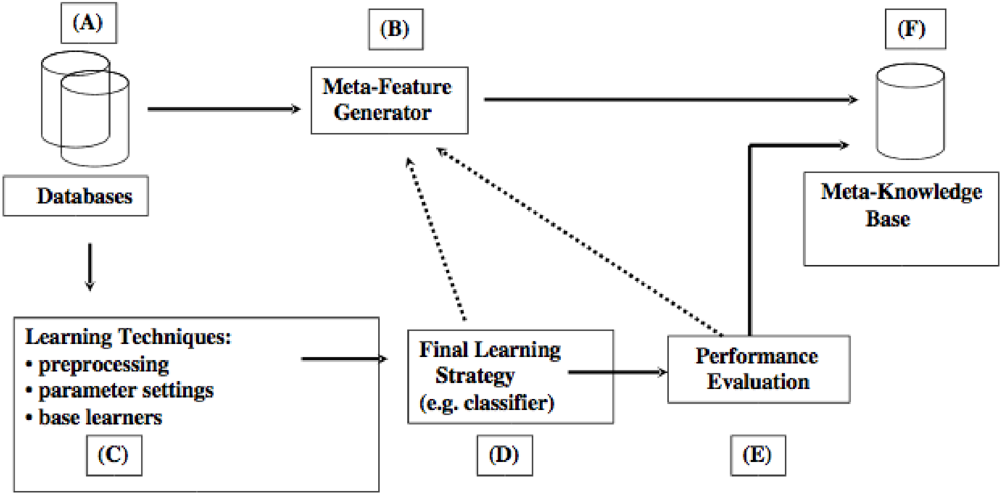
Ensemble Learning



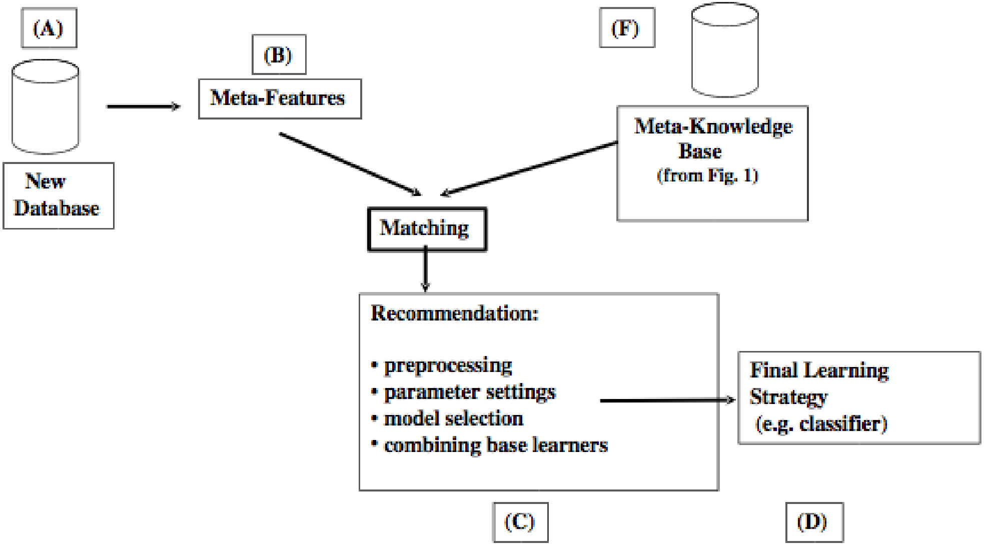
Outlook



The Knowledge-Acquisition Mode



The Advisory Mode



Techniques in Meta Learning

Dataset Characterization:

- Statistical and Information-Theoretic Characterization
  - # classes, # features, # examples, $\frac{\#\text{ examples}}{\#\text{ features}}$
  - degree of correlation between features and target concept
  - skewness, kurtosis, signal-to-noise ratio
- Model-Based Characterization
  - e.g. support vector machine: C, kernel, kernel parameters
- Landmarking
  - performance of different learning mechanisms
  - sampling landmarks

Introduction to Ensemble methods

Motivation

When *wise people* make **critical** decisions,
they usually take into account
the opinions of *several* experts
rather than relying
on their own judgment or
that of a solitary trusted adviser.

