Introduction to Machine Learning (67577)

Exercise 5
Kernel Methods, Unsupervised Learning & Gradient-Based Learning

Second Semester, 2023

Contents

1 Theoretical Part **2**
- 1.1 Kernels . . . 2
- 1.2 PCA . . . 3
- 1.3 Convex optimization . . . 3
- 1.4 Sub-gradients for Soft-SVM Objective . . . 3

2 Practical Part **4**
- 2.1 Gradient Descent . . . 4
  - 2.1.1 Comparing Fixed learning rates . . . 5
  - 2.1.2 Comparing Exponentially Decaying learning rates (Optional) . . . 5
- 2.2 Minimizing Regularized Logistic Regression . . . 6

Submission

Please make sure to follow the general submission instructions available on the course website. In addition, for the following assignment, submit a single `ex5_ID.tar` file containing:

- An `Answers.pdf` file with the answers for all theoretical and practical questions (include plotted graphs *in* the PDF file).
- The following python files (without any directories): `gradient_descent.py`, `learning_rate.py` , `modules.py`, `logistic_regression.py` and `gradient_descent_investigation.py`

The `ex5_ID.tar` file must be submitted in the designated Moodle activity prior to the date specified *in the activity*.

- Late submissions will result in reduction of points.
- Plots included as separate files will be considered as not provided.

1 Theoretical Part

1.1 Kernels

Based on Lecture 8 and Recitation 10

1. Let $k(\mathbf{x},\mathbf{x}')$ be a valid PSD kernel. Provide a valid PSD kernel $\tilde{k}(\mathbf{x},\mathbf{x}')$, constructed from k, which is guaranteed to be normalized. That is, for all $\mathbf{x}$ it holds that $\tilde{k}(\mathbf{x},\mathbf{x}) = 1$. Prove your answer.
2. Consider a data set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$, and a feature map $\psi : \mathbb{R}^d \to \mathcal{F}$ where $\mathcal{F}$ is some feature space. Give an example of a data set S and a feature map ψ such that S is not linearly separable in $\mathbb{R}^d$ (for $d \geq 2$) but that the transformed data set $S_\psi = \{(\psi(\mathbf{x}_i), y_i)\}_{i=1}^m$ is linearly separable in $\mathcal{F}$.
3. $k_1(\mathbf{x},\mathbf{y})$ and $k_2(\mathbf{x},\mathbf{y})$ are valid kernels, then:

$$k_\times(\mathbf{x},\mathbf{y}) = k_1(\mathbf{x},\mathbf{y}) \cdot k_2(\mathbf{x},\mathbf{y})$$

is also a valid kernel. To prove this we'll use the fact that valid kernels are positive semi-definite.

You may find the following identities helpful (but don't have to use them):

$$\mathbf{x}^\top A\mathbf{y} = Tr\left[\mathbf{x}^\top A\mathbf{y}\right] = Tr\left[\mathbf{y}\mathbf{x}^\top A\right] \tag{1}$$

$$Tr[AB] = \sum_i [AB]_{ii} = \sum_i \sum_j A_{ij}B_{ji} \tag{2}$$

where $\mathbf{x}$ and $\mathbf{y}$ are vectors while A and B are matrices.

(a) Let $k(\mathbf{x},\mathbf{y})$ be a valid kernel and suppose that K is the kernel's Gram matrix over some finite set of points $\{\mathbf{x}_i\}_{i=1}^N$, such that $K_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$. Show that for any finite set of N points, there exists some function $f : \mathcal{X} \mapsto \mathbb{R}^N$ such that:

$$k(\mathbf{x}_i,\mathbf{x}_j) = f^\top(\mathbf{x}_i) f(\mathbf{x}_j) \tag{3}$$

where $\mathcal{X}$ is space of the points $\mathbf{x}_i$. Using this fact, show that:

$$k_1(\mathbf{x},\mathbf{y}) \cdot k_2(\mathbf{x},\mathbf{y}) = \sum_i \sum_j g_i(\mathbf{x}) f_j(\mathbf{x}) f_j(\mathbf{y}) g_i(\mathbf{y}) \tag{4}$$

where $k_1(\cdot,\cdot)$ and $k_2(\cdot,\cdot)$ are valid kernels, and some functions $f,g:\mathcal{X}\mapsto\mathbb{R}^N$, where $f_i(\mathbf{x})$ denotes the ith index of the output of $f(\mathbf{x})$.

(b) Conclude that $k_\times(\mathbf{x},\mathbf{y}) = k_1(\mathbf{x},\mathbf{y})\cdot k_2(\mathbf{x},\mathbf{y}) = h^\top(\mathbf{x})\cdot h(\mathbf{y})$ for some function $h(\cdot)$, thereby proving that $k_\times(\cdot,\cdot)$ is a valid kernel.

1.2 PCA

Based on Lecture 9 and Recitation 11

4. Let $X:\Omega\to\mathbb{R}^d$ be a random variable with zero mean and covariance $\Sigma\in\mathbb{R}^{d\times d}$. Show that for any $\mathbf{v}\in\mathbb{R}^d$, where $\|\mathbf{v}\|_2=1$, the variance of $\langle\mathbf{v},X\rangle$ is not larger then variance obtained by the PCA embedding of X into a one-dimension subspace (assume that the PCA uses the actual Σ).

1.3 Convex optimization

Based on Lecture 11 and Recitations 2,12

1. Let $f_1,\dots,f_m:C\to\mathbb{R}$ be a set of convex functions and $\gamma_1,\dots,\gamma_m\in\mathbb{R}_+$. Prove from definition that $g(\mathbf{u})=\sum_{i=1}^m\gamma_i f_i(\mathbf{u})$ is a convex function.
2. Give a counterexample for the following claim: Given two functions $f,g:\mathbb{R}\to\mathbb{R}$, define a new function $h:\mathbb{R}\to\mathbb{R}$ by $h=f\circ g$. If f and g are convex then h is convex as well.
3. Let $f:C\to\mathbb{R}$ be a function defined over a convex set C. Prove that f is convex iff its *epigraph* is a convex set, where epi(f) $=\{(u,t):f(u)\le t\}$.
4. Let $f_i:V\to\mathbb{R}$, $i\in I$. Let $f:V\to\mathbb{R}$ given by
$$f(u)=\sup_{i\in I}f_i(u).$$
If f_i are convex for every $i\in I$, then f is also convex.

