Regularization

In the beginning, we have motivated the use of a loss function with a Bayesian formulation combining the probability of the data given the model and the probability of the model

$$\log \mu_W(w|\mathcal{D} = \mathbf{d}) = \log \mu_{\mathcal{D}}(\mathbf{d}|W = w) + \log \mu_W(w) - \log Z$$

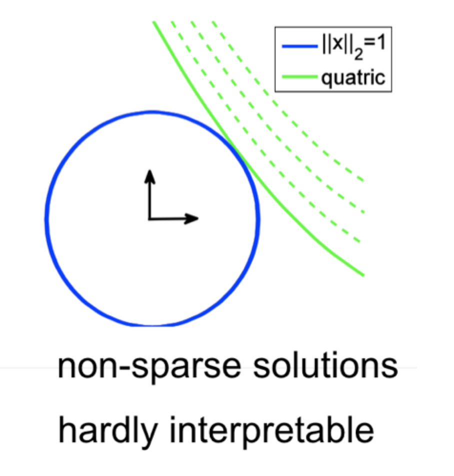
If μ_W is a Gaussian density with an isotropic covariance matrix $Cov = \lambda\mathbf{1}$ the log-prior $\log \mu_W(w)$ results in a quadratic penalty

$$\lambda\|w\|^2$$

Since this is convex, its sum with a convex functional is convex

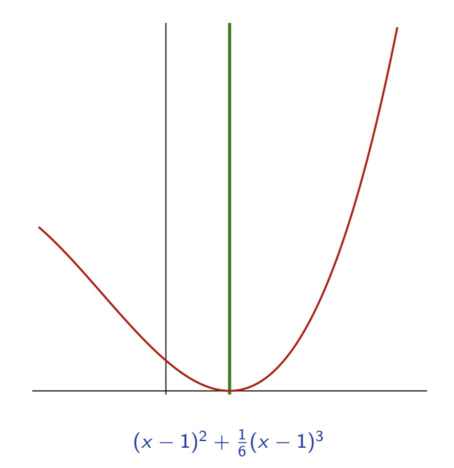
This is called the $\boldsymbol{\ell_2}$-**regularization**, or 'weight decay'

ℓ_2-norm visualization:



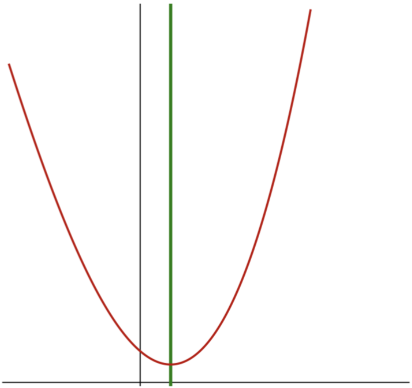
Increasing λ moves the optimum closer to 0 and away from the optimum for the loss alone

Since the derivative of $\|x\|^2$ is zero at zero, the optimum will never move there if it was not already there



Increasing λ moves the optimum closer to 0 and away from the optimum for the loss alone

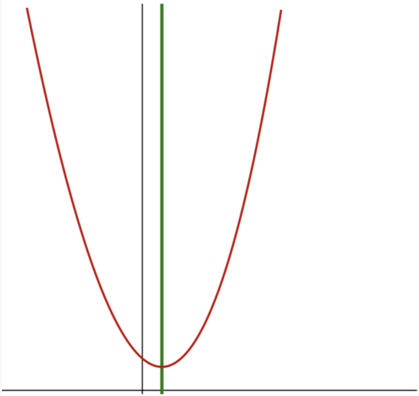
Since the derivative of $\|x\|^2$ is zero at zero, the optimum will never move there if it was not already there



$(x-1)^2 + \frac{1}{6}(x-1)^3 + x^2$

Increasing λ moves the optimum closer to 0 and away from the optimum for the loss alone

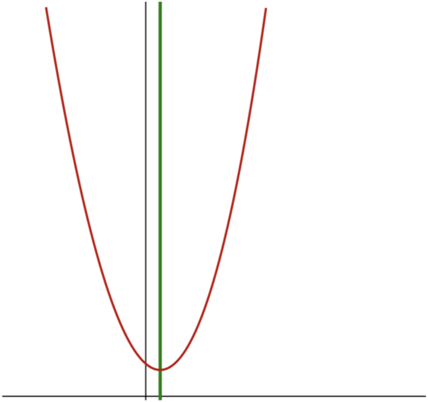
Since the derivative of $\|x\|^2$ is zero at zero, the optimum will never move there if it was not already there