Ensemble methods

Ensemble: using many classifiers together

- could be different types of classifiers
- each classifier gives a different view
- usually each is trained differently
- need a sensible method of combining classifiers

Popular methods:

- bagging
- boosting
- stacking
- random forest (see Classification 1)

Combining classifiers

Suppose that we have:

- m labeled examples $\langle \mathbf{x}, f(\mathbf{x}) \rangle$, where $f : R \rightarrow \{-1, +1\}$
- a number of T hypotheses $h_1, \ldots, h_T$, each trained on the data
- each $h_t : R \rightarrow \{-1, +1\}$

How should we combine the T classifiers?

- choose h_t with the least training error
- choose h_t with the least validation error
- use all T classifiers in a sum
- use all T classifiers in a weighted sum

Bias-Variance Trade-off

Bias-Variance Decomposition

In the **ideal situation** of an infinite dataset:

→ error will still occur because no learning scheme is perfect

→ **bias** for the learning problem

In **practical situations** there is another error:

- stems from the particular training set used (unavoidably finite)

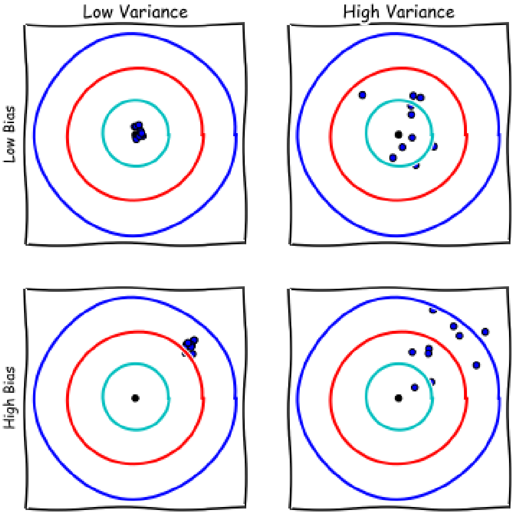
→ not fully representative of the actual population of instances

→ **variance** for the learning method

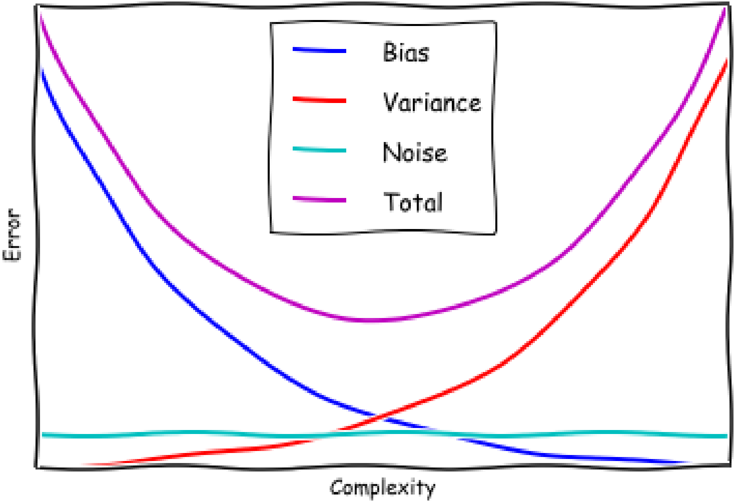
total expected error is a sum of **bias** and **variance** (and noise)

⇒ combining multiple classifiers decreases the expected error by reducing the variance component

Bias / Variance



Bias / Variance



Inductive Bias vs. Bias Error

Can you imagine an example where we have an inductive bias but no bias error?

Example:

- We know that we want to approximate a sine function:

$$f(x) = a \sin(bx + c)$$

- We don't know a, b, c
- We can fit a, b, c from data

→ Strong inductive bias

→ No bias error

Bagging

Unstable learners

Imagine:

- several randomly chosen training sets of the same size
- build a decision tree for each dataset

Will the trees make the same predictions?

→ No! (particularly for small datasets)

→ reason: unstable learning process

⇒ voting becomes more reliable with more votes

⇒ combined classifiers will seldom be less accurate

⇒ but improvement is not guaranteed

Bagging – bootstrap aggregation

- reduces variance (typically uses complex models)
- random sampling with replacement from dataset
- applies base learner to each new dataset
- votes with equal weights

combined model ...

