Lecture 17: Kernel Theory

With kernel ridge regression (KRR), we learned <u>three</u> concepts:

1) Primal and dual formulations of a model

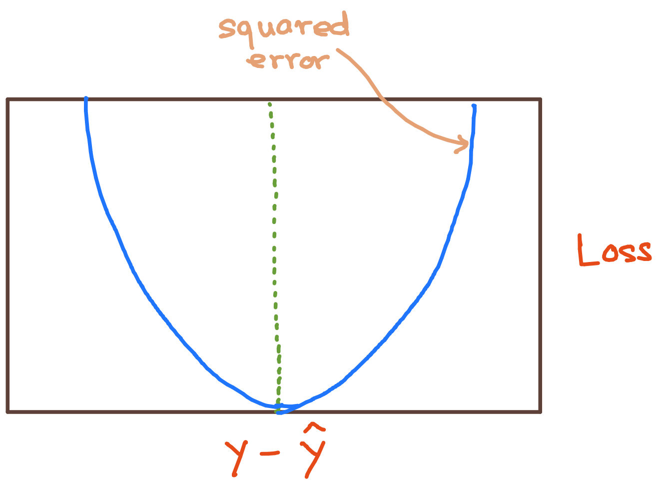
- Primal formulation expresses the model in terms of $\underline{\theta} \in \mathbb{R}^d$
- Dual formulation uses $\underline{\alpha} \in \mathbb{R}^N$ ($N \leftarrow$ size of training data set), and does not depend on the value of 'd'
- Both formulations are mathematically equivalent
    - Primal formulation is useful if $N > d$
    - Dual formulation is useful if $d > N$

2) We introduced kernels $K(\underline{x}, \underline{x}')$ that allows us to let $d \to \infty$ without explicitly formulating an infinite vector of non-linear transformations $\underline{\phi}(\underline{x})$

- The dual formulation is particularly useful when using kernel methods, since the dimension of $\underline{\theta}$ in the primal formulation could be very large

3) We can different loss functions (and included L_2-regularization)

- KRR makes use of squared error loss



Kernel theory

Lets look a bit more into kernels

- Kernel was defined as being any function that takes in two arguments and returns a scalar

- We also suggested that we will restrict ourselves to PSD (positive semi-definite) kernels

- Vanilla kNN $\longrightarrow$ kernel kNN (provides a variety of distance metrics)
  - Recall that vanilla kNN constructs prediction for $\underline{x}_*$ by taking the average or a majority vote among the k "nearest" neighbours
  - In its standard form, "nearest" was defined by the Euclidean distance
  - Euclidean distance between 2 points $\underline{x}$ and $\underline{x}'$: $\|\underline{x} - \underline{x}'\|_2$ (always +ve)

Euclidean distance between 2 points $\underline{x}$ and $\underline{x}'$: $\|\underline{x} - \underline{x}'\|_2$ (always +ve)

- Since Euclidean distance is positive, we can consider squared Euclidean distance instead

$$\|\underline{x} - \underline{x}'\|_2^2 = (\underline{x} - \underline{x}')^T (\underline{x} - \underline{x}')$$

$$= \underline{x}^T \underline{x} + \underline{x}'^T \underline{x}' - 2\underline{x}^T \underline{x}'$$

For many kernels, these terms are mostly constants (e.g. RBF kernel)

Define a kernel $K(\underline{x}, \underline{x}') = \underline{x}^T \underline{x}'$

$$= K(\underline{x}, \underline{x}) + K(\underline{x}', \underline{x}') - \underbrace{2K(\underline{x}, \underline{x}')}$$

this term is more interesting

this term determines how close any two points are

$K(\underline{x}, \underline{x}')$ takes a large value if $\underline{x}$ and $\underline{x}'$ are close

- In kernel kNN, $K(\underline{x}, \underline{x}')$ can be replaced with any PSD kernel

- How can you use vanilla kNN where Euclidean distance has no natural meaning?