$(x-1)^2 + \frac{1}{6}(x-1)^3 + 2x^2$

Increasing λ moves the optimum closer to 0 and away from the optimum for the loss alone

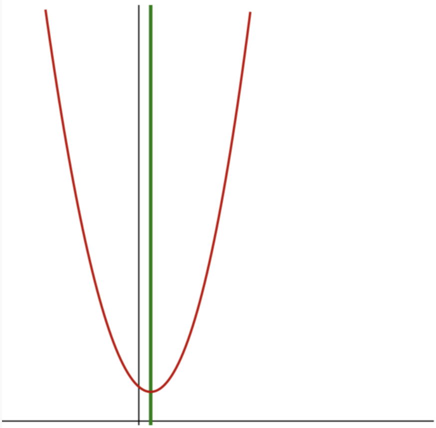
Since the derivative of $\|x\|^2$ is zero at zero, the optimum will never move there if it was not already there



$(x-1)^2 + \frac{1}{6}(x-1)^3 + 3x^2$

Increasing λ moves the optimum closer to 0 and away from the optimum for the loss alone

Since the derivative of $\|x\|^2$ is zero at zero, the optimum will never move there if it was not already there



$(x-1)^2 + \frac{1}{6}(x-1)^3 + 4x^2$

We can apply the exact same scheme with a Laplace prior

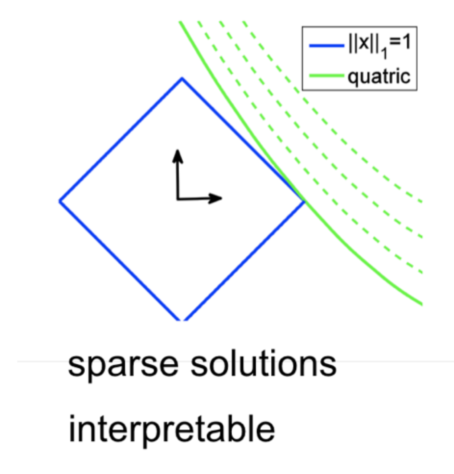
$$\begin{aligned}\mu(w) &= \frac{1}{(2b)^D} \exp\left(-\frac{\|w\|_1}{b}\right) \\ &= \frac{1}{(2b)^D} \exp\left(-\frac{1}{b}\sum_{d=1}^{D} |w|\right)\end{aligned}$$

which results in a penalty of the form

$$\lambda \|w\|_1$$

and is known as the **ℓ_1-regularization**. Similar to its ℓ_2 counterpart, the penalty is convex and its sum with a convex function also convex

ℓ_1-norm visualization:



An important property of the ℓ_1 regularization is that if E is convex and

$$w^* = \operatorname*{argmin}_w E(w) + \lambda \|w\|_1$$

then

$$\forall d, \left| \frac{\partial E}{\partial w_d} \right| < \lambda \Rightarrow w_d^* = 0$$

In practice it means that this penalty pushes some of the variables to zero, but in contrast to the ℓ_2 penalty, they remain there. ℓ_1 regularization requires the optimal solution to lie on a simplex

The λ parameter controls the sparsity of the solution.

With the ℓ_1 regularization, the update rule becomes

$$w_{t+1} = w_t - \eta g_t - \lambda \operatorname{sign}(w_t)$$

where sign is applied per component. This is almost identical to

$$\begin{aligned} w'_t &= w_t - \eta g_t \\ w_{t+1} &= w'_t - \lambda \operatorname{sign}(w'_t) \end{aligned}$$

This update may overshoot and result in a component of w'_t strictly on one side of 0 while the same component in w_{t+1} is strictly on the other

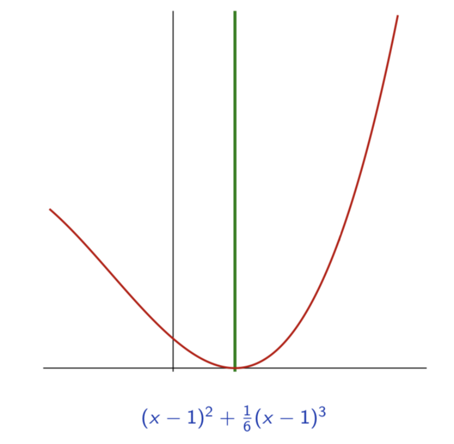
While this is not a problem in principle, since w_t will fluctuate around zero, it can be an issue if the zeroed weights are handled in a specific manner (e.g. sparse coding to reduce memory footprint or computation).

