Machine Learning

Multiclass classification and Learning as Optimization

Dan Goldwasser

dgoldwas@purdue.edu

MIDTERM

Midterm!

- Hour long exam.
- 105 possible points
  - 5 bonus points
- **Allowed**: Cheatsheet, calculator, pen
- **Not Allowed**: anything else!

Midterm Question

- Short Question:
  - **True/False:** *"the decision boundary of perceptron and dual perceptron with RBF kernel are similar"*
  - **How would you../what would be the result of doing:**
    - *Reducing the size of a decision tree/changing learning rate.*

- *Short answers (1-2 sentences), consisting of answer (e.g., true/false) and a short explanation*
- ***Avoid guessing**. We really just care about the explanation*

Midterm Question

- Calculation Questions:
  - Simulate an algorithm run on some data (small set)
  - Understand the principle behind the algorithm’s performance

- Bring a calculator, mostly so you don’t waste time.
- If you did the HW you should be fine.
- Make sure answers to questions are consistent with the algorithm ”run”

Midterm Questions

- Algorithmic Questions:
  - Adapt the algorithms we have seen to work better in a given setting (little data, high dimensional data, level of noise, etc.)
  - Adapt the algorithms we have seen to new scenarios (come up with new algorithms)
    - We have seen algorithms for learning monotone conjunctions, computing kernels etc. *How can these algorithms be changed?*
  - Analyze algorithms performance

Midterm Questions

- Theoretical Question:
  - Combinatorial questions (counting) and the connection to relevant to ML concepts
  - Show that an algorithm is a mistake bound algorithm
  - Understand the difference between hypothesis classes

- *Make sure to review the definitions and understand them!*

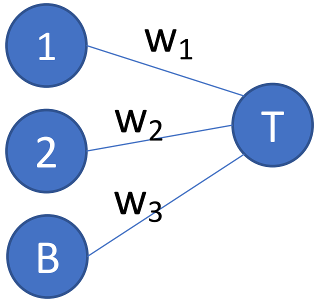
Midterm Topics

- KNN
- Decision Trees
- Learning Boolean Functions
- Winnow
- Perceptron
- Multi-class classification
- Gradient Descent (as far as we’ll get today)
- Mistake Bounds analysis, Perceptron convergence
- Performance evaluation, cross-validations, overfitting and underfitting.
- Connections between concepts (how does DT control overfitting?)

- **Explain**. What is one similarity and one difference between..
  - Winnow and Perceptron
- **Pick an option and explain**.
  - Overfitting is more likely in Decision trees or Perceptron.
- **True or False.** A dual perceptron with a linear kernel has the same expressive power as a primal perceptron.
- **True or False**: the size of the hypothesis space (e.g., 3^n for conjunctions, 2^2^n for Boolean Functions) is a good indicator for the expressiveness of the space

Example Question

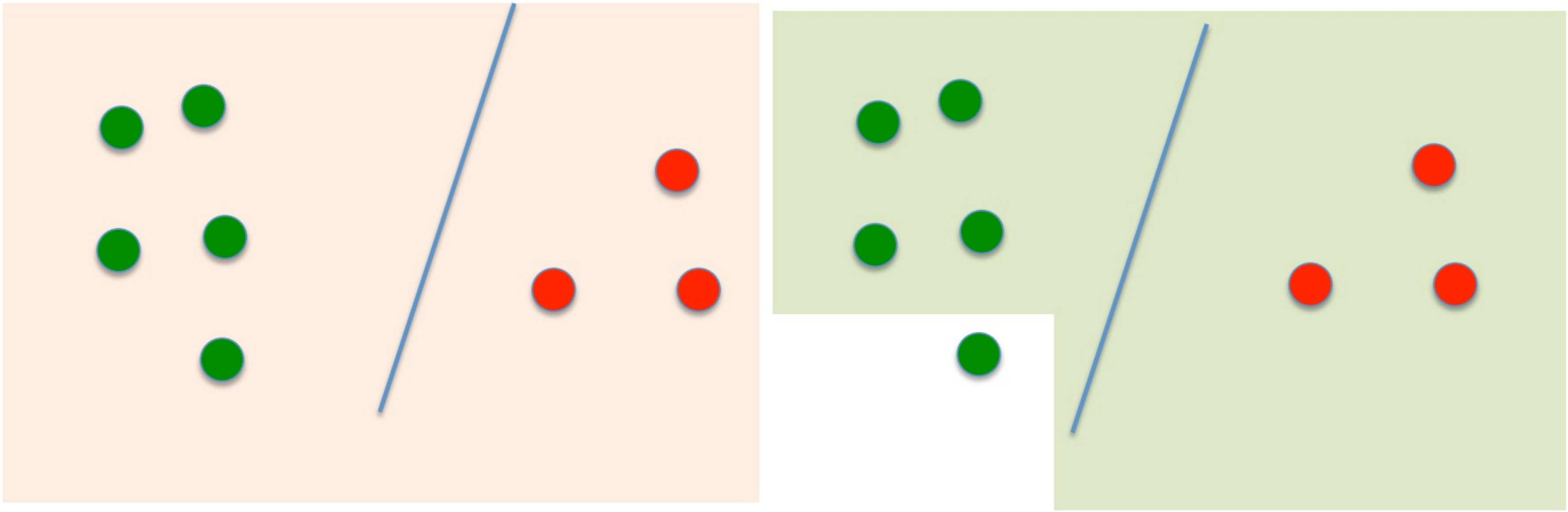
- How would you represent an OR and AND function using a linear threshold function?



