Mathematical Notations for Learning from Data Course

Islem Rekik*

Istanbul Technical University

Abstract. Inspiring quotes. *"You'll need to set small, specific goals to master a skill, but first you'll want to be sure of the basics."* Source: *Learn better* by Ulrich Boser.
"Learning is an iterative process that requires that you revisit what you have learnt." Source: *Make it stick* by Henry L. Roediger III and Mark A. McDaniel.
"The good news is that we now know of simple and practical strategies that anybody can use; at any point in life, to learn better and remember longer: various forms of retrieval practice, such as low-stakes quizzing and self-testing, spacing out practice, interleaving the practice of different but related topics or skills, trying to solve a problem before being taught the solution, distilling the underlying principles or rules that differentiate types of problems, and so on." Source: *Make it stick* by Henry L. Roediger III and Mark A. McDaniel.
Pre-requisites: Linear algebra. To refresh your memory, you can check 3blue1brown YouTube playlist[a].
Machine Learning Blinks. Check the YouTube link below to watch the lectures[b].

* Corresponding author: irekik@itu.edu.tr; http://basira-lab.com
[a] https://www.youtube.com/watch?v=fNk_zzaMoSs&list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab
[b] https://www.youtube.com/watch?v=HyWmnlahXAA&list=PLug43ldmRSo1LDlvQOPzgoJ6wKnfmzimQ

Table 1: *Major mathematical notations used in lecture 1.*

Mathematical notation	Definition
$\mathcal{D}$	dataset
n	number of samples in a dataset $\mathcal{D}$
d	number of features
$\mathbf{x} \in \mathbb{R}^{d \times 1}$	feature vector or data point (sample)
$\mathbf{x}_{feature}^{sample}$	–
$\mathbf{x}^i \in \mathbb{R}^{d \times 1}$	i^{th} sample in the population
$\mathbf{x}_j^i \in \mathbb{R}$	j^{th} feature of i^{th} sample in the population
$\mathcal{D} = \{\mathbf{x}^i, y^i\}_{i=1}^n$	training dataset where $\mathbf{x}^i \in \mathbb{R}^d$ denotes the feature vector for the i^{th} sample and $y^i \in \mathbb{R}$ denotes its score
$\mathbf{X} \in \mathbb{R}^{d \times n}$	data matrix stacking all samples vertically
f	mapping or transformation function to learn
$f : \mathbb{R} \mapsto \mathbb{R}$	one-to-one mapping
$f : \mathbb{R} \mapsto \mathbb{R}^m$	one-to-many mapping
$f : \mathbb{R}^p \mapsto \mathbb{R}$	many-to-one mapping
$f : \mathbb{R}^p \mapsto \mathbb{R}^m$	many-to-many mapping