1.4 Sub-gradients for Soft-SVM Objective

Based on Lecture 11 and Recitations 2,12

The Soft-SVM objective, though convex, is not differentiable in all of it's domain due to the use of the hinge-loss. Therefore, to implement a sub-gradient descent solver for this problem we must first describe sub-gradients of the objective.

5. Given $\mathbf{x}\in\mathbb{R}^d$ and $y\in\{\pm1\}$. Show that the hinge loss is convex in $\mathbf{w},b$. That is, define
$$f(\mathbf{w},b):=\ell^{hinge}_{\mathbf{x},y}(\mathbf{w},b)=\max\Big(0,1-y(\mathbf{x}^\top\mathbf{w}+b)\Big)$$
and show that f is convex in $\mathbf{w},b$.
6. Deduce some sub-gradient of the hinge loss function $g\in\partial\ell^{hinge}_{\mathbf{x},y}(\mathbf{w},b)$.
7. Let $f_1,\dots,f_m:\mathbb{R}^d\to\mathbb{R}$ be a set of convex functions and $\mathbf{g}_k\in\partial f_k(\mathbf{x})$ for all $k\in[m]$ be sub-gradients of these functions. Define $f:\mathbb{R}^d\to\mathbb{R}$ by $f(\mathbf{x})=\sum_{i=1}^m f_i(\mathbf{x})$. Show that $\sum_k\mathbf{g}_k\in\partial\sum_k f_k(\mathbf{x})$.
8. Let $S=\{(\mathbf{x}_i,y_i)\}_{i=1}^m\subseteq\mathbb{R}^d\times\{\pm1\}$ be a sample and define $f:\mathbb{R}^d\to\mathbb{R}$ by:
$$f(\mathbf{w},b)=\tfrac{1}{m}\sum_{i=1}^m\ell^{hinge}_{\mathbf{x}_i,y_i}(\mathbf{w},b)+\tfrac{\lambda}{2}\,||\mathbf{w}||^2$$
Find a sub-gradient of f for any $\mathbf{w}$.

2 Practical Part

Before starting the practical part please make sure to have cloned the IML.HUJI GitHub repository and setup the virtual environment as specified in the instructions. Write the necessary code in the files specified in the questions.

2.1 Gradient Descent

In the following section you will implement a generic Gradient Descent algorithm, and explore and visualize its performance on different objective functions. To assist you with the implementation please start by reading the documentation of the `GradientDescent` class in the `IMLearn.descent_methods.gradient_descent.py` file and following the steps as described below.

Learning Rate: The `GradientDescent` class, when initialized, receives a *learning rate strategy* in the form of a `BaseLR` instance. Read the documentation of the `BaseLR` base class in the `IMLearn.base.base_learning_rate.py` file and then implement two learning rate strategies in the :

- Constant (Fixed) Learning Rate (i.e. $\eta_t = \eta$) - `FixedLR` class in the `IMLearn.descent_methods.learning_rate.py` file
- (Optional) Exponentially decaying Learning Rate. $\eta_t = \eta \cdot \gamma^t$ - `ExponentialLR` class in the `IMLearn.descent_methods.learning_rate.py` file

Objective Functions (Modules): When running the `GradientDescent.fit` function it receives an instance derived from the `BaseModule` class. This class defines the generic abstract form of any objective to be minimized using graident descent. Its two main functions are used to compute the value of the function and the derivative of the function at a given point of interest. Read the documentation of the `BaseModule` base class in the `IMLearn.base.base_module.py` file.

Implement the `L2` and `L1` modules in the `IMLearn.descent_methods.modules.py` file. Note that both these modules ignore any passed inputs in the `compute_output` and `compute_jacobian` functions and simply use the `weights` defined in the base class. Some of the other modules will be implemented later.

Gradient Descent Algorithm Implement the `GradientDescent` class in the `descent_methods.gradient_descent.py` file.

- Note that when instantiating a `GradientDescent` object a `callback` can be passed. This will be used to investigate different properties of the algorithm's run and will be specified in the questions below.
- Implementation must support several solution types, one of which is the average of $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(t)}$. In your implementation do not store all solutions to avoid wasting memory.

2.1.1 Comparing Fixed learning rates

We begin with investigating the GD convergence over the L1 and L2 objectives using fixed learning rates. In the `gradient_descent_investigation.py` file implement the function `compare_fixed_learning_rates` as specified in function documentation:

- Implement the `get_gd_state_recorder_callback` function as specified in function documentation. This function returns a "fresh" callback function and lists for losses and weights throughout the GD iterations.
- Minimize the `L1` and `L2` modules for each of the following fixed learning rates $\eta \in \{1, 0.1, 0.01, 0.001\}$, setting the initial starting point (i.e. the initial value of the module's weights) to $\mathbf{w}_0 = \left(\sqrt{2}, \frac{e}{3}\right)$
  - Notice that the The `L2` module actually implements the **squared** L2 norm.

Of note, all the objective functions we saw so far depended on the training data $\mathbf{X}, \mathbf{y}$. In this section, we minimize a function that ignores the given training data (like all regularization modules). Modules implemented later in this exercise will use the training data.

Then, answer the following questions:

1. Plot the descent path for each of the settings described above (you can use the `plot_descent_path`). Add below the plots for $\eta = 0.01$ and explain the differences seen between the `L1` and `L2` modules.
2. Describe two phenomena that can be seen in the descent path of the ℓ_1 objective when using GD and a fixed learning rate.
3. For each of the modules, plot the convergence rate (i.e. the norm as a function of the GD iteration) for all specified learning rates. Explain your results
4. What is the lowest loss achieved when minimizing each of the modules? Explain the differences

2.1.2 Comparing Exponentially Decaying learning rates (Optional)

Next, we will use the exponential decay (instead of the fixed learning rate) to optimize the `L1` module. Starting from $\mathbf{w}_0 = \left(\sqrt{2}, \frac{e}{3}\right)$ use the exponential decay with $\eta = 0.1$ and $\gamma \in \{0.9, 0.95, 0.99, 1\}$. Then, answer the following questions.