- Would a perceptron be able to learn these functions?

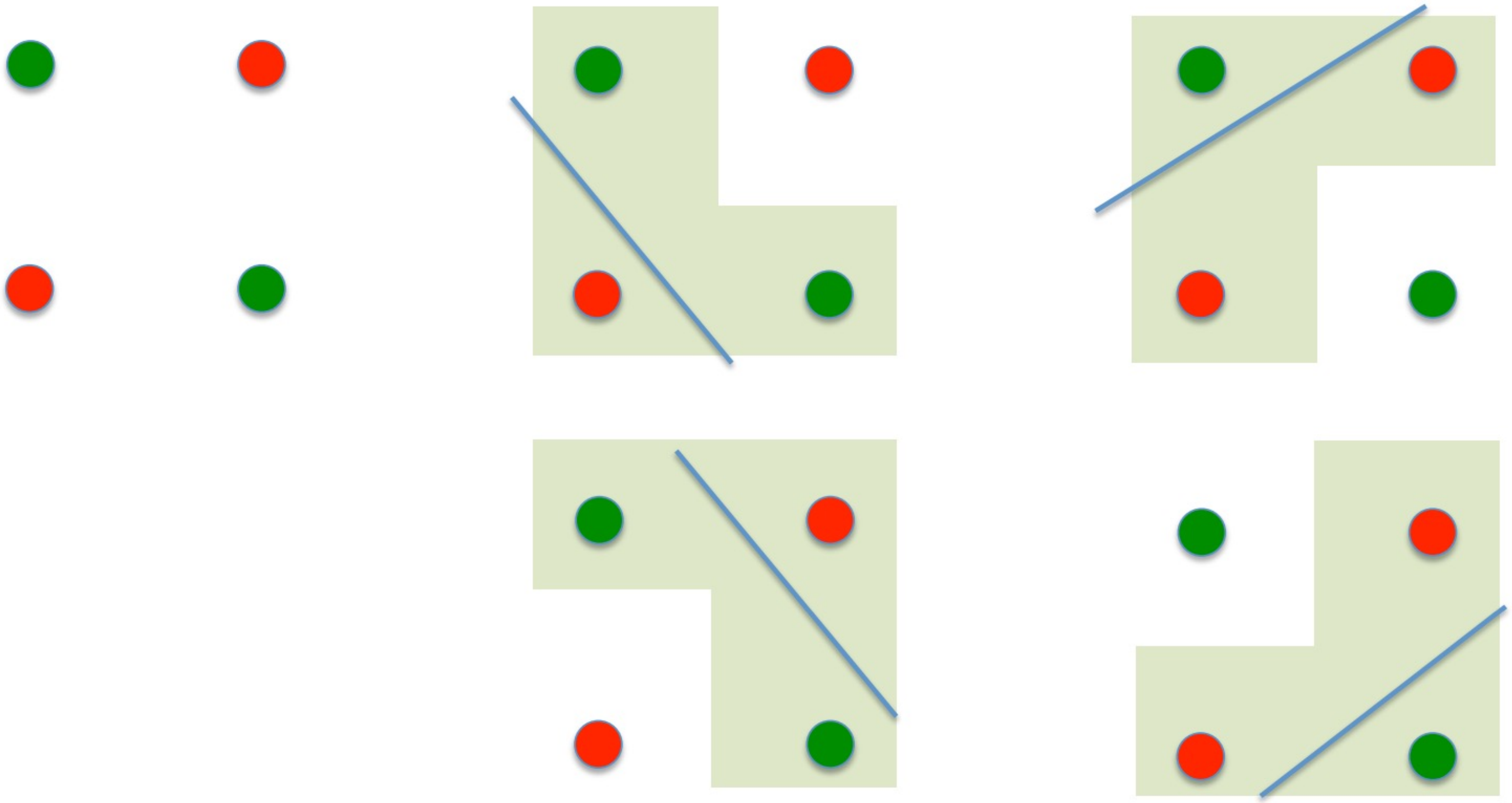
Example Question

place points and label them s.t *perceptron will always have zero training error and non-zero leave-one-out validation error*



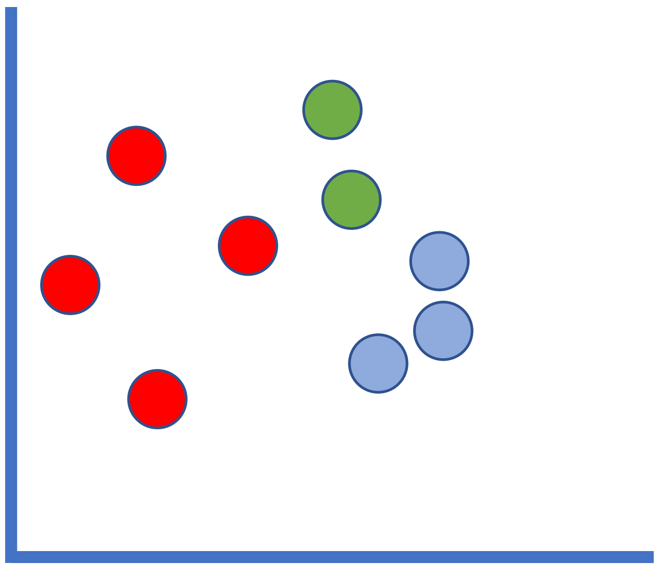
Example Question