Table 2: *Major mathematical notations used in lecture 3.*

Mathematical notation	Definition
$\mathcal{D}$	dataset
n	number of samples in a dataset $\mathcal{D}$
d	number of features
$\mathbf{x} \in \mathbb{R}^{d \times 1}$	feature vector or data point (sample)
$\mathbf{x}_{feature}^{sample}$	–
$\mathbf{x}^i \in \mathbb{R}^{d \times 1}$	i^{th} sample in the population
$\mathbf{x}_j^i \in \mathbb{R}$	j^{th} feature of i^{th} sample in the population
$\mathbf{\Sigma} \in \mathbb{R}^{d \times d}$	covariance matrix of data population $\{\mathbf{x}^i\}_{i=1}^n$
$\lvert\mathbf{A}\rvert \in \mathbb{R}$	determinant of matrix A
$p(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(\frac{-1}{2}\frac{\lvert\lvert x-\mu\rvert\rvert_2^2}{\sigma^2})$	probability density function of a variable $x \in \mathbb{R}$
$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\lvert\mathbf{\Sigma}\rvert^{1/2}} exp[\frac{-1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)]$	probability density function of a multidimensional variable $\mathbf{x} \in \mathbb{R}^{d \times 1}$
$\mu \in \mathbb{R}^{d \times 1}$	sample mean $\mu = \frac{1}{n}\sum_{i=1}^n \mathbf{x}^i$
$\lvert\lvert\mathbf{x}-\mu\rvert\rvert_{\mathbf{\Sigma}^{-1}} = (\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu) \in \mathbb{R}$	Mahalanobis distance between $\mathbf{x}$ and μ
$\mathbf{I}_{d \times d} \in \mathbb{R}^{d \times d}$	identify matrix of size d ×d
$\lvert\lvert\mathbf{x}-\mu\rvert\rvert_{\mathbf{I}_{d \times d}} = (\mathbf{x}-\mu)^T(\mathbf{x}-\mu) \in \mathbb{R}$	Euclidean distance between $\mathbf{x}$ and μ also noted as L_2 norm $\lvert\lvert \cdot \rvert\rvert_2$
$g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + \mathbf{w}_{i0}$	discriminant Bayes function for class i when $\mathbf{\Sigma}_i = \sigma_i^2\mathbf{I}$ general case: when $\sigma_i^2 \neq \sigma_j^2$ for classes i and j (i.e., different means $\mu_i \neq \mu_j$ but constant variance for all data features in each class) special case: when $\sigma_i^2 = \sigma_j^2$ for classes i and j (i.e., different means $\mu_i \neq \mu_j$ but constant variances across all classes) (i.e., lines connecting means of different classes are perpendicular to decision boundaries) if $ln(p(c_i)) = ln(p(c_j))$, $g_i(\mathbf{x}) = -\lvert\lvert\mathbf{x}-\mu_i\rvert\rvert_2^2$
$g_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + \mathbf{w}_{i0}$	discriminant Bayes function for class i when $\mathbf{\Sigma}_i = \mathbf{\Sigma}_j = \mathbf{\Sigma}$ (i.e., constant data feature covariance $\mathbf{\Sigma}$ across classes) (i.e., lines connecting means of different classes are not perpendicular to decision boundaries) if $ln(p(c_i)) = ln(p(c_j))$, $g_i(\mathbf{x}) = -\frac{1}{2}\lvert\lvert\mathbf{x}-\mu_i\rvert\rvert_{\mathbf{\Sigma}^{-1}}^2$
$g_i(\mathbf{x}) = \mathbf{x}^T\mathbf{W}\mathbf{x} + \mathbf{w}_i^T\mathbf{x} + \mathbf{w}_{i0}$	quadratic discriminant function (decision boundaries are nonlinear)

Table 3: *Major mathematical notations used in lectures 4.*

Mathematical notation	Definition
$arg\ min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \frac{1}{n} \sum_{(\mathbf{x}^i, y^i) \in \mathcal{D}} E(f_{\mathbf{w}}(\mathbf{x}^i), y^i) + R(\dots)$	supervised learning energy cost (loss function)