The **proximal operator** takes care of preventing parameters from 'crossing zero', by adapting λ when it is too large

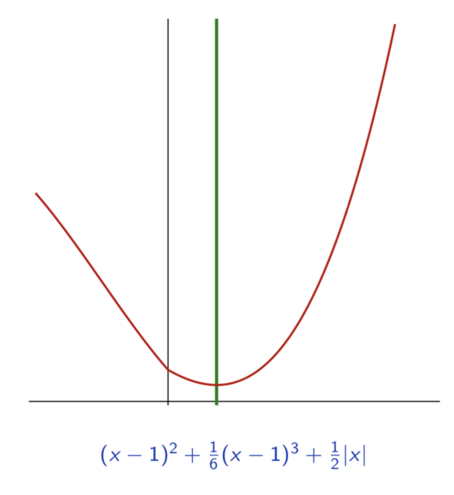
$$\begin{aligned} w'_t &= w_t - \eta g_t \\ w_{t+1} &= w'_t - \min(\lambda, |w'_t|) \odot \operatorname{sign}(w'_t) \end{aligned}$$

where min is component-wise and $\odot$ is the Hadamard component-wise product

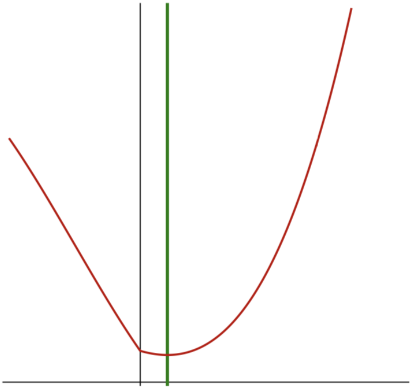
Increasing λ moves the optimum closer to 0, and away from the optimum of the loss without penalty



Increasing λ moves the optimum closer to 0, and away from the optimum of the loss without penalty

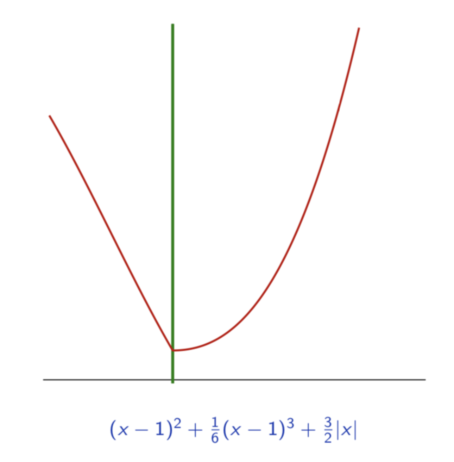


Increasing λ moves the optimum closer to 0, and away from the optimum of the loss without penalty

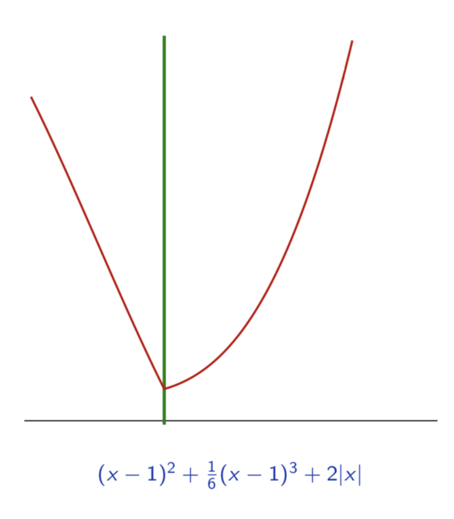


$(x-1)^2 + \frac{1}{6}(x-1)^3 + |x|$

Increasing λ moves the optimum closer to 0, and away from the optimum of the loss without penalty



Increasing λ moves the optimum closer to 0, and away from the optimum of the loss without penalty



In general, regularization is often useful when dealing with complex models and data at small scales. While they have a limited impact for large-scale deep learning, they may still provide the little push needed to beat baselines