Example: Distance between words which reflect sentiment

Word	Sentiment
Tremendous	Positive
Horrific	Negative
Outrageous	Negative

$x_* =$ Horrendous

$K = 1 \longrightarrow$ Positive

$K = 3 \longrightarrow$ Negative

- what could be the label for "horrendous"?
- One may think of converting the input space to numbers first and then use Euclidean distance
- An easier way to compare is using, for ex, Levenstein distance (LD), which is the number of single-character edits needed to transform one word (string) into another

- One can construct a kernel as $K(\underline{x}, \underline{x}') = \exp\left(-\frac{\left(LD(x, x')\right)^2}{2\ell^2}\right)$

  to implement kernel kNN (instead of vanilla kNN)

Lessons learned about kernels so far

- A kernel defines how close / similar any two points are
  - If $\kappa(\underline{x}_i, \underline{x}_*) > \kappa(\underline{x}_j, \underline{x}_*)$, then $\underline{x}_*$ is more similar to $\underline{x}_i$ than $\underline{x}_j$
  - It also implies that prediction $\hat{y}(\underline{x}_*)$ is most influenced by the training data points that are closest to $\underline{x}_*$
  - Therefore, a kernel plays an important role of determining the individual influence of each training data point when making a prediction
- No need to bother about the inner product $\underline{\phi}(\underline{x})^T \underline{\phi}(\underline{x}')$ once we have introduced the kernel $\kappa(\underline{x}, \underline{x}')$

Lessons learned about kernels so far

- Choice of a kernel corresponds to preference for certain types of functions

  - For example, the squared exponential (or RBF) kernel

    $$K(\underline{x}, \underline{x}') = \exp\left(-\frac{\|\underline{x}-\underline{x}'\|_2^2}{2\ell^2}\right)$$

    implies a preference for smooth functions

  - In primal formulation, we choose features $\underline{\phi}(\underline{x})$ which will reflect the type of transformations we want to introduce. This choice is reflected to some extent in choosing kernels in the dual formulation

A machine learning engineer must choose a kernel wisely and should not simply resort to 'default' choices

What are valid choices of kernels?

- We already know that kernels are a way to represent non-linear feature transformation $\underline{\phi}(\underline{x})$

$$K(\underline{x}, \underline{x}') = \underline{\phi}(\underline{x})^T \underline{\phi}(\underline{x}')$$

- Question: Does an arbitrary kernel $K(\underline{x}, \underline{x}')$ always correspond to a feature transformation $\underline{\phi}(\underline{x})$?

  - The question is primarily of theoretical nature
  - Practically, it matters very less whether a kernel $K(\underline{x}, \underline{x}')$ admits a factorization $K(\underline{x}, \underline{x}') = \underline{\phi}(\underline{x})^T \underline{\phi}(\underline{x}')$ or not
  - Furthermore, the factorization has no direct correspondence to how well the kernel will perform in terms of E_{new}, which still has to be evaluated using cross-validation

Question: Does an arbitrary kernel $\kappa(\underline{x}, \underline{x}')$ always correspond to a feature transformation $\underline{\phi}(\underline{x})$?

Answer: Yes, if the kernel $\kappa(\underline{x}, \underline{x}')$ is PSD (positive semi-definite)
(no negative eigen-values)

Recall that a kernel is PSD if the Gram matrix $\underline{\underline{K}}(\underline{\underline{X}}, \underline{\underline{X}})$ is PSD for any $\underline{\underline{X}}$

- It holds that any kernel $\kappa(\underline{x}, \underline{x}')$ that is defined as an inner product between feature vectors $\underline{\phi}(\underline{x})$ is always PSD

$$
\begin{aligned}
\kappa(\underline{x}, \underline{x}') &= \underline{\phi}(\underline{x})^T \underline{\phi}(\underline{x}') \\
&= \langle \phi(\underline{x}), \phi(\underline{x}') \rangle
\end{aligned}
$$

$\langle \cdot, \cdot \rangle \leftarrow$ inner product

Show $\underline{v}^T \underline{\underline{K}}(\underline{\underline{X}}, \underline{\underline{X}}) \underline{v} \geq 0$ for any vector v (do yourself)

$$
\underbrace{\underline{\phi}(\underline{x})}_{\text{feature vector}} \xrightarrow{\text{inner product}} \underbrace{\kappa(\underline{x}, \underline{x}')}_{\text{PSD}}
$$

Question: Does an arbitrary kernel $K(\underline{x}, \underline{x}')$ always correspond to a feature transformation $\underline{\phi}(\underline{x})$?