f	the mapping function to learn from $\mathbf{x}^i \rightarrow y^i$
E	the error function between the predicted target by f and the ground truth observation y^i
R	regularization term to avoid overfitting and control model complexity
$\mathbf{w}$	optimization parameters (weight vector)
	set of parameters that minimize the loss function $\mathcal{L}(\mathbf{w})$
$h(w) = l(v) + l'(v)(w - v)$	first-order Taylor approximation of the loss function l at point w (in 1-dimensional space)
$l'(w)$	first derivative of function l evaluated at point w
$h(w) = l(v) + l'(v)(w - v) + \frac{1}{2} l''(v)(w - v)^2$	second-order Taylor approximation of the loss function l at point w (in 1-dimensional space)
$l''(w)$	second derivative of function l evaluated at point w
$h(\mathbf{w}) = l(\mathbf{v}) + \nabla l(\mathbf{w})(\mathbf{w} - \mathbf{v})$	first-order Taylor approximation of high-dimensional loss function l at vector point $\mathbf{w} \in \mathbb{R}^d$
$\mathbf{w} = [w_1\ w_2 \dots w_N] \in \mathbb{R}^d$	weight vector to learn
$\nabla l(\mathbf{w}) \in \mathbb{R}^{d \times 1}$	gradient vector of the multivariate loss function l at location $\mathbf{w}$
	note that $\nabla l(\mathbf{w})^T$ is $\in \mathbb{R}^{1 \times d}$ (row vector)
	$\nabla l(\mathbf{v}) = [\frac{\partial}{\partial w_1} l(\mathbf{v})\ \frac{\partial}{\partial w_2} l(\mathbf{v}) \dots \frac{\partial}{\partial w_d} l(\mathbf{v})]^T$
$h(\mathbf{w}) = l(\mathbf{v}) + \nabla l(\mathbf{w})(\mathbf{w} - \mathbf{v}) + \frac{1}{2}(\mathbf{w} - \mathbf{v})^T \nabla^2 l(\mathbf{w})(\mathbf{w} - \mathbf{v})$	second-order Taylor approximation of high-dimensional loss function l at vector point $\mathbf{w} \in \mathbb{R}^d$
$\nabla^2 l(\mathbf{w}) \in \mathbb{R}^{d \times d}$	Hessian symmetric matrix of second derivatives of l along all its dimensions (variables)
	l is many times differentiable at the vector valued input $\mathbf{w}$
$l'(w) = 0$	stationary point (min, max or saddle) for a 1-dimensional function
$\nabla l(\mathbf{w}) = \mathbf{0}_{d \times 1}$	stationary point (min, max or saddle) for an N-dimensional function
	(all elements of the gradient are zero)
$\mathbf{Q} = \mathbf{Q}^T$	$\mathbf{Q}$ is symmetric
$\frac{\partial \mathbf{a}^T \mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}$	matrix cookbook[a] (also check[b])
$\frac{\partial \mathbf{a}^T \mathbf{X} \mathbf{a}}{\partial \mathbf{a}} = 2\mathbf{X}\mathbf{a}$ (for $X = X^T$)	matrix cookbook
$l''(w) > 0$	convex function (facing upward)
$l''(w) < 0$	concave function (facing downward)
$\frac{\partial \mathbf{A} \mathbf{x}}{\partial \mathbf{x}} = \frac{1}{2}(\mathbf{A}^T + \mathbf{A})$	matrix cookbook
$\nabla^2 l(\mathbf{w}) = \frac{1}{2}(\mathbf{Q}^T + \mathbf{Q}) \in \mathbb{R}^{d \times d}$	is the Hessian matrix of l equal to $l(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T \mathbf{Q} \mathbf{w} + \mathbf{r}^T \mathbf{w} + d$
	$d \in \mathbb{R}$ and $\mathbf{r} \in \mathbb{R}^{d \times 1}$
$\nabla l(\mathbf{w}) = \mathbf{Q}\mathbf{w} + \mathbf{r} \in \mathbb{R}^{d \times 1}$	is the gradient vector of l equal to $l(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T \mathbf{Q} \mathbf{w} + \mathbf{r}^T \mathbf{w} + d$
$\mathbf{w}^k = \mathbf{w}^{k-1} - \alpha_k \nabla l(\mathbf{w}^{\mathbf{k-1}})$	gradient descent for finding the optimal $\mathbf{w}$