5. (Optional) Plot the convergence rate for all decay rates in a single plot. Explain your results.
6. (Optional) How does the algorithm perform using the exponential decay compared to the fixed learning rate? What is the lowest ℓ_1 norm achieved using the exponential decay. Explain why there are differences.
7. (Optional) Plot the descent path for the $\gamma = 0.95$. Describe how the descent path changed from when using a fixed learning rate.

2.2 Minimizing Regularized Logistic Regression

In the following part you will use your implementation of Gradient Descent to solve a regularized logistic regression optimization problem. Implement the following as described below:

- Implement the `LogisticModule` in the `IMLearn.desent_methods.modules.py` file, as described in class documentation.
  - In the `compute_output`, you should return the negative log-likelihood

$$f(\mathbf{w}) = -\frac{1}{m}\log\left(\prod_i P(Y = y_i \mid X = \mathbf{x}_i, \mathbf{w})\right)$$

    recall derivations seen in class and recitations, as well as in the course book.
  - In the `compute_jacobian`, you should return the derivative of the objective above $\frac{\partial f(\mathbf{w})}{\partial \mathbf{w}}$.
  - Recall the calculations done in recitations 5 and 11.
- Implement the `RegularizedModule` in the `IMLearn.descent_methods.modules.py` file, as described in class documentation. This module receives two generic modules to be used as the fidelity and regularization terms. For example: `RegularizedModule(LogisticModule(), L1())`.
- Implement the `LogisticRegression` class in the `IMLearn.learners.classifiers.logistic_regression.py` file as specified in class documentation. This class should wrap the usage of your gradient descent implementation on the `LogisticModule`, `RegularizedModule`, `L1` and `L2`.
- When initializing the model's weights sample from $\mathbf{w} \sim \mathcal{N}\left(0, \frac{1}{d}I\right)$. This is equivalent to sampling from the standard Gaussian distribution and dividing by $\sqrt{d}$.

R Notice that the `LogisticRegression` class does not create an instance of the `GradientDescent` class. Instead, it receives it as a *dependency* in the constructor. As such, your `LogisticRegression` implementation is open for future extensions and support of different solvers. This is one of the 5 SOLID coding principles - if you wish to write good code and have a good design - be SOLID.

Then, load the South Africa Heart Disease dataset (SAheart.data), split it to train- and test sets (80% train) and answer the following questions:

8. Using your implementation, fit a logistic regression model over the data. Use the `predict_proba` to plot an ROC curve. You can use sklearn's `metrics.roc_curve` function and the code provided in Lab 04.
9. Which value of α achieves the optimal ROC value according to the criterion below. Using this value of α^* what is the model's test error?

$$\alpha^* = \text{argmax}_\alpha\{\text{TPR}_\alpha - \text{FPR}_\alpha\}$$

10. Fit an ℓ_1-regularized logistic regression by passing `penalty="l1"` when instantiating a logistic regression estimator
    - Set $\alpha = 0.5$
    - Use your previously implemented cross-validation procedure to choose λ
    - After selecting λ repeat fitting with the chosen λ and $\alpha = 0.5$ over the entire train portion.

What value of λ was selected and what is the model's test error?

When fitting the model you can (but don't have to) set the parameters as follows:

- Use `max_iter=20,000` and `lr=1e-4`.
- When searching for the optimal λ:
  - Search the following values $\lambda \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1\}$.
  - Use $\alpha = 0.5$ as the cutoff.

11. Repeat question 10 for ℓ_2 regularized logistic regression. Values used for `max_iter`, `lr`, λ and α can be set as they were set in the previous question. What value of λ was selected and what is the model's test error?

Based on Lecture 6 and Recitation 10

1. Let $k(\mathbf{x},\mathbf{x}')$ be a valid PSD kernel. Provide a valid PSD kernel $\tilde{k}(\mathbf{x},\mathbf{x}')$, constructed from k, which is guaranteed to be normalized. That is, for all $\mathbf{x}$ it holds that $\tilde{k}(\mathbf{x},\mathbf{x}) = 1$. Prove your answer.

מאחר מכפר K משרה מכפלה פנימית

לגבי

$$\tilde{K}(x,x') = \frac{K(x,x')}{\sqrt{K(x,x)}\cdot\sqrt{K(x',x')}}$$

ולכן ל

$$\tilde{K}(x,x) = \frac{K(x,x)}{K(x,x)} = 1$$

(כיון ל K אם $K(x,x)\neq 0$.)

$\tilde{K}$ מנורמלת.

PSD K : PSD ל $\tilde{K}$ נראה

לכן $K(x,x') = K(x',x)$ *

$$\tilde{K}(x,x') = \frac{K(x,x')}{\sqrt{K(x,x)K(x',x')}} = \frac{K(x',x)}{\sqrt{K(x',x')K(x,x)}} =$$

$$\tilde{K}(x',x)$$

לכן סימטרי. נשאר להראות שכל ע"ע של $\tilde{K}$ חיובי. יהי x וקטור עצמי של K ו"ע $1-x$. אזי:

$$\lambda\|x\|^2 = \tilde{K}(x,x) = \frac{K(x,x)}{K(x,x)} = 1 \Rightarrow$$

לכל $x \neq 0$ $\quad \lambda = \frac{1}{\|x\|^2} > 0$

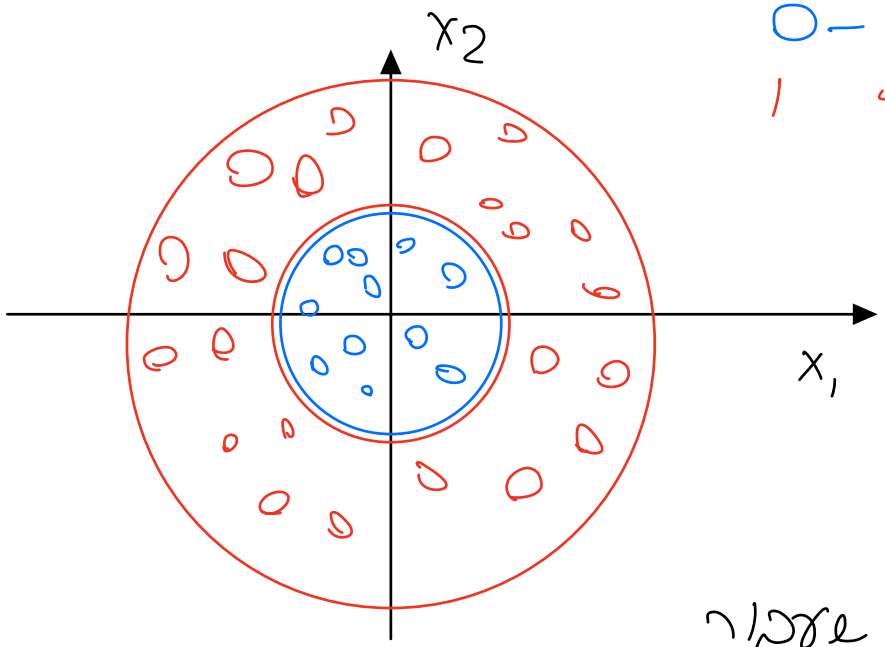
כנדרש

2. Consider a data set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{\pm 1\}$, and a feature map $\psi : \mathbb{R}^d \to \mathcal{F}$ where $\mathcal{F}$ is some feature space. Give an example of a data set S and a feature map ψ such that S is not linearly separable in $\mathbb{R}^d$ (for $d \geq 2$) but that the transformed data set $S_\psi = \{(\psi(\mathbf{x}_i), y_i)\}_{i=1}^m$ is linearly separable in $\mathcal{F}$.

לדוגמה

-1

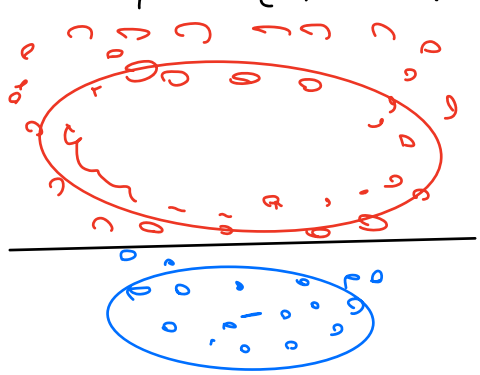
1



קחו שעבור
הדוגמאים הנתון (x_1, x_2) הבאנו לו
נתון לנפרדה.

נתבונן בפונקציה עבורה

$$\psi(x_1, x_2) = (x_1, x_2, x_1^2 + x_2^2) \longrightarrow$$



לכן מרחב
שנתון לינארית.

3.

3. $k_1(\mathbf{x},\mathbf{y})$ and $k_2(\mathbf{x},\mathbf{y})$ are valid kernels, then:

$$k_\times(\mathbf{x},\mathbf{y}) = k_1(\mathbf{x},\mathbf{y}) \cdot k_2(\mathbf{x},\mathbf{y})$$

is also a valid kernel. To prove this we'll use the fact that valid kernels are positive semi-definite.
You may find the following identities helpful (but don't have to use them):

$$\mathbf{x}^\top A\mathbf{y} = Tr\left[\mathbf{x}^\top A\mathbf{y}\right] = Tr\left[\mathbf{y}\mathbf{x}^\top A\right] \tag{1}$$

$$Tr[AB] = \sum_i [AB]_{ii} = \sum_i \sum_j A_{ij}B_{ji} \tag{2}$$

where $\mathbf{x}$ and $\mathbf{y}$ are vectors while A and B are matrices.

(a) Let $k(\mathbf{x},\mathbf{y})$ be a valid kernel and suppose that K is the kernel's Gram matrix over some finite set of points $\{\mathbf{x}_i\}_{i=1}^N$, such that $K_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$. Show that for any finite set of N points, there exists some function $f : \mathcal{X} \mapsto \mathbb{R}^N$ such that:

$$k(\mathbf{x}_i,\mathbf{x}_j) = f^\top(\mathbf{x}_i)f(\mathbf{x}_j) \tag{3}$$

where $\mathcal{X}$ is space of the points $\mathbf{x}_i$. Using this fact, show that:

$$k_1(\mathbf{x},\mathbf{y}) \cdot k_2(\mathbf{x},\mathbf{y}) = \sum_i \sum_j g_i(\mathbf{x})f_j(\mathbf{x})f_j(\mathbf{y})g_i(\mathbf{y}) \tag{4}$$

---

where $k_1(\cdot,\cdot)$ and $k_2(\cdot,\cdot)$ are valid kernels, and some functions $f,g : \mathcal{X} \mapsto \mathbb{R}^N$, where $f_i(\mathbf{x})$ denotes the ith index of the output of $f(\mathbf{x})$.

(b) Conclude that $k_\times(\mathbf{x},\mathbf{y}) = k_1(\mathbf{x},\mathbf{y}) \cdot k_2(\mathbf{x},\mathbf{y}) = h^\top(\mathbf{x}) \cdot h(\mathbf{y})$ for some function $h(\cdot)$, thereby proving that $k_\times(\cdot,\cdot)$ is a valid kernel.

PCA

(ב) מפני שמדובר בקרנל של כפל קרנלים

$$K(x_i, x_j) = \langle f(x_i)\, f(x_j) \rangle = f^T(x_i)\, f(x_j)$$

.2

$$K_1(x,y)\cdot K_2(x,y) = \qquad \forall\, i,j \in [N]$$

$$f^T(x) f(y)\, g^T(x) g(y) = \sum_{j=0}^{N} f(x_j) f(y_j) \sum_{i=0}^{N} g(x_i) g(y_i) =$$

$$\sum_{i=0}^{N}\sum_{j=0}^{N} f^T(x_j) f^T(y_j)\, g(x_i) g(y_i) =$$

$$\sum_{i=0}^{N}\sum_{j=0}^{N} g(x_i)^T f(x_j)^T f(y_j)\, g(y_i)$$

$$\Downarrow$$

לכן

$$K(x,y) = K_1(x,y)\, K_2(x,y) =$$

$$\sum_{i}^{N}\sum_{j}^{N} g_i^T(x_j)\, f^T(x_i)\, f(y_i)\, g(y_j) =$$

$$h^T(x)\cdot h(y) \qquad \blacksquare$$

1.2 PCA

Based on Lecture 9 and Recitation 11

4. Let $X : \Omega \to \mathbb{R}^d$ be a random variable with zero mean and covariance $\Sigma \in \mathbb{R}^{d \times d}$. Show that for any $\mathbf{v} \in \mathbb{R}^d$, where $\|\mathbf{v}\|_2 = 1$, the variance of $\langle \mathbf{v}, X \rangle$ is not larger then variance obtained by the PCA embedding of X into a one-dimension subspace (assume that the PCA uses the actual Σ).