place points and label them s.t *perceptron will always have zero training error and non-zero leave-one-out validation error*



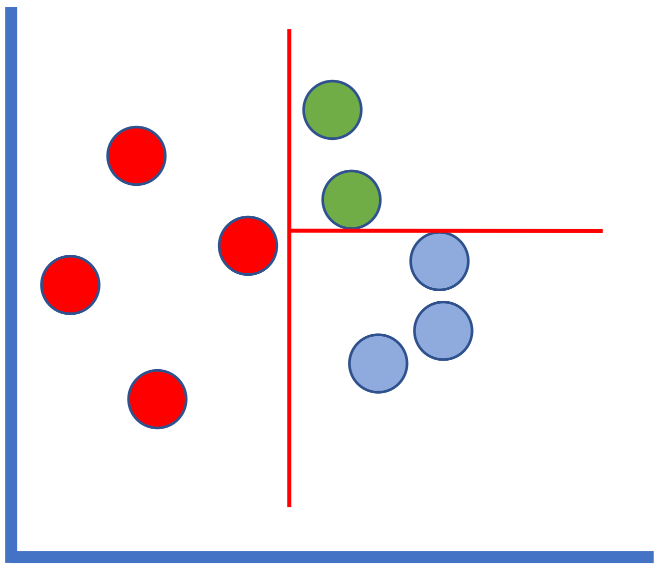
Example Questions

- What will be the result of decision tree learning vs. multi-class perceptron on this dataset?