Answer: Yes, if the kernel $K(\underline{x}, \underline{x}')$ is PSD (positive semi-definite)
(no negative eigen-values)

- It holds that any kernel $K(\underline{x}, \underline{x}')$ that is defined as an inner product between feature vectors $\underline{\phi}(\underline{x})$ is always PSD

$$\underbrace{\underline{\phi}(\underline{x})}_{\text{feature vector}} \xrightarrow{\text{inner product}} \underbrace{K(\underline{x}, \underline{x}')}_{\text{PSD}}$$

- The other direction also holds true, that is, for any PSD Kernel $K(\underline{x}, \underline{x}')$ there always exist a feature vector $\underline{\phi}(\underline{x})$ such that $K(\underline{x}, \underline{x}')$ can be written as its inner product

$$\underbrace{\underline{\phi}(\underline{x})}_{\text{feature vector}} \longleftarrow \underbrace{K(\underline{x}, \underline{x}')}_{\text{if PSD}}$$

- The other direction also holds true, that is, for any PSD kernel $K(\underline{x}, \underline{x}')$ there always exist a feature vector $\underline{\phi}(\underline{x})$ such that $K(\underline{x}, \underline{x}')$ can be written as its inner product

$$\underbrace{\underline{\phi}(\underline{x})}_{\text{feature vector}} \longleftarrow \underbrace{K(\underline{x}, \underline{x}')}_{\text{if PSD}}$$

- It can be shown that for any PSD kernel, it is possible to construct a function space, more specifically a Hilbert space, that is spanned by a feature vector $\underline{\phi}(\underline{x})$ s.t. $K(\underline{x}, \underline{x}') = \underline{\phi}(\underline{x})^T \underline{\phi}(\underline{x}')$

  - There are multiple ways to construct a Hilbert space space spanned by $\underline{\phi}(\underline{x})$. One of the ways is using the so-called reproducing kernel Hilbert space (RKHS) mapping

A brief introduction to Reproducing Kernel Hilbert Spaces (RKHS) [Digression]

- Euclidean space is a space of vectors equipped with inner products between vectors

- Hilbert space is a generalization of Euclidean space to functions (which can be treated as infinite dimensional vectors). It allows inner product between functions

  space of functions with inner product

- A Hilbert space H is called the RKHS if there exists a kernel $K(\underline{x}, \underline{x}')$ with the reproducing property that

$$f(\underline{x}') = \langle f(\cdot), K(\cdot, \underline{x}') \rangle \quad \forall f \in H, \ \forall \underline{x}'$$

  - If we set $f(\cdot) = K(\cdot, \underline{x})$, then

$$\langle K(\cdot, \underline{x}), K(\cdot, \underline{x}') \rangle = K(\underline{x}, \underline{x}')$$

This reproducing property is the main building block of RKHS. This RKHS is spanned by the corresponding feature $\underline{\phi}(\underline{x})$ of kernel $K(\underline{x}, \underline{x}')$

Question: Does an arbitrary kernel $K(\underline{x}, \underline{x}')$ always correspond to a feature transformation $\underline{\phi}(\underline{x})$?

Answer: Yes, if the kernel $K(\underline{x}, \underline{x}')$ is PSD (positive semi-definite)
(no negative eigen-values)

$$\underbrace{\underline{\phi}(\underline{x})}_{\text{feature vector}} \xrightarrow{\text{inner product}} \underbrace{K(\underline{x}, \underline{x}')}_{\text{PSD}}$$

$$\underbrace{\underline{\phi}(\underline{x})}_{\text{feature vector}} \longleftarrow \underbrace{K(\underline{x}, \underline{x}')}_{\text{if PSD}}$$

(spans a RKHS)

- A given Hilbert space uniquely defines a kernel, but for a kernel there exists multiple Hilbert spaces which correspond to it

$$\underline{\phi}_1(\underline{x}) \longleftarrow \underbrace{K(\underline{x}, \underline{x}')}_{\text{if PSD}} \longrightarrow \underline{\phi}_2(\underline{x})$$

$\underline{\phi}_1(\underline{x})$: feature vector (spans a RKHS)

$\underline{\phi}_2(\underline{x})$: feature vector

E.g. $K(\underline{x}, \underline{x}') = \underline{x}^T \underline{x}'$

$$\phi_1(\underline{x}) = x \qquad \phi_2(\underline{x}) = \begin{bmatrix} x/\sqrt{2} \\ x/\sqrt{2} \end{bmatrix}$$

(one-dimensional) (two-dimensional)