+ ... often performs significantly better than single model
+ ... is never substantially worse

Bagging – bootstrap aggregation

```
Bagging($\mathcal{D}, T$):
    model generation($\mathcal{D}, T$):
        $m \leftarrow |\mathcal{D}|$
        for $t \in \{1, \dots, T\}$ do
            $\mathcal{D}_t \leftarrow bootstrap(\mathcal{D}, m)$
            $h_t \leftarrow build\_model(\mathcal{D}_t)$
        end
    classification($x$):
        for $t \in \{1, \dots, T\}$ do
            $store(h_t(x))$
        end
        return class that has been predicted most often
```

Boosting

Ideas behind Boosting

- reduces bias (typically uses simple models)
- design significantly different models
- each model treats a reasonable percentage of the data correctly

→ aim: models complement one another, each being a specialist in a part of the domain

- weighting is used to give more influence to the more successful models

Bagging vs. Boosting

Similarities:

- both use voting
- both combine models of the same type, e.g. decision trees

Differences:

- Boosting is sequential
- Boosting weights a model's contribution by its performance
- Boosting reduces bias, bagging reduces variance

The AdaBoost Algorithm

```
AdaBoost($\mathcal{D}$, $T$):
    model generation($\mathcal{D}$, $T$):
        for $i \in \{1, \ldots, m\}$ do  $W_1(i) \leftarrow 1/m$ // sample weights
        ;
        for $t \in \{1, \ldots, T\}$ do
            $h_t \leftarrow build\_model(\mathcal{D}, W_t)$;
            $\epsilon_t \leftarrow error(\mathcal{D}, W_t, h_t)$;
            if $\epsilon_t = 0$ or $\epsilon_t \geq 1/2$ then continue;
            $\alpha_t \leftarrow \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$;
            for $i \in \{1, \ldots, m\}$ do
                $W_{t+1}(i) \leftarrow \frac{W_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = f(x_i) \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq f(x_i) \end{cases}$ ;
                // $Z_t$ to normalize $W_{t+1}$
            end
        end
```

The AdaBoost Algorithm

```
AdaBoost($\mathcal{D}$, $T$):
    ⋮
    classification($x$):
        $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
```

If model class can handle weighted instances:

$$\epsilon = \frac{\sum w_i}{\sum w_j} \text{ for } i \in \{i | h(x_i) \neq f(x_i)\} \text{ and } j \in \{1, \ldots, m\}$$

Otherwise, resample the data according to a distribution W_t

How much can the weights change?

$$W_{t+1}(i) \leftarrow \frac{W_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = f(x_i) \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq f(x_i) \end{cases}$$

recall:

- $0 < \epsilon_t < 0.5$
- $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t}) \rightarrow \alpha \in (0, \infty)$
- $e^{-\alpha_t} = \sqrt{\epsilon_t/(1-\epsilon_t)}$, correct → sample weigth decreases
- $e^{\alpha_t} = \sqrt{(1-\epsilon_t)/\epsilon_t}$, error → sample weight increases

Thus, if $\epsilon_t \approx 0$ then $W_t(i) \rightarrow 0$

Role of α_t

Recall the final classifiers's form:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Also,

$$\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$$

Thus,

- $\epsilon_t = 0.5 \rightarrow \alpha_t = 0$
- $\epsilon_t < 0.5 \rightarrow \alpha_t > 0$
- $\epsilon_t > 0.5 \rightarrow \alpha_t < 0$, will not happen, because those classifiers are ignored!

Greedy search

Interestingly, AdaBoost is a greedy strategy.

It always minimizes the error with respect to the current W_t, and does not backtrack.

How does boosting help?

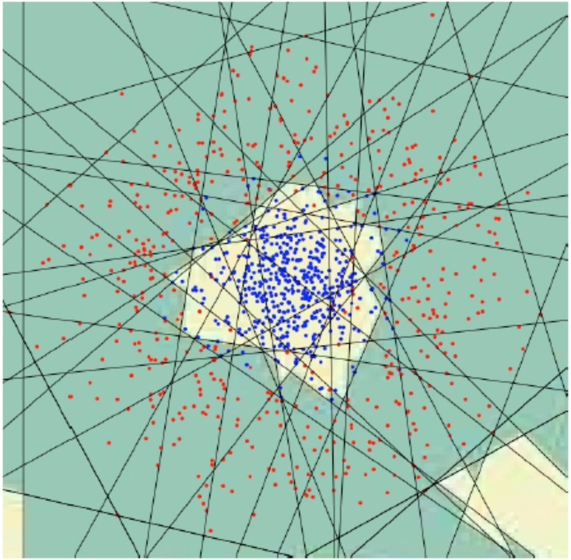
Assumption:

- easy to find *weak learners* that are “often” correct (slightly more than random)
- hard to find a *strong learner*, that is highly accurate

Combination of weak learners to generate a strong learner:

- focuses effort on hard-to-classify examples
- takes hard work off the learner
- learners only have to specialize in small areas
- can identify outliers

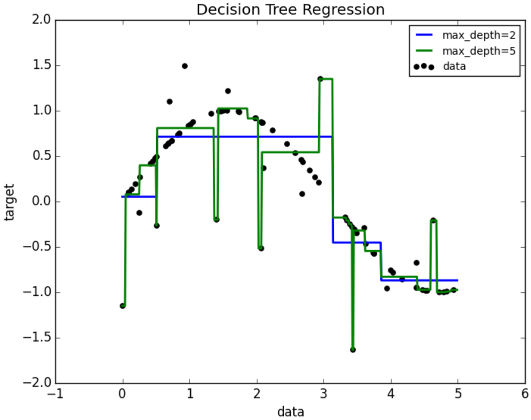
How does boosting help?



Gradient boosting

- generalizes AdaBoost to optimize an arbitrary differentiable loss function

→ inspired from regression



Gradient boosting

- boosting can be seen as an iterative functional gradient descent algorithm
- example: MSE for regression

→ residuals are the negative gradients of the squared error loss function

$$-\nabla_{F(x)}\left[\frac{1}{2}(y - F(x))^2\right] = y - F(x)$$

Gradient Boosting – Idea of the algorithm

- input: $\mathcal{D}$, T, and some specified loss function $L(y, F(x))$
- in each step, the previous model F_{t-1} is improved by adding an estimator h (weak learner)

→ $F_t = F_{t-1} + h(x)$

- How to find h?

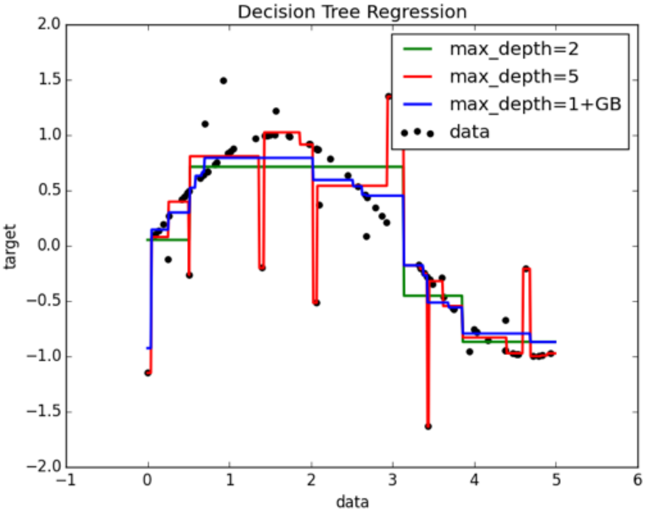
→ optimal h would imply

$$F_t = F_{t-1} + h(x) = y \Leftrightarrow h(x) = y - F_{t-1}$$

→ train h that approximates the negative gradient of L

- $F_t(x) = F_{t-1}(x) + \gamma_t h_t(x)$
- $\gamma_t = \arg\min_{\gamma} \sum_{i=1}^{m} L(y_i, F_{t-1}(x_i) + \gamma h_t(x_i))$

Gradient boosting for decision tree regression



Modern Ensemble Methods

Deep Ensembles

- Simple idea, make an ensemble of neural networks [Lakshminarayanan et al. 2017].
- Trained on the same data, models converge to different solutions due to random weight initialization.
- It has properties of good uncertainty quantification and out of distribution detection.
- It can estimate aleatoric (data) and epistemic (model) uncertainty. For aleatoric uncertainty in regression it uses the following loss:

$$-\log p_\theta(y_n|x_n) = 0.5\left(\log \sigma^2_\theta(x) + \frac{(y-\mu_\theta(x))^2}{\sigma^2_\theta(x)}\right)$$

Deep Ensembles

Classification

Take average of ensemble member output probabilities.