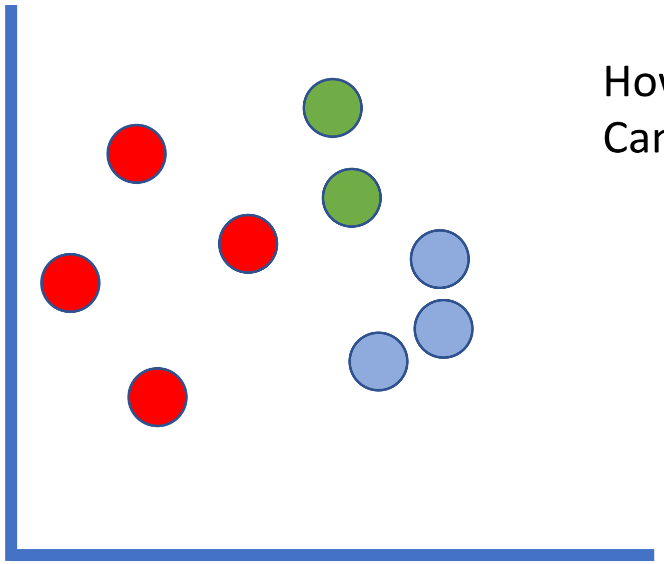
Example Questions

- What will be the result of decision tree learning vs. multi-class perceptron on this dataset?



Example Questions

- What will be the result of decision tree learning vs. multi-class perceptron on this dataset?



How would it look for Perceptron?
Can you “draw” it on the slide?

Reminder: Loss functions

- To formalize performance let's define a *loss function:*

$$loss(y, \hat{y})$$

  - Where $\hat{y}$ is the gold label

- *The loss function measures the error on a single instance*
  - Specific definition depends on the learning task

Regression

$$loss(y, \hat{y}) = (y - \hat{y})^2$$

Binary classification

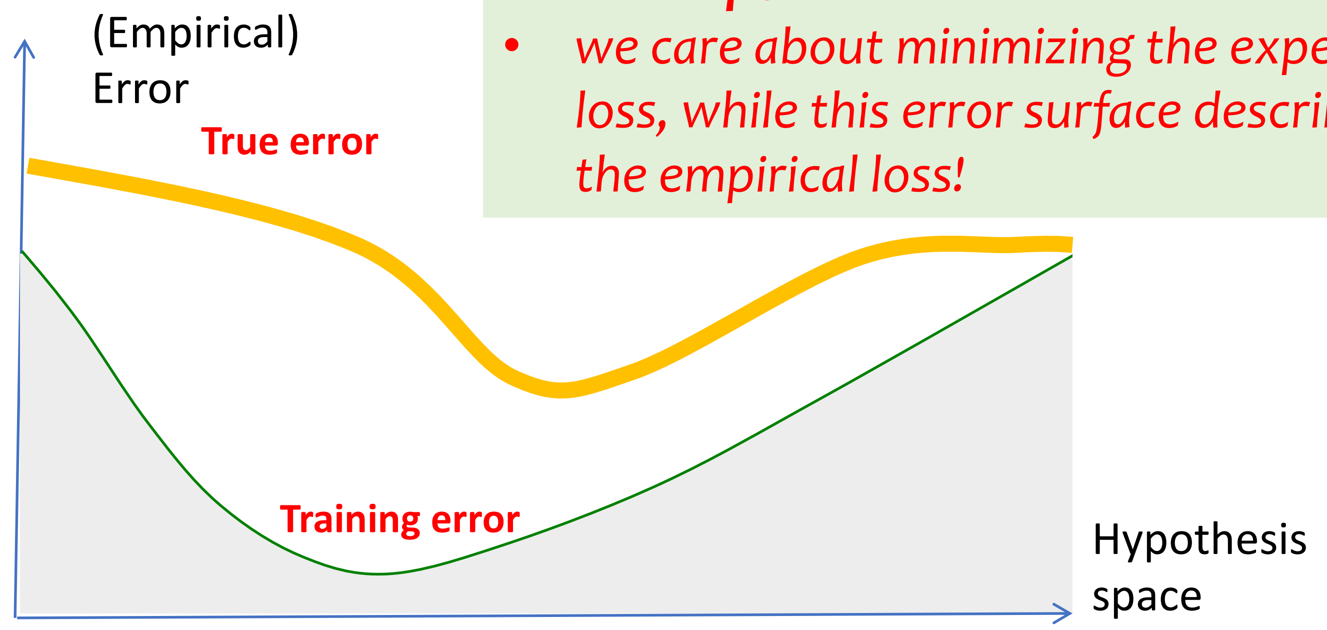
$$loss(y, \hat{y}) = \begin{cases} 0 & y = \hat{y} \\ 1 & otherwise \end{cases}$$

Loss minimization

- **Let's consider the square loss.**
  - Convex loss function, error surface has a global minimum (~any local minimum is also global).