Examples of kernels

- Linear kernel

c ≥ 0 to maintain PSD property

$$k(\underline{x}, \underline{x}') = \underline{x}^T \underline{x}' + c$$

hyperparameter

- Simplest kernel
- Used when the number of features are already large

- Polynomial kernel

polynomial order (integer)

$$K(\underline{x}, \underline{x}') = (\underline{x}^T \underline{x}' + c)^{d-1}$$

hyperparameter

- The polynomial corresponds to a finite-dimensional feature vector $\underline{\phi}(\underline{x})$ of monomials up to order $d-1$

- Squared exponential (RBF) kernel

$$K(\underline{x}, \underline{x}') = \exp\left(-\frac{\|\underline{x} - \underline{x}'\|_2^2}{2\ell^2}\right)$$

$\ell \geq 0$

Commonly used kernel

- $\ell \leftarrow$ hyperparameter (called lengthscale)
- This kernel has a local nature because $K(\underline{x}, \underline{x}') \to 0$ as $\|\underline{x} - \underline{x}'\| \to \infty$
- Infinite-dimensional features

- Matérn family of kernels

Modified Bessel function

$$\kappa(\underline{x}, \underline{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\, \|\underline{x} - \underline{x}'\|_2}{\ell} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}\, \|\underline{x} - \underline{x}'\|}{\ell} \right)$$

Gamma function

with hyperparameters $\ell > 0$, $\nu > 0$

smoothness parameter

exponential kernel

Commonly used

$$\nu = \frac{1}{2} \Rightarrow \quad \kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2}{\ell}\right),$$

$$\nu = \frac{3}{2} \Rightarrow \quad \kappa(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\sqrt{3}\|\mathbf{x} - \mathbf{x}'\|_2}{\ell}\right) \exp\left(-\frac{\sqrt{3}\|\mathbf{x} - \mathbf{x}'\|_2}{\ell}\right),$$

$$\nu = \frac{5}{2} \Rightarrow \quad \kappa(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\sqrt{5}\|\mathbf{x} - \mathbf{x}'\|_2}{\ell} + \frac{5\|\mathbf{x} - \mathbf{x}'\|_2^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}\|\mathbf{x} - \mathbf{x}'\|_2}{\ell}\right)$$

As $\nu \to \infty$, Matérn kernel equals squared exponential kernel

- Rational Quadratic kernel

$$k(\underline{x}, \underline{x}') = \left(1 + \frac{\|\underline{x} - \underline{x}'\|_2^2}{2a\ell^2}\right)^{-a} \qquad \left.\begin{matrix} \ell > 0 \\ a > 0 \end{matrix}\right] \text{hyperparameter}$$

- Squared exponential, Matérn, and rational quadratic kernel are examples of stationary kernels, since they are functions of $(\underline{x} - \underline{x}')$

- An example of non-PSD kernel is the sigmoid kernel

$$k(\underline{x}, \underline{x}') = \tanh(a\, \underline{x}^T \underline{x}' + b)$$

$$\underbrace{a > 0 \quad b < 0}_{\text{hyperparameters}}$$

Techniques for constructing new kernels

Given valid kernels $k_1(\underline{x}, \underline{x}')$ and $k_2(\underline{x}, \underline{x}')$, you can construct new kernels the following ways:

$$k(\underline{x}, \underline{x}') = c\, k_1(\underline{x}, \underline{x}') \qquad c > 0 \text{ is a constant}$$

$$= f(\underline{x})\, k_1(\underline{x}, \underline{x}')\, f(\underline{x}') \qquad f(\cdot) \leftarrow \text{any function}$$

$$= q\left(k_1(\underline{x}, \underline{x}')\right) \qquad \text{where } q(\cdot) \text{ is a polynomial with non-negative coefficients}$$

$$= \exp\left(k_1(\underline{x}, \underline{x}')\right)$$

$$= k_1(\underline{x}, \underline{x}') + k_2(\underline{x}, \underline{x}') \qquad \text{(Addition)}$$

$$= k_1(\underline{x}, \underline{x}')\, k_2(\underline{x}, \underline{x}') \qquad \text{(Multiplication)}$$

Kernel-based Classification

- Using kernels, we have seen kernel ridge regression

- The main ideas of the dual formulation, kernel trick, and change of loss function can be applied to classification as well

- Earlier, for binary classification $y \in \{-1, 1\}$, we saw logistic regression

Logistic model with margin formulation

Logistic loss

$$y = \text{sign}(\underline{\theta}^T \underline{x}) \quad \text{with} \quad L = \ln\left(1 + e^{-y\,\underline{\theta}^T \underline{x}}\right)$$