$$E\langle v, X\rangle = 0 \Rightarrow$$

$$E\left(\left(\langle v, X\rangle - E(v, X)\right)^2\right) =$$

$$E\left(\langle v, X\rangle^2 - E(\langle v, X\rangle)\cdot\langle v, X\rangle + E\langle v, X\rangle^2\right)$$

$$= E\left(\langle v, X\rangle^2\right) = E\left(v^T X \cdot X v\right) =$$

$$v^T E\left[X \cdot X^T\right] v = v^T \Sigma v \leq$$

$$v_1^T \Sigma v_1$$

כאשר v_1 הוא וקטור העצמי ה גדול ביותר של Σ

Based on Lecture 11 and Recitations 2,12

1. Let $f_1, \ldots, f_m : C \to \mathbb{R}$ be a set of convex functions and $\gamma_1, \ldots, \gamma_m \in \mathbb{R}_+$. Prove from definition that $g(\mathbf{u}) = \sum_{i=1}^{m} \gamma_i f_i(\mathbf{u})$ is a convex function.

ראינו בתרגול שסכום כל סקלר חיובי
פי פונקציה קמורה מתקיים
שסכום הפונקציה פי פי קמורה. לכן
ראינו שסכום של פונקציות קמורות הוא
קמור ולכן הטענה מתקיימת.

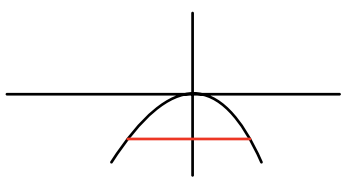
2. Give a counterexample for the following claim: Given two functions $f, g : \mathbb{R} \to \mathbb{R}$, define a new function $h : \mathbb{R} \to \mathbb{R}$ by $h = f \circ g$. If f and g are convex then h is convex as well.

עבור $g(x) = -x$ $\quad f(x) = x^2$

כל פונקציה לינארית היא קמורה

$\Rightarrow f \circ g(x) = -x^2$

ובפרט שלא אינה פונקציה קמורה



3. Let $f : C \to \mathbb{R}$ be a function defined over a convex set C. Prove that f is convex iff its *epigraph* is a convex set, where $\mathrm{epi(f)} = \{(u,t) : f(u) \le t\}$.

$\Rightarrow$

f קמורה.

נניח בשלילה ש- $epi(f)$ אינה קמורה $\Leftarrow$

$\exists \lambda \in [0,1]$! $x_1, x_2 \in epi(f)$ כך ש

$x_1 = (u_1, t_1)$ $x_2 = (u_2, t_2)$ המקיים

$\lambda x_1 + (1-\lambda) x_2 \notin epi(f)$

כלומר אומר ש

$$\circledast = f(\lambda u_1 + (1-\lambda) u_2) > \lambda t_1 + (1-\lambda) t_2$$

מהקמירות של f מתקיים

$$\lambda t_1 + (1-\lambda) t_2 \ge \lambda f(u_1) + (1-\lambda)(u_2) \ge \circledast$$

סתירה

$\Leftarrow$ נניח ש $epi(f)$ קמורה $\Leftarrow$

לכל $x_1, x_2 \in epi(f)$ (מלבנים כמו קודם)

$$\lambda x_1 + (1-\lambda) x_2 \in epi(f) \Rightarrow$$

$$f(\lambda u_1 + (1-\lambda) u_2) \le \lambda t_1 + (1-\lambda) t_2$$

נשים לב ש $(u_1, f(u_1)) \in epi(f)$
גם $(u_2, f(u_2)) \in epi(f)$

כלומר המספרים שנמצאים מעל הגרף
$t_1 = f(u_1)$ $t_2 = f(u_2)$

$$f(\lambda u_1 + (1-\lambda) u_2) \le \lambda f(u_1) + (1-\lambda) f(u_2)$$

! - f קמורה.

4. Let $f_i : V \to \mathbb{R}$, $i \in I$. Let $f : V \to \mathbb{R}$ given by

$$f(u) = \sup_{i \in I} f_i(u).$$

If f_i are convex for every $i \in I$, then f is also convex.

$$f(u) = \max(f_1(u), \dots f_n(u)) \Rightarrow$$

$$f(\lambda u + (1-\lambda) v) = f_i(\lambda u + (1-\lambda) v)$$

עבור $i \in I$ כלשהו.

$$\le \lambda f_i(u) + (1-\lambda) f_i(v) \le \lambda f(u) + (1-\lambda) f(v)$$

מקסימום.

מהגדרת f
כמקסימום.

1.4 Sub-gradients for Soft-SVM Objective

Based on Lecture 11 and Recitations 2,12

The Soft-SVM objective, though convex, is not differentiable in all of it's domain due to the use of the hinge-loss. Therefore, to implement a sub-gradient descent solver for this problem we must first describe sub-gradients of the objective.

5. Given $\mathbf{x} \in \mathbb{R}^d$ and $y \in \{\pm 1\}$. Show that the hinge loss is convex in $\mathbf{w}, b$. That is, define

$$f(\mathbf{w}, b) := \ell^{hinge}_{\mathbf{x},y}(\mathbf{w}, b) = \max\left(0, 1 - y(\mathbf{x}^\top \mathbf{w} + b)\right)$$

   and show that f is convex in $\mathbf{w}, b$.

