**DS603: Robust Machine Learning** **Autumn 2025**

# Lecture 5 — 13/08

*Lecturer: Arjun Bhagoji* *Scribe: Aditya Agrawal*

## 5.1 Input Representations

We start off by looking at whether the way we represent data for classification good enough?

A *feature map* $\phi : \mathbb{R}^d \to \mathbb{R}^p$ is simply a map which maps the input data into another space known as the *feature* or *embedding* space.

The k-class classification problem asks for a *map* $f : X \to [0,1]^k$ to denote *confidence* of a particular input lying in some class.

The standard classification algorithms directly operate on the input data while *representational learning* first maps raw data into *feature space* on which standard algorithms are applied. Representational learning is generally of two types:

(a) *Fixed* representations: These are explicit maps which are used on basis on some prior knowledge of data distribution

(b) *Learned* representations: These are maps which are *learnt* through Neural Networks, or other such techniques

A *kernel* is defined as a dot product in feature space:

$$k : X \times X \to \mathbb{R} \quad k(x,x') = \left\langle \phi(x), \phi(x') \right\rangle$$

The standard dot product arises when we use the linear feature map $\phi(x) = x$.

### 5.1.1 Feature maps from kernels

We ask the question of when do arbitrary kernel functions arise as a dot product between some feature vectors.

**Theorem 5.1.** *If $k(x,x')$ is real valued and positive semi definite, then there exists an inner product space $\mathscr{X}$ and feature map $\phi : \mathbb{R}^d \to \mathscr{X}$ such that $k(x,x') = \langle \phi(x), \phi(x') \rangle$*

We simply define $\mathscr{X}$ as the set of all maps $f : \mathbb{R}^d \to \mathbb{R}$ with an appropriate inner product and the feature map $\phi : \mathbb{R}^d \to \mathscr{X}$ as $\phi(x)(.) = k(x,.)$. We also have to validate the existence of the inner product on the function space but the positive semi definiteness of the product will allow us to do so.

### 5.1.2 Examples of Kernels

(a) Polynomial Kernels : $k(x,x') = \langle x, x' \rangle^d$

(b) Gaussian Kernels : $k(x,x') = \mathrm{e}^{-\|x-x'\|^2/2\sigma^2}$

### 5.1.3 The Kernel Trick

Well, the representations we dealt with earlier either fixed or learned were into finite dimensional spaces. But fortunately, in all our optimization equations, the mentions of $\phi(x)$ were all as dot products of the form $\phi(x)^T\phi(x')$, which can instead be replaced by $\tilde{k}(x,x')$ leading to a feature map $\tilde{\phi} : \tilde{\phi}(x)(.) = \tilde{k}(x,.)$ where this feature map could be infinite dimensional!!

## 5.2 Anomaly Detection

We come back to the original question of anomaly detection and propose another method.

### 5.2.1 One Class SVM

Consider the following constrained parametric optimization problem:

$$\begin{array}{ll} \min\limits_{w,\gamma,\rho} & \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}(\sum_{i=1}^{n}\gamma_i) - \rho \\ \text{subject to} & \begin{cases} \langle w, \phi(x_i)\rangle \geqslant \rho - \gamma_i \quad i = 1,\ldots,n, \\ \gamma_i \geqslant 0 \quad i = 1,\ldots,n, \end{cases} \end{array}$$

where $\nu$ is a hyperparameter which asymptotically represents the fraction of anomalies. At test time, we define the hypothesis maps $f, h$ as the following:

$$f_\theta(x) = \langle w, \phi(x_i)\rangle - \rho$$
$$h_\theta(x) = sgn(f_\theta(x))$$

**Note:** Finally, the map $h$ is the one which maps the input into a class 0 or 1 where 0 indicates an anomaly.

### 5.2.2 HW Problems

1. Derive the primal optimization problem of the *lagrangian* of a standard one class SVM machine for both the soft and the hard cases using the KKT conditions
2. Derive the same for the $\nu$-SVM as well
3. Derive the dual of these primal objective functions as well

### 5.2.3 SLT- like Guarantees on the One Class SVM

**Theorem 5.2.** *(informally) Let $R_{w,\gamma} = \{x|f_\theta(x) \geqslant 0\}$. Then for all $\delta > 0$, with probability $1-\delta$ over the draws $\{x_i\}_{i=1}^n$, for any $B > 0$, we get*

$$\mathop{\mathbb{P}}_{x\sim\mathbb{P}^+}[x|x \notin R_{w,\rho-B}] \leqslant \frac{2}{n}\left(k(n,B,\|W\|,\rho) + \log_2(\frac{n^2}{2\delta})\right)$$

where $k$ is some non-negative function of problem parameters

## 5.3 OOD Detection

The OOD problem or the *Out Of Distribution* problem is the supervised case of the anomaly detection problem where we are now also provided labels on the input draws. Our key question is to understand how OOD detection is done nowadays.