Do we really want to get to that global minimum point?

- *we care about minimizing the expected loss, while this error surface describes the empirical loss!*



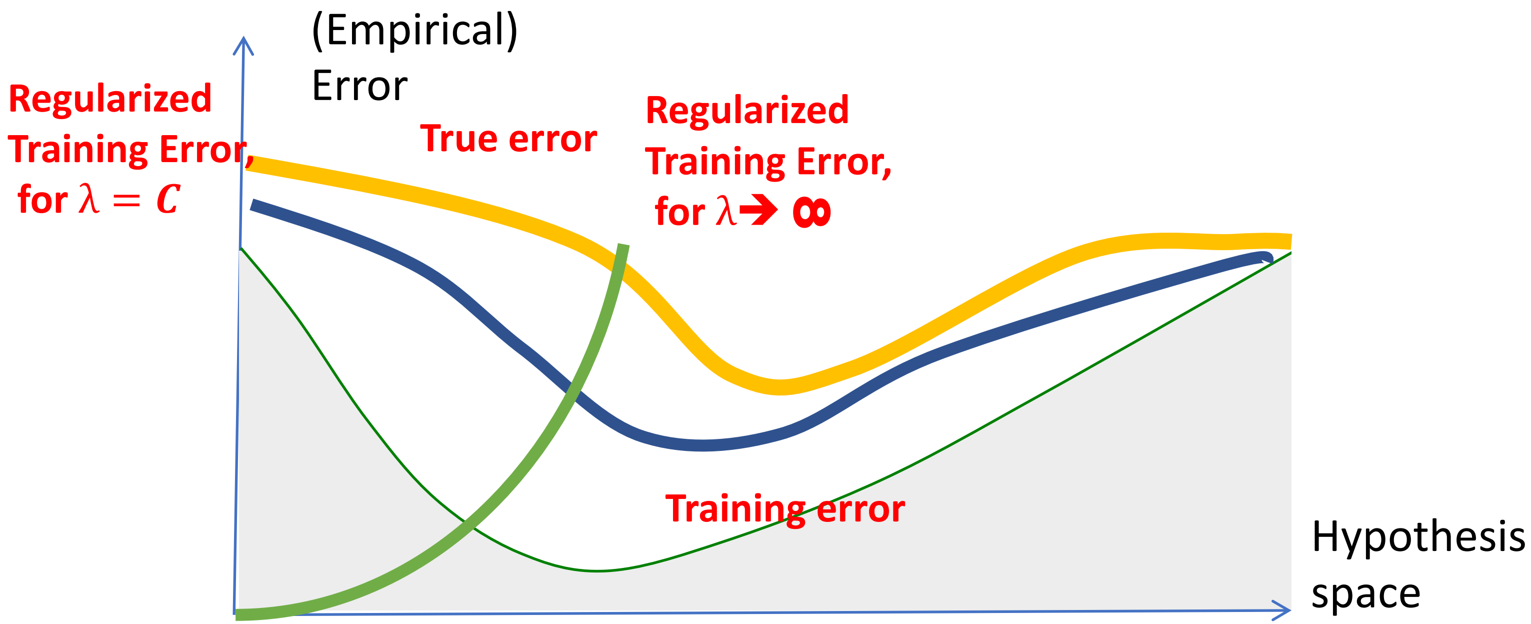
Regularization

- ***A form of inductive bias – we prefer simpler functions!***
- A very popular choice of regularization term is to minimize the norm of the weight vector
  - For convenience: ½ squared norm

$$\min_{\mathrm{w}} \quad \sum_{n} loss(y_n, \mathrm{w}_n) + \frac{\lambda}{2}||\mathrm{w}||^2$$

Loss minimization

- **Let's consider the square loss.**
  - Convex loss function, error surface has a global minimum (~any local minimum is also global).