Margin of a classifier for a datapoint $(\underline{x}, y)$

$= y \cdot f(\underline{x})$

- To obtain a kernelized version of logistic regression, certain modifications are to be made:

$$\underline{x} \longrightarrow \underline{\phi}(\underline{x})$$

$$L = \ln\left(1 + e^{-y\,\underline{\theta}^T \underline{\phi}(\underline{x})}\right) + \lambda \|\theta\|_2^2$$

added to allow dual formulation using Representer's theorem

Support Vector Classification

- Unlike kernel ridge regression, kernel logistic regression is not popular

- For assification, SVC is very popular

  - It is the classification counterpart of SVR

  - Both have sparse dual parameter vectors

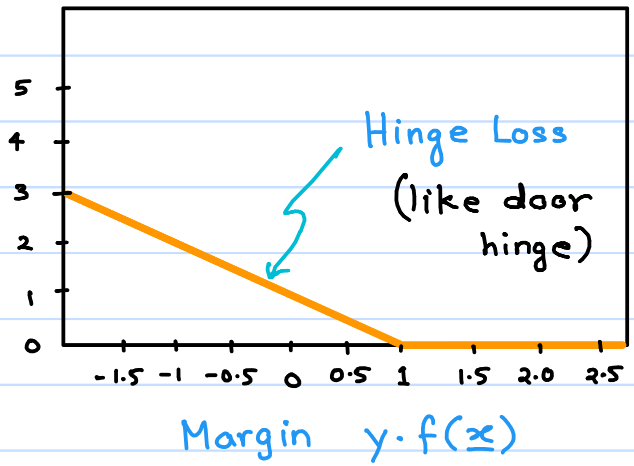
- KRR $\longrightarrow$ SVR was obtained via change of loss function

  Similarly, we use the hinge loss instead of logistic loss in SVC

- Recall hinge loss (from Lecture 11 b)

$$\mathcal{L}(y \cdot f(\underline{x})) = \begin{cases} 1 - y \cdot f(\underline{x}) & \text{for } \overbrace{y \cdot f(\underline{x})}^{\text{Margin}} \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$= \max\{0,\ 1 - y \cdot f(\underline{x})\}$$



- In SVC, $f(\underline{x}) = \underline{\theta}^T \underline{\phi}(\underline{x})$, so the hinge loss will be

$$L(\underline{x}, y, \underline{\theta}) = \begin{cases} 1 - y \cdot \underline{\theta}^T \underline{\phi}(x) & \text{for } y\,\underline{\theta}^T \underline{\phi}(\underline{x}) < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$= \max\{0,\ 1 - y\,\underline{\theta}^T \underline{\phi}(\underline{x})\}$$

- Just like the ϵ-insensitive loss, the main advantage of hinge loss comes when we look at the dual formulation using $\underline{\alpha}$, instead of the primal formulation with $\underline{\theta}$

- Primal formulation with $\underline{\theta}$

non-differentiable due to max fun.

$$\hat{\underline{\theta}} = \underset{\underline{\theta}}{\text{argmin}}\ \frac{1}{N} \sum_{i=1}^{n} \max\{0,\ 1 - y^{(i)}\,\underline{\theta}^T \underline{\phi}(x^{(i)})\} + \lambda \|\underline{\theta}\|_2^2$$

The feature vector does not appear as $\underline{\phi}^T(\underline{x})\,\underline{\phi}(\underline{x}')$ in primal form

- The kernel trick cannot be applied in primal form

- Therefore, we will consider the dual form. The dual form can be obtained by using slack variables to replace the "max" in objective function and then constructing Lagrangian

$$\underset{\underline{\theta}}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N} \max\{0,\ 1 - y^{(i)}\underline{\theta}^T\underline{\phi}(\underline{x}^{(i)})\} + \lambda\|\underline{\theta}\|_2^2$$

equivalent

$$\underset{\underline{\theta},\,\underline{\xi}}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N} \xi_i + \lambda\|\underline{\theta}\|_2^2$$

$$\text{subject to} \quad \xi_i \geq 1 - y^{(i)}\underline{\theta}^T\underline{\phi}(\underline{x}^{(i)})$$

$$\xi_i \geq 0 \qquad (i = 1, 2, \cdots, N)$$

In optimization, slack variable transforms an inequality constraint to an equality constraint and non-negativity constraint on the slack variable