[a] https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

[b] https://ninova.itu.edu.tr/en/courses/institute-of-science-and-technology/1580/blg-527e/ekkaynaklar?g179937

Table 4: *Major mathematical notations used in lecture 5.*

Mathematical notation	Definition
$\mathbf{w}^k = \mathbf{w}^{k-1} - \alpha_k \nabla l(\mathbf{w}^{\mathbf{k-1}})$	gradient descent for finding the optimal $\mathbf{w}^*$
$\nabla^2 l(\mathbf{w}^{k-1})\mathbf{w}^k = \nabla^2 l(\mathbf{w}^{k-1})\mathbf{w}^{k-1} - \nabla l(\mathbf{w}^{k-1})$	Newton's method update rule (second order optimization)
$\mathbf{w}^k = \mathbf{w}^{k-1} - [\nabla^2 l(\mathbf{w}^{k-1})]^{-1} \nabla l(\mathbf{w}^{k-1})$	Newton's method update rule when the Hessian matrix $\nabla^2 l(\mathbf{w}^{k-1})$ is invertible

Table 5: *Major mathematical notations used in lecture 6 (linear regression and logistic regression).*

Mathematical notation	Definition
$\{\mathbf{x}^i, y^i\}_{i=1}^n$	training dataset where $\mathbf{x}^i \in \mathbb{R}^d$ denotes the feature vector for the i^{th} sample and $y^i \in \mathbb{R}$ denotes its score
$\mathbf{x}^T\mathbf{z}$	inner product $<\mathbf{x}, \mathbf{z}>$ (or dot product) of vectors $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{z} \in \mathbb{R}^d$ it outputs a scalar $\in \mathbb{R}$
$\mathbf{x}\mathbf{z}^T$	outer product $\mathbf{x} \otimes \mathbf{z}$ of vectors $\mathbf{x} \in \mathbb{R}^d$ and $\mathbf{z} \in \mathbb{R}^{d'}$ it outputs a matrix of size $d \times d'$
$l(\mathbf{w}, b) = \sum_{i=1}^n (\mathbf{x}^{iT}\mathbf{w} + b - y^i)^2$	least squares loss function for linear regression where $\mathbf{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$
$l(\mathbf{w}, b) = \sum_{i=1}^n (\tilde{\mathbf{x}}^{iT}\tilde{\mathbf{w}} - y^i)^2$	least squares loss function for linear regression, where $\tilde{\mathbf{x}}^\mathbf{i} = [1\ \mathbf{x}^i]^T$, $\tilde{\mathbf{w}} = [b\ \mathbf{w}]^T$, $\tilde{\mathbf{x}}^\mathbf{i} \in \mathbb{R}^{d+1}$, and $\tilde{\mathbf{w}}^\star \in \mathbb{R}^{d+1}$
$\nabla l(\tilde{\mathbf{w}}) = \frac{\partial l(\tilde{\mathbf{w}})}{\partial \tilde{\mathbf{w}}} = 0$	setting the gradient to zero to find the optimal solution
$\tilde{\mathbf{w}}^\star = [\sum_{i=1}^n \tilde{\mathbf{x}}^i \tilde{\mathbf{x}}^{iT}]^{-1} \sum_{i=1}^n \tilde{\mathbf{x}}^i y^i$	closed form optimum of the least squares regression loss function
$\tilde{\mathbf{w}}^\star = [\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T]^{-1}\tilde{\mathbf{X}}\mathbf{y}$	compact closed form where $\tilde{\mathbf{X}}$ vertically stacks all training samples and $\mathbf{y} = [y^1, y^2, \ldots, y^n]^T$
$y^i \in \mathbb{R}$	is a scalar denoting the target value (or score) for a single sample i
$\mathbf{y} = [y^1, y^2, \ldots, y^n]^T \in \mathbb{R}^n$	is a column vector storing all target values for the n training samples
$\sigma(s) = \frac{1}{1+e^{-s}}$	typical sigmoid function without any scaling and shifting parameters σ maps an input value s to a point $\sigma(s)$ lying on the sigmoid curve
$\sigma(\mathbf{x}^{iT}\mathbf{w} + b) \sim y^i$	if the distribution of training samples $\{\mathbf{x}^i\}_{i=1}^n$ slightly vary around the a sigmoid curve parameterized by a scaling vector $\mathbf{w}$ and a shifting/translational scalar b then identifying the best $(\mathbf{w}, b)$ will identify the best sigmoid curve that fits the data
$l(\mathbf{w}, b) = \sum_{i=1}^n (\sigma(\mathbf{x}^{iT}\mathbf{w} + b) - y^i)^2$	denotes the logistic loss function to minimize
$l(\tilde{\mathbf{w}}) = \sum_{i=1}^n (\sigma(\tilde{\mathbf{x}}^i\tilde{\mathbf{w}}) - y^i)^2$	denotes the compact logistic loss function to minimize
$\frac{\partial \sigma(\tilde{\mathbf{x}}^T\tilde{\mathbf{w}})}{\partial \tilde{\mathbf{w}}} = \sigma(\tilde{\mathbf{x}}^T\tilde{\mathbf{w}}) \times (1 - \sigma(\tilde{\mathbf{x}}^T\tilde{\mathbf{w}})) \times \tilde{\mathbf{x}}^T$	partial derivative of a sigmoid dot product between two vectors $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{w}}$
$\frac{\partial l(\tilde{\mathbf{w}})}{\partial \tilde{\mathbf{w}}} = 0$	we first try to find a closed form of the optimal parameters by setting the gradient of the loss function to 0 to find the optimal parameters $\tilde{\mathbf{w}}^\star$
$\frac{\partial l(\tilde{\mathbf{w}})}{\partial \tilde{\mathbf{w}}} = 2\sum_{i=1}^n (\sigma(\tilde{\mathbf{x}}^{iT}\tilde{\mathbf{w}}) - y^i)\sigma(\tilde{\mathbf{x}}^{iT}\tilde{\mathbf{w}})(1 - \sigma(\tilde{\mathbf{x}}^{iT}\tilde{\mathbf{w}}))\tilde{\mathbf{x}}^i$	we need to use gradient descent to optimize l since *explicitly* computing $\tilde{\mathbf{w}}^\star$ is not feasible in this case
$l_{reg}(\mathbf{w}, b) = l(\mathbf{w}, b) + \lambda \|\mathbf{w}\|_2^2$	since gradient descent using the original logistic loss function is sensitive to the initial parameters $(\mathbf{w}^0, b^0)$ we introduce a regularization term (squared l_2 norm of the weights to learn if λ is very large, the regularizer drowns out the new loss function $l_{reg}(\mathbf{w}, b)$ λ should be fairly small ~ 0.1 to convexify or smooth the loss function in problematic regions
$\|\mathbf{w}\|_2^2 = \sum_{i=1}^d w_i^2$	the l_2 of a vector is also its Euclidean norm the squared l_2 of a vector is its squared Euclidean norm this is a convex function in $\mathbf{w}$ adding a convex function to a non-convex function convexifies the non-convex function to some extent