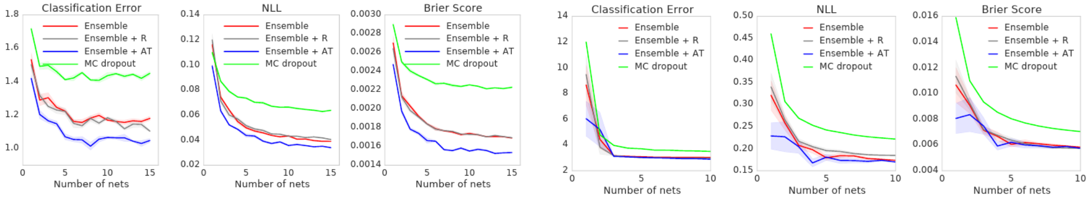
Regression

Each ensemble member outputs $\mu_{\theta_m}(x)$ and $\sigma^2_{\theta_m}(x)$, and they are combined as a mixture of gaussians $M^{-1}\sum\mathcal{N}(\mu_{\theta_m}(x),\sigma^2_{\theta_m}(x))$ represented as $\mathcal{N}(\mu_*(x),\sigma^2_*(x))$ where:

$$\mu_*(x) = M^{-1}\sum_m \mu_{\theta_m}(x)$$

$$\sigma^2_*(x) = M^{-1}\sum_m(\sigma^2_{\theta_m}(x) + \mu^2_{\theta_m}(x)) - \mu^2_*(x)$$

Deep Ensembles - Performance

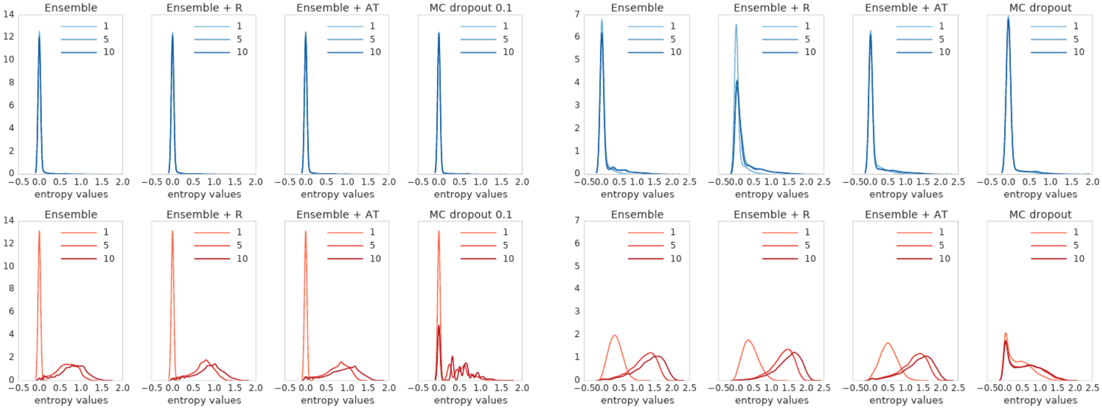


(a) MNIST dataset using 3-layer MLP

(b) SVHN using VGG-style convnet

Figure 2: Evaluating predictive uncertainty as a function of ensemble size M (number of networks in the ensemble or the number of MC-dropout samples): Ensemble variants significantly outperform MC-dropout performance with the corresponding M in terms of all 3 metrics. Adversarial training improves results for MNIST for all M and SVHN when $M = 1$, but the effect drops as M increases.

Deep Ensembles - Out of Distribution Detection



(a) MNIST-NotMNIST

(b) SVHN-CIFAR10

references

Literature

- “Meta-Learning – Concepts and Techniques”, O. Maimon, L. Rokach (eds.), In: *Data Mining and Knowledge Discovery Handbook* (2nd Ed.), 2010
- “Data Mining: Practical Machine Learning Tools and Techniques” (2nd Ed.), Ian H. Witten, Eibe Frank, Morgan Kaufmann, 2005
- “A short introduction to boosting”, Y. Freund, R. Schapire, N. Abe,In: JOURNAL-JAPANESE SOCIETY FOR ARTIFICIAL INTELLIGENCE, Vol. 14, p.771–780,1999
- Lakshminarayanan B, Pritzel A, Blundell C. Simple and scalable predictive uncertainty estimation using deep ensembles. NeurIPS 2017.

Thank You!
Please feel free to ask questions in the forums.