ex. $\underline{\xi} \geq 0$

$\underline{A}\,\underline{x} \leq \underline{b}$

$\downarrow$

$\underline{A}\,\underline{x} + \underline{s} = \underline{b}$

- Construct Lagrangian

$$L(\underline{\theta}, \underline{\xi}, \underbrace{\underline{\beta}, \underline{\gamma}}_{\text{Lagrange multipliers}}) = \frac{1}{N}\sum_{i=1}^{N}\xi_i + \lambda\|\underline{\theta}\|_2^2 - \sum_{i=1}^{N}\beta_i\left(\xi_i + y^{(i)}\underline{\theta}^T\underline{\phi}(\underline{x}^{(i)}) - 1\right) - \sum_{i=1}^{N}\gamma_i\xi_i$$

$\beta_i,\ \gamma_i \geq 0$

- According to Lagrange's duality theory, instead of solving this

$$\underset{\underline{\theta}}{\text{minimize}} \quad \frac{1}{N}\sum_{i=1}^{N} \max\{0, 1 - y^{(i)}\underline{\theta}^T\underline{\phi}(x^{(i)})\} + \lambda\|\theta\|_2^2$$

we can minimize this w.r.t $\underline{\theta}$ and $\underline{\xi}$ and maximize it w.r.t. $\underline{\beta}$ and $\underline{\gamma}$

$$\boxed{L(\underline{\theta}, \underline{\xi}, \underbrace{\underline{\beta}, \underline{\gamma}}_{\text{Lagrange multipliers}}) = \frac{1}{N}\sum_{i=1}^{N}\xi_i + \lambda\|\underline{\theta}\|_2^2 - \sum_{i=1}^{N}\beta_i\left(\xi_i + y^{(i)}\underline{\theta}^T\underline{\phi}(\underline{x}^{(i)}) - 1\right) - \sum_{i=1}^{N}\gamma_i\xi_i \qquad \beta_i, \gamma_i \geq 0}$$

- Two necessary conditions for optimality are

$$\frac{\partial}{\partial\underline{\theta}} L(\underline{\theta}, \underline{\xi}, \underline{\beta}, \underline{\gamma}) = 0 \quad , \quad \frac{\partial}{\partial\underline{\xi}} L(\underline{\theta}, \underline{\xi}, \underline{\beta}, \underline{\gamma}) = 0$$

- We can use the fact that $\|\underline{\theta}\|_2^2 = \underline{\theta}^T\underline{\theta}$, and hence $\frac{\partial}{\partial\underline{\theta}}\|\underline{\theta}\|_2^2 = 2\underline{\theta}$

$$L(\underline{\theta}, \underline{\xi}, \underbrace{\underline{\beta}, \underline{\gamma}}_{\text{Lagrange multipliers}}) = \frac{1}{N}\sum_{i=1}^{N} \xi_i + \lambda \|\underline{\theta}\|_2^2 - \sum_{i=1}^{N} \beta_i \left(\xi_i + y^{(i)} \underline{\theta}^T \underline{\phi}(\underline{x}^{(i)}) - 1\right) - \sum_{i=1}^{N} \gamma_i \xi_i \qquad \beta_i, \gamma_i \geq 0$$

- Using $\frac{\partial}{\partial \underline{\theta}} L(\underline{\theta}, \underline{\xi}, \underline{\beta}, \underline{\gamma}) = 0$, we get

$$\underline{\theta} = \frac{1}{2\lambda} \sum_{i=1}^{N} y^{(i)} \beta_i \underline{\phi}(x^{(i)}) \quad \text{— (a)}$$

- Using $\frac{\partial}{\partial \underline{\xi}} L(\underline{\theta}, \underline{\xi}, \underline{\beta}, \underline{\gamma}) = 0$, we get

$$\gamma_i = \frac{1}{N} - \beta_i \quad \text{— (b)}$$

- Inserting (a) & (b) in the Lagrangian and scaling it with $1/2\lambda$, we get

$$L(\underline{\theta}, \underline{\xi}, \underline{\beta}, \underline{\gamma}) = \frac{1}{N}\sum_{i=1}^{N} \xi_i + \lambda \|\underline{\theta}\|_2^2 - \sum_{i=1}^{N} \beta_i \left(\xi_i + y^{(i)} \underline{\theta}^T \underline{\phi}(\underline{x}^{(i)}) - 1\right)$$