Table 6: *Major mathematical notations used in lecture 7 (perceptron for classification).*

Mathematical notation	Definition
$\{\mathbf{x}^i, y^i\}_{i=1}^n$	training dataset where $\mathbf{x}^i \in \mathbb{R}^d$ denotes the feature vector for the i^{th} sample and $y^i \in \{-1, 1\}$ denotes its class label
$-y^i(\mathbf{x}^{iT}\mathbf{w} + b) < 0$	criterion for correctly classifying sample $\mathbf{x}^i$
$-y^i(\mathbf{x}^{iT}\mathbf{w} + b) = 0$	criterion for incorrectly classifying sample $\mathbf{x}^i$
$l_{max}(\mathbf{w}, b) = \sum_{i=1}^n max(0, -y^i(\mathbf{x}^{iT}\mathbf{w} + b))$	max loss function or perception cost for classifying samples $\{\mathbf{x}^i\}_{i=1}^n$ issue 1: l_{max} has a trivial solution $\mathbf{w} = 0_{d\times 1}$ and $b = 0$ issue 2: l_{max} is non-differentiable at $\mathbf{w} = 0_{d\times 1}$ and $b = 0$
$soft(s_1, s_2) = log(e^{s_1} + e^{s_2})$	soft function approximating $max(s_1, s_2)$
$l_{softmax}(\mathbf{w}, b) = \sum_{i=1}^n soft(0, -y^i(\mathbf{x}^{iT}\mathbf{w} + b))$	soft max loss function approximating the original perceptron loss $l_{softmax}(\mathbf{w}, b) = \sum_{i=1}^n log(1 + e^{-y^i(\mathbf{x}^{iT}\mathbf{w}+b)})$ solves issue 1: $\mathbf{w} = 0_{d\times 1}$ and $b = 0$ is no trivial solution anymore solves issue 2: is differentiable
$min_{\mathbf{w},b}\ l_{softmax}(\mathbf{w}, b)$	this minimization problem is also referred to as logistic regression for classification and log-loss support vector machines (see lecture 8) it can be optimized using gradient descent of Newton's method
$y^{test} = sign(\mathbf{x}^{testT}\mathbf{w}^\star + b^\star))$	predicting the label y^{test} of a new testing sample $\mathbf{x}^{test}$ checks if sample $\mathbf{x}^{test}$ lies above $(+1)$ or below (-1) the estimated hyperplane parameterized by $(\mathbf{w}^\star, b^\star))$ $\mathbf{w}^\star$ is the normal vector to decision hyperplane
$1 - y^i(\mathbf{x}^{iT}\mathbf{w} + b) \leq 0$	margin perceptron (margin of size $2 = 1 + 1$) criterion for correctly classifying sample $\mathbf{x}^i$ constrained by predefined C-sized margin, $C = 2$ above the translated hyperplane $\mathbf{x}^T\mathbf{w} + b = 1$ or below the translated hyperplane $\mathbf{x}^T\mathbf{w} + b = -1$ $\leq$ is used since a data point can lie on either of translated hyperplanes (by $+1$ or -1) and still get correctly classified
$l_{margin}(\mathbf{w}, b) = \sum_{i=1}^n max(0, 1 - y^i(\mathbf{x}^{iT}\mathbf{w} + b))$	margin perceptron loss function for classifying samples $\{\mathbf{x}^i\}_{i=1}^n$ issue: non-differentiable
$l_{softmargin}(\mathbf{w}, b) = \sum_{i=1}^n log(1 + e^{1-y^i(\mathbf{x}^{iT}\mathbf{w}+b)})$	soft margin perceptron loss function for classifying samples $\{\mathbf{x}^i\}_{i=1}^n$ differentiable and convex $\rightarrow$ we can use gradient descent or Newton's method for optimization
$l_{squaredmargin}(\mathbf{w}, b) = \sum_{i=1}^n max(0, 1 - y^i(\mathbf{x}^{iT}\mathbf{w} + b))^2$	sqaured margin perceptron loss function for classifying samples $\{\mathbf{x}^i\}_{i=1}^n$ differentiable and convex $\rightarrow$ we can use gradient descent or Newton's method for optimization