6. Deduce some sub-gradient of the hinge loss function $g \in \partial \ell^{hinge}_{\mathbf{x},y}(\mathbf{w}, b)$.
7. Let $f_1, \ldots, f_m : \mathbb{R}^d \to \mathbb{R}$ be a set of convex functions and $\mathbf{g}_k \in \partial f_k(\mathbf{x})$ for all $k \in [m]$ be sub-gradients of these functions. Define $f : \mathbb{R}^d \to \mathbb{R}$ by $f(\mathbf{x}) = \sum_{i=1}^m f_i(\mathbf{x})$. Show that $\sum_k \mathbf{g}_k \in \partial \sum_k f_k(\mathbf{x})$.

8. Let $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \subseteq \mathbb{R}^d \times \{\pm 1\}$ be a sample and define $f : \mathbb{R}^d \to \mathbb{R}$ by:

$$f(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^m \ell^{hinge}_{\mathbf{x}_i,y_i}(\mathbf{w}, b) + \frac{\lambda}{2} ||\mathbf{w}||^2$$

   Find a sub-gradient of f for any $\mathbf{w}$.

5. כדי לראות פונקציה שמקסימום
של קמורות קמורים גם הוא.

$f_i(w,b) = 0$ קמורה

$f_i(w,b) = 1 - y(x^T w + b)$

$g(b) = b$ $\quad h(w) = x^T w$ נגדיר

ונקבל

$f_i(w,b) = 1 - y(h(w) + g(b))$

כאשר g, h קמורות כיוון שזן לינאריות
y קבוע ולכן הפונקציה היא סכום של קמורות

וקמורה. בנוסף, לכן גם f שהוא
מקסימום של פונקציות הפונקציות קמורה.

$$\frac{\partial(-y(w^T x+b))}{\partial w} = -yx$$

$$\frac{\partial(-y(w^T x+b))}{\partial b} = -y \Rightarrow$$

$$g = \begin{cases} 0 & \text{hingh}_{x,y}(w,b) = 0 \\ (-yx, -y) & l^{hinge}(w,b) = -y(w^T x+b) \end{cases}$$

$$g_k \in \partial f_k(x) \Rightarrow f_k(y) \geq f_i(x) + \langle g_k, y-x \rangle$$

$$\sum_k f_k(y) \geq \sum_k (f_k(x) + \langle g_k, y-x \rangle) =$$

$$\sum_k f_k(x) + \langle \sum g_k, y-x \rangle$$

$$\sum_k g_k \in \partial \sum_k f_k(x)$$

אם כיוון לשאלה 6 תאם לרב"ג

$f^{hinge}_{x,y}(w,b)$:

$$g = \begin{cases} 0 & hinge_{x,y}(w,b) = 0 \quad x \in \mathbb{R}^2 \\ (-yx, -y) & f^{hinge}_{x,y}(w,b) = -y(w^T x + b) \end{cases}$$

מתקיימת הסאב גרדיאנט שאינו בגלל

מתקיים

$$\partial g(\gamma v) = \gamma \partial g(v)$$

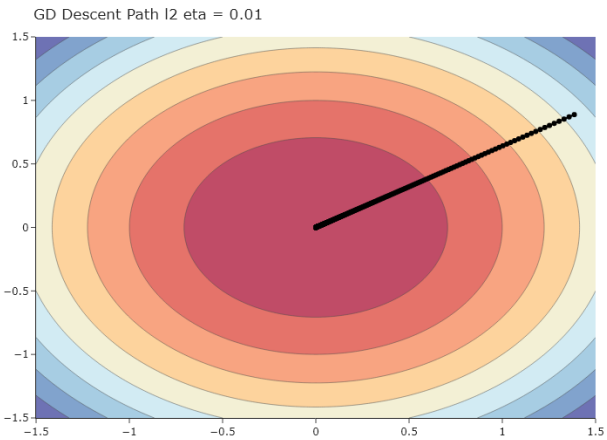
לכן נוכל לחבר את g ב- $\frac{1}{m}$:

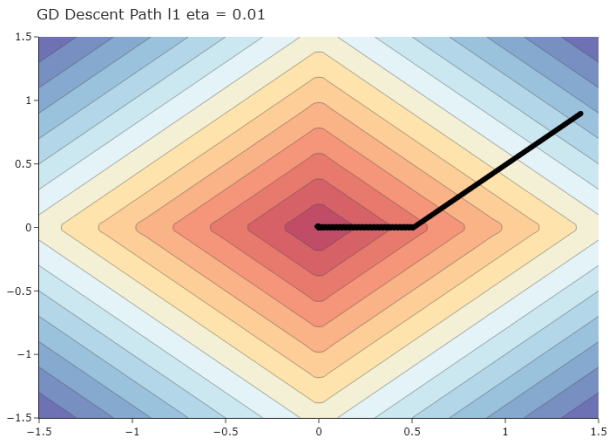
$$\partial \frac{\lambda}{2}\|w\|^2 = \lambda w$$

מסקנה ∂ :

$$\frac{1}{m}\sum_i g_i + \lambda w \in \partial \frac{1}{m}\sum_i f^{hinge}_{x_i,y_i}(w,b) + \frac{\lambda}{2}\|w\|^2$$

ל.





ב- l_1 הגודל של הגרדיאנט קבוע ולכן נקבל צעד באורך קבוע בכל איטרציה (עד שנגיע לנק' בה הגרדיאנט 0).

לעומת לא-כן, ב- l_2 הגרדיאנט גדל יחד עם שמתרחקים מנק' ה-0 ולכן גודל הצעד קטן ככל שמתקרבים ל-0.

ג.