### 5.3.1 Deep OCSVM or Deep SVDD

In this methodology, we use a learned feature map $f_\theta(x)$, where $\theta$ paramterizes some neural network, with $\theta$ learned is jointly over the supervised learning component and the anomaly detection $(R, c)$ or $(w, \gamma, \rho)$. The anomaly detection optimization problem is given as

$$\min_{R,c,\theta} \quad R^2 + \frac{1}{\nu n}\sum_{i=1}^{n} \max(0, \|\phi_\theta(x_i) - c\|^2 - R^2) + Reg(\theta)$$

Here, typically some regularization is put on $\theta$ since otherwise it tends to fit maps such that all points in feature space start falling within the sphere. It is typically trained through stochastic gradient descent

### 5.3.2 MSP and ODIN

MSP stands for *Maximum Softmax Probability* and works on the principle that the NN holds enough discriminative power to classify in distribution samples confidently. This leads to the formulation where we call an input out of distribution when the NN cannot classify it confidently.

ODIN or *Out of Order Distribution for Neural Networks* enhances MSP by preprocessing data smartly and adding a temperature control on the softmax performed. It works in the following way:

1. Preprocesses input $x$ as

$$x' = x + \varepsilon * sgn\left(\nabla_x log(f_\theta^{\hat{y}}(x, \tau))\right)$$
$$f_\theta^i(x, \tau) = \frac{e^{\frac{l_i(x)}{\tau}}}{\sum_{i=1}^n e^{\frac{l_i(x)}{\tau}}}$$

   where $l_i(x)$ is the logit output of neural network $l : \mathbb{R}^d \to \mathbb{R}^p$

2. Defines the anomaly hypothesis map $h$ as

$$s_\theta(x) = \max f_\theta^i(x, \tau)$$
$$h_\theta(x) = 2 * (\mathbf{1}[s_\theta(x) < \delta] - 1) - 1$$

### 5.3.3 Outlier Exposure

Well the issue with our current methods is that they are overfitted on in distribution data. It is very likely that out of distribution data looks very different and might lead the overfit classifier to falsely label them as being in distribution. To prevent this, we perform outlier exposure by slightly tweaking our NN loss functions. Our goal is to now minimize

$$\min_\theta \quad \mathop{\mathbb{E}}_{x\sim\mathbb{P}^+} l(h_\theta(x), y) + \lambda \mathop{\mathbb{E}}_{x\sim\mathbb{P}^-} l_{OE}(f_\theta(x)$$

Here $\lambda$ controls the rate of outliers we expect in testing and $l_{OE}$ is the loss we want to calculate over outlier data. We generally assume that the out of order ditribution is uniform in some sense.

### 5.3.4 Enhancing MSP

We essentially want to temper the confidence of our classifier, by saying ”Hey! Don’t be so confident about your predictions, they might be on out of distribution data!!”. We design two outlier two loss functions for the same.

1. $$l_{OE}(x) = -\sum_{j=1}^{|C|} \frac{1}{|C|} \log(f_\theta^j(x))$$

   We *assume* that a good classification is a confident classification.

2. To make a distinction between a *good* and *confident* classification we artificially introduce a confidence estimation branch $c(x) \in [0,1]$ and set $l_{OE}(x) = -\log(c(x))$.
   During training we then try to minimize

   $$l(x) = -\sum_{j=1}^{|C|} \log(cf_\theta(x_j) + (1-c)y_j)y_j - B\log(c(x))$$

   Essentially, what we try to say is that if we are not confident about whether the input is from distribution, then our NN doesn’t pay a training loss, however we pay a confidence loss. If we are confident, then we incur a training loss as well. This flips our notion from good $\implies$ confident to good $\impliedby$ confident

### 5.3.5 Grad Norm KL Divergence

Instead of looking at the maximum softmax probability as a measure of confidence, we look at the gradient of the norm of the KL divergence as such a measure. The intuition is that low gradient norm implies points close to margins and so more likely to be OOD. We define this using

$$\begin{aligned} D_{KL}(U||f_\theta(x)) &= \sum_{j=1}^{|C|} \frac{1}{|C|} \log\left(\frac{1/|C|}{f_\theta^j(x)}\right) \\ &= -\frac{1}{|C|}\sum_{j=1}^{|C|} \log\left(|C| f_\theta^j(x)\right) \end{aligned}$$

Our hypothesis function $h$ then is $h_\theta(x) = 2 * (\mathbf{1}[\|\nabla_\theta D_{KL}\|_2 < \delta] - 1) - 1$

### 5.3.6 Cons of the above methods

While the above methods are very intuitive and make sense, most of their guarantees are empirical and not theoretical. They have also been designed in an adhoc manner with not much theoretical basis as to why they might be good. It might be useful to try and get some SLT like bounds on these like the one we had for the one class SVM.