Advanced Python

Chapter 6

Automatic Differentiation and Accelerated Linear Algebre

for Machine Linearing

Agenda

- Jax
- SVM
- Kernel
- SVM with JAX - Notebook

JAX

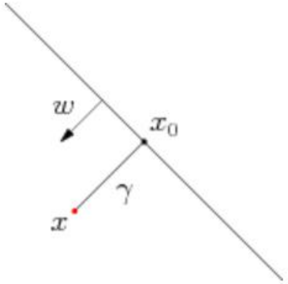
Developed by Google

Core Features of JAX:

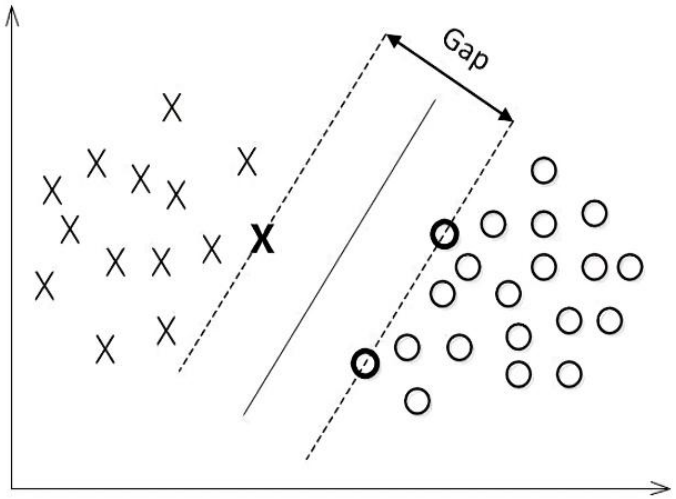
1. Autodiff: Efficient automatic differentiation for gradient calculations jax.grad
2. JIT Compilation: Accelerates code with the XLA compiler jax.jit
3. Vectorization & Parallelism: jax.vmap, jax.pmap
4. Compatibility with NumPy

Use Cases: Large-scale gradient calculations and parallel acceleration

SVM-1



$$\tilde{\gamma} = \frac{|w^T x + b|}{||w||} = \frac{y(w^T x + b)}{||w||}$$



Objective: max $\tilde{\gamma}$

Optimization Prob1 - hard margin(linearly separable)

$min\ \frac{1}{2}||w||^2\ s.t\ y_i(w^T x_i + b) \geq 1, i = 1, \ldots, n$

Lagrange

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i (y_i (w^T x_i + b) - 1)$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j\, y_i y_j x_i^T x_j$$

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \quad f(x) = \sum_{i=1}^{n} \alpha_i y_i < x_i, x > + b$$

only inner product -> Kernel -> complex, high dimensional spaces

Optimization Prob2 - soft margin and slack variables(linearly non-separable)

$$min\ \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \delta_i\ \ s.t\ y_i(w^T x_i + b) \geq 1 - \delta_i, \delta_i \geq 0, i = 1, \ldots, n$$

=> $min \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} max(0, 1 - y_i(w^T x_i + b))$

SVM-2

- Hinge loss is used for "maximum-margin" classification.

$$l(y) = max(0, 1 - t * y)$$

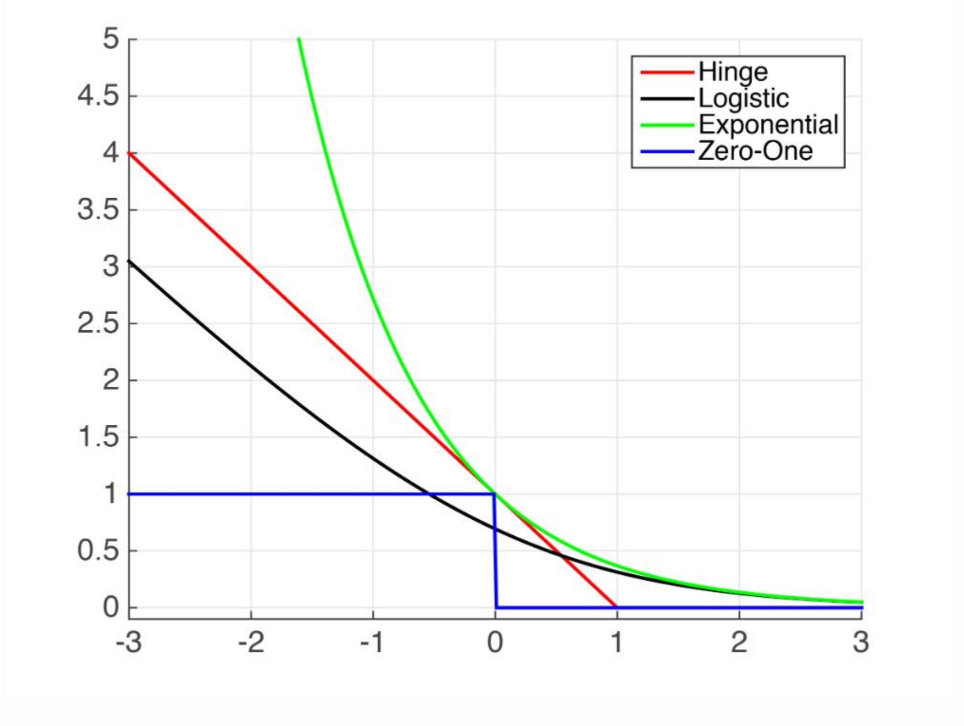


Figure 4.1: Plots of Common Classification Loss Functions - x-axis: $h(\mathbf{x}_i)y_i$, or "correctness" of prediction; y-axis: loss value

$$\min_{\mathbf{w}} \; \underbrace{C\sum_{i=1}^{n} \max[1 - y_i \underbrace{(w^\top \mathbf{x}_i + b)}_{h(\mathbf{x}_i)}, 0]}_{Hinge-Loss} + \underbrace{\| w \|_2^2}_{l_2-Regularizer}$$