<u>תוצאה א':</u>

גודל צעד קבוע, גם שמתקרב לסף הקודם

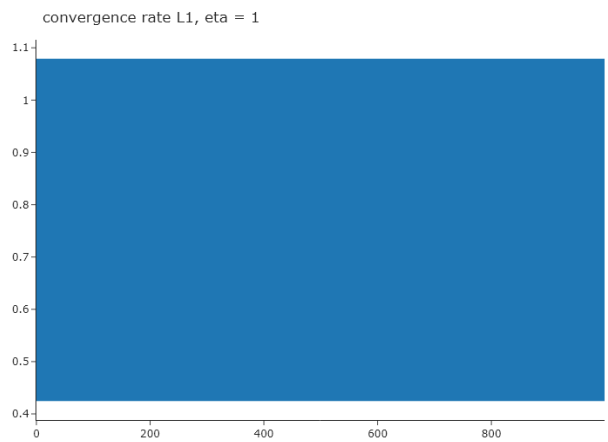
<u>תוצאה ב':</u>

כאן הגרדיאנט קבוע עד לנק בה אחד מהמ'ג'רים אינו

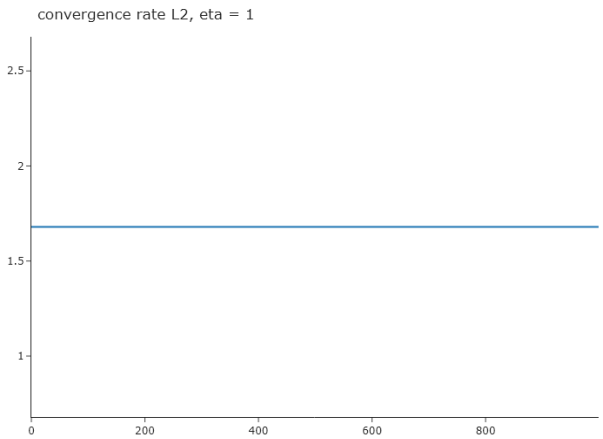
אפשר יותר, אולי בכיוון שאינה ההתכנסות
מתחשב בין 1-0 בהיבטים.

3

I



II



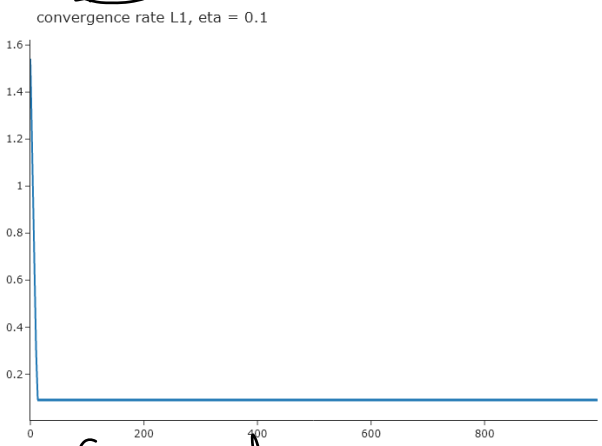
I

אנו מעריכים שהאופטימיזציה
שאנו רואים היא
overshooting - כלומר
איטרציה אחת מצליחה
מעל המינימום ולוקחת
הרבה זמן קבוע
כיון שהגרדיאנט קבוע ב- η.

II

ה loss נשאר קבוע -
אולי גם כיון הבקר נמצא
מאוד שולי סימטרי
למינימום קביעה?

III

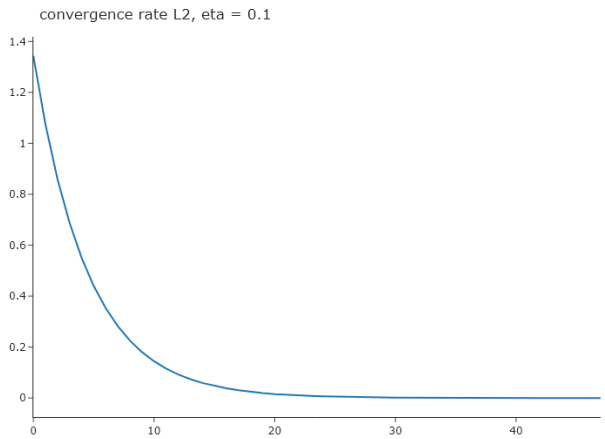


III

יש התכנסות ב' מהירה
אולם אין יותר שיפור
כי הגענו שוב לנקודה
רק יש אופק שוויון
סימטרי קבוע לאינסוף.

לעומת אלגוריתם אלתנו משלים את המשקולות אבל
בלוס גם ב הוא לוגריתמי בפרט:
ל מושג ביולוגי

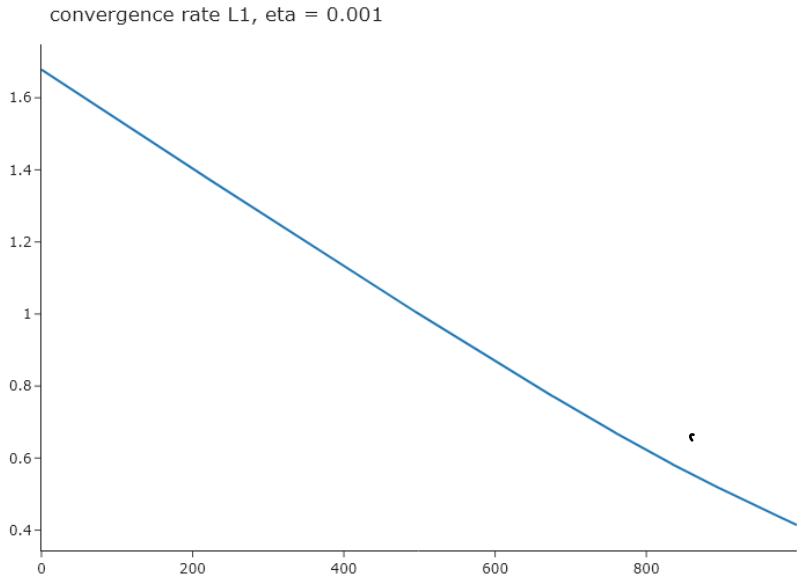
IV



IV

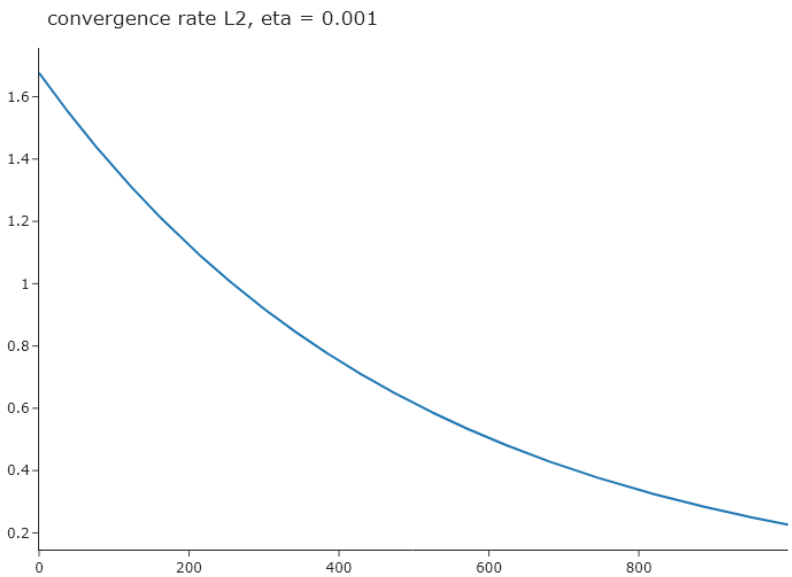
יש התכנסות אבל
בגלל רגישות אופק יותר
(בלוס קטן יותר)
כי בגרדיאנט קטן
בגלל שמרחקים / - מות.