Table 7: *Major mathematical notations used in lecture 8 (support vector machines).*

Mathematical notation	Definition
$\{\mathbf{x}^i, y^i\}_{i=1}^n$	training dataset where $\mathbf{x}^i \in \mathbb{R}^d$ denotes the feature vector for the i^{th} sample and $y^i \in \{-1, 1\}$ denotes its class label
$1 - y^i(\mathbf{x}^{iT}\mathbf{w} + b) \leq 0$	criterion for correctly classifying sample $\mathbf{x}^i$ for margin perceptron defined by hyperplanes $\mathbf{x}^T\mathbf{w} + b = 1$ and $\mathbf{x}^T\mathbf{w} + b = -1$ Note: both hyperplanes can be translated by $\pm m$, but it is customary to set $m = 1$ $1 - y^i(\mathbf{x}^{iT}\mathbf{w} + b) \leq 0$ is also equivalent to solving $y^i(\mathbf{x}^{iT}\mathbf{w} + b) - 1 \geq 0$
$\mathcal{L}(\mathbf{w}, b, \alpha)$	Lagrangian function, $\mathbf{w} \in \mathbb{R}^d$,$\alpha \in \mathbb{R}^n$, $\alpha_i \in \mathbb{R}$ for the i^{th} constraint for sample $\mathbf{x}^i$ $\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_{i=1}^n \alpha_i(y_i(\mathbf{x}^{iT}\mathbf{w} + b) - 1)$ $\mathcal{L}(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}^{iT}\mathbf{w} + b(\sum_{i=1}^n \alpha_i y_i) + \sum_{i=1}^n \alpha_i$
Solving constrained optimization problem	**Step 1:** solve the **primal** minimization problem: $min_{primal\ variables}\ \mathcal{L}$ Compute the partial derivatives of $\mathcal{L}$ wrt its primal variables and set them to zero solve $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{0}_d \rightarrow \mathbf{w}^\star = \sum_{i=1}^n \alpha_i y_i \mathbf{x}^i$ (1) solve $\frac{\partial \mathcal{L}}{\partial b} = 0 \rightarrow \sum_{i=1}^n \alpha_i y_i = 0$ (2) **Step 2:** solve the **dual** maximization problem: $max_{Lagrangian\ multipliers}\mathcal{L}(\mathbf{w}^\star, b^\star, \alpha)$ Use constraints (1-2) to estimate Lagrangian multipliers $\{\alpha_i\}_{i=1}^n$ by maximizing the dual Lagrangian (compute $\frac{\partial \mathcal{L}}{\partial \alpha} = \mathbf{0}_n$)
$\alpha_1^\star, \ldots, \alpha_n^\star = max_{\alpha_1,\ldots,\alpha_n}\mathcal{L}(\mathbf{w}^\star, b^\star, \alpha)$	defines the dual Lagrangian evaluated at optimal parameters $\mathbf{w}^\star, b^\star$ for SVM to maximize under positivity constraints $\alpha_i \geq 0, i \in 1, \ldots n$ and one equality constraint $\sum_{i=1}^n \alpha_i y_i = 0$
$\mathcal{L}(\mathbf{w}^\star, b^\star, \alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}^{iT}\mathbf{x}^j$	dual Lagrangian to maximize subject to $\alpha_i \geq 0, i \in 1, \ldots n$ and $\sum_{i=1}^n \alpha_i y_i = 0$
support vectors	$\alpha_i > 0$ for support vectors and $\alpha_i = 0$ otherwise (for non-support vectors)

There's another useful learning tool called *self-quizzing*. This learning strategy involves repeatedly recalling and testing yourself on what you've been taught. It's a technique designed to help new ideas stick in your long-term memory.

In fact, research has shown that self-quizzing is 50 percent more effective than some other learning strategies.

A 2006 Washington University study demonstrated just this. Researchers Jeffrey Karpicke and Henry Roediger gave a text to two groups of participants. The first group read it four times. The second group only read it once, but they practiced recalling it three times over.

When Karpicke and Roediger tested them all a few days later, they found that the passage had stuck significantly more in the minds of the group members who'd performed self-quizzing.

Fig. 1: *Blink 1 from 'Learn Better' by Ulrich Boser.*