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \underbrace{\ell(h_{\mathbf{w}}(\mathbf{x}_i), y_i)}_{Loss} + \underbrace{\lambda r(w)}_{Regularizer}$$

Github:
https://github.com/nfmcclure/tensorflow_cookbook/tree/master/04_Support_Vector_Machines

Source: https://www.cs.cornell.edu/courses/cs4780/2023sp/lectures/lecturenote10.html

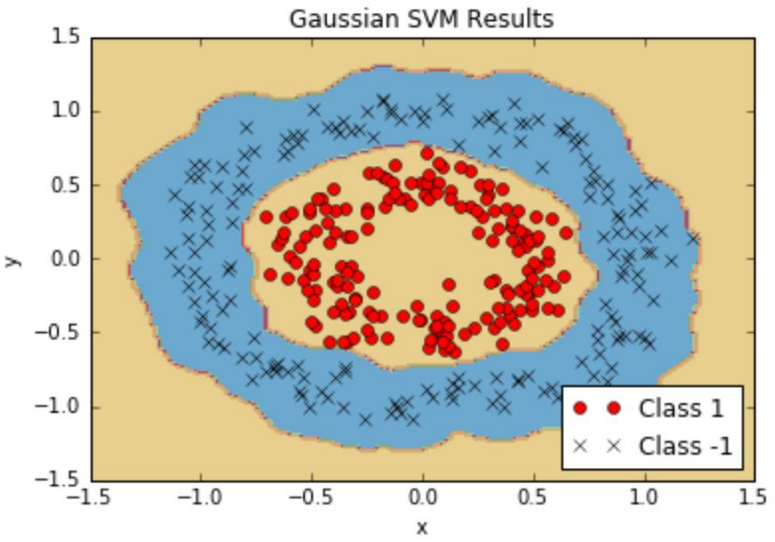
Kernel

- Many learning methods rely on training and test data only in form of inner products, e.g regularized least sqaure, NN, SVM.
- kernel function φ_i to introduce nonlinearity

$$f(x) = \sum_{i=1}^{N} w_i \varphi_i(x) + b$$

SVM - $f(x) = \sum_{i=1}^{N} w_i < \varphi_i(x), \varphi(x) > + b$

- $K(x_i, x_j) =< \varphi(x_i), \varphi(x_j) >$
- Gaussian Kernel $K(x_i, x_j) = exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$



Appendix

Loss $\ell(h_{\mathbf{w}}(\mathbf{x}_i, y_i))$	Usage	Comments
Hinge-Loss $\max\left[1 - h_{\mathbf{w}}(\mathbf{x}_i)y_i, 0\right]^p$	• Standard SVM($p = 1$) • (Differentiable) Squared Hingeless SVM ($p = 2$)	When used for Standard SVM, the loss function denotes the size of the margin between linear separator and its closest points in either class. Only differentiable everywhere with $p = 2$.
Log-Loss $\log(1 + e^{-h_{\mathbf{w}}(\mathbf{x}_i)y_i})$	Logistic Regression	One of the most popular loss functions in Machine Learning, since its outputs are well-calibrated probabilities.
Exponential Loss $e^{-h_{\mathbf{w}}(\mathbf{x}_i)y_i}$	AdaBoost	This function is very aggressive. The loss of a mis-prediction increases *exponentially* with the value of $-h_{\mathbf{w}}(\mathbf{x}_i)y_i$. This can lead to nice convergence results, for example in the case of Adaboost, but it can also cause problems with noisy data.
Zero-One Loss $\delta(\text{sign}(h_{\mathbf{w}}(\mathbf{x}_i)) \neq y_i)$	Actual Classification Loss	Non-continuous and thus impractical to optimize.

Table 4.1: Loss Functions With Classification $y \in \{-1, +1\}$

Loss $\ell(h_{\mathbf{w}}(\mathbf{x}_i, y_i))$	Comments
Squared Loss $(h(\mathbf{x}_i) - y_i)^2$	◦ Most popular regression loss function ◦ Estimates Mean Label ◦ Also known as Ordinary Least Squares (OLS) ◦ 🙂 Differentiable everywhere ◦ 😡 Somewhat sensitive to outliers/noise
Absolute Loss $\|h(\mathbf{x}_i) - y_i\|$	◦ Also a very popular loss function ◦ Estimates Median Label ◦ 🙂 Less sensitive to noise ◦ 😡 Not differentiable at 0
Huber Loss ◦ $\frac{1}{2}(h(\mathbf{x}_i) - y_i)^2$ if $\|h(\mathbf{x}_i) - y_i\| < \delta$, ◦ otherwise $\delta(\|h(\mathbf{x}_i) - y_i\| - \frac{\delta}{2})$	◦ Also known as Smooth Absolute Loss ◦ "Best of Both Worlds" of Squared and Absolute Loss ◦ Once-differentiable ◦ Takes on behavior of Squared-Loss when loss is small, and Absolute Loss when loss is large.
Log-Cosh Loss $log(cosh(h(\mathbf{x}_i) - y_i))$, $cosh(x) = \frac{e^x + e^{-x}}{2}$	◦ 🙂 Similar to Huber Loss, but twice differentiable everywhere ◦ 😡 More expensive to compute

Table 4.2: Loss Functions With Regression, i.e. $y \in \mathbb{R}$

Source: https://www.cs.cornell.edu/courses/cs4780/2023sp/lectures/lecturenote10.html