V



כדאי קצת שב'
ונתבונן ל max
iteration

לפני שנתבונן ל-ε
ברגע זה.



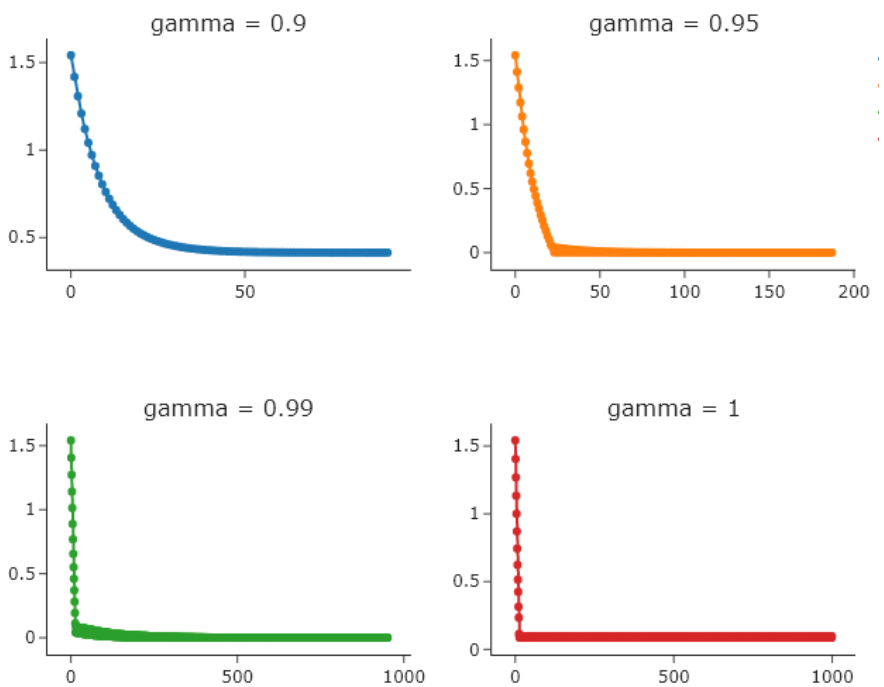
VI
אולי כדאי
כמו שנהוג
לנתיב בקצרה.

4

```
min loss l1 : 0.00840351114116088
min loss l2 : 3.7456000184675695e-05
```

כיון שכ- l_2 הישרות משענה
אבל הגדר כאן אותנו מקבלים קירוב
אם יותר.

5



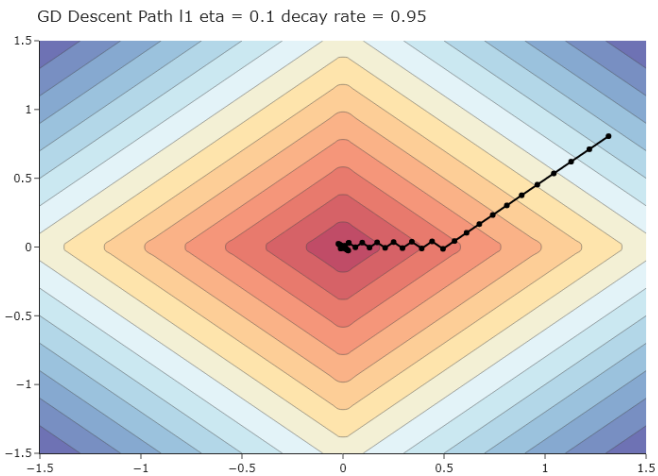
ניתן לראות שבאמצעות שימוש ב Decay
שמקטין את גודל הצעד מקלים במספר איטרציות
מתקבלות איכויות אחרות לכ ל- l

```
the min loss is : 3.216480204937497e-07
```

6

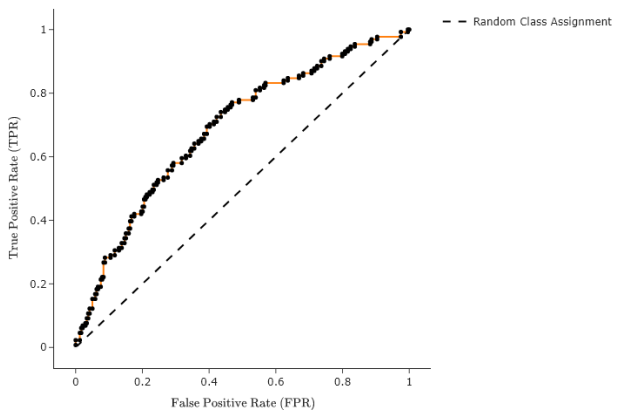
בגלל הפסיק היקובס, הצמצום ? decay rate
שפי איתנו אבל לא לקבל מבוקש יותר

7



ניתן לראות
שהגרדיאנט קיים
כל שמתקדמים
בכיוון הנגדי
מהשלה ו.

8



9, 10, 11 לצערי מיקוד לא עבור ובשל אבעיות פיתחנו
אין לי זמן לעבור כרגע

```
Q9 alpha star = 0.00032693601848671466, and the loss over the test is : 0.6847826086956522
l1 loss: 0.31521739130434784, lambda = 0.002, alpha = 0.5
l2 loss: 0.6847826086956522, lambda = 0.05, alpha = 0.5
```