Gradient Descent for Squared Loss

```
Initialize w⁰ randomly
for i = 0...T:
    Δw = (0, ...., 0)
    for every training item d = 1...D:
        f(x_d) = w^i·x_d
        for every component of w     j = 0...N:
            Δw_j += α(y_d – f(x_d))·x_dj
    w^{i+1} = w^i + Δw
    return w^{i+1} when it has converged
```

Initialize $\mathbf{w}^0$ randomly

for $i = 0 \ldots T$:

$\quad \Delta\mathbf{w} = (0, \ldots., 0)$

$\quad$ for every training item $d = 1 \ldots D$:

$\quad\quad f(\mathbf{x}_d) = \mathbf{w}^i \cdot \mathbf{x}_d$

$\quad\quad$ for every component of $\mathbf{w}$ $\quad j = 0 \ldots N$:

$\quad\quad\quad \Delta w_j \mathrel{+}= \alpha(y_d - f(\mathbf{x}_d)) \cdot x_{dj}$

$\quad \mathbf{w}^{i+1} = \mathbf{w}^i + \Delta\mathbf{w}$

$\quad$ return $\mathbf{w}^{i+1}$ when it has converged

Stochastic Gradient Descent

Initialize $\mathbf{w^o}$ randomly

for m = 0...M:

$$f(\mathbf{x_m}) = \mathbf{w^i} \cdot \mathbf{x_m}$$

$$\Delta w_j = \alpha(y_d - f(\mathbf{x_m})) \cdot x_{mj}$$

$$\mathbf{w^{i+1}} = \mathbf{w^i} + \Delta \mathbf{w}$$

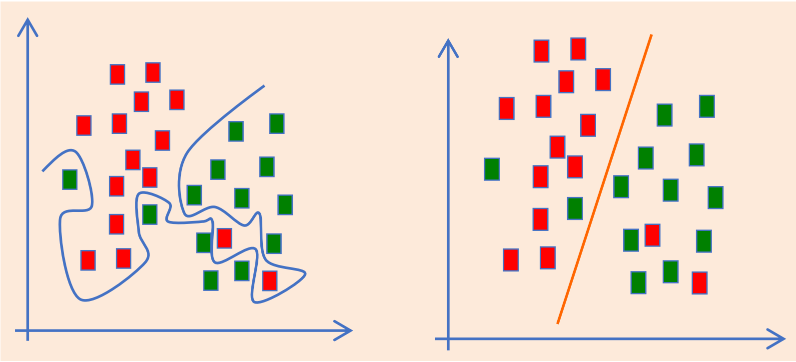
return $\mathbf{w^{i+1}}$ when it has converged

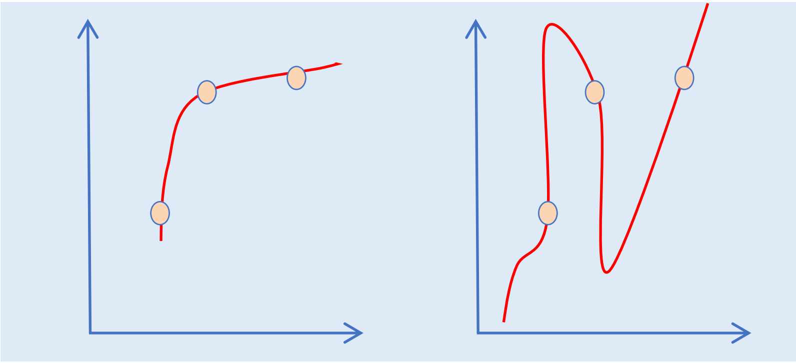
Regularization

- Both for regression and classification, for a given error we prefer a *simpler model*
  - *Keep W small: ε changes in the input cause ε*w in the output*
- *Some times we are even willing to trade a higher error rate for a simpler model (why?)*
- *Add a regularization term:*
  - *This is a form of **inductive bias***

$$\min_{\mathrm{w}} = \sum_{n} loss(y_n, \mathrm{w}_n) + \lambda R(\mathrm{w})$$

How different values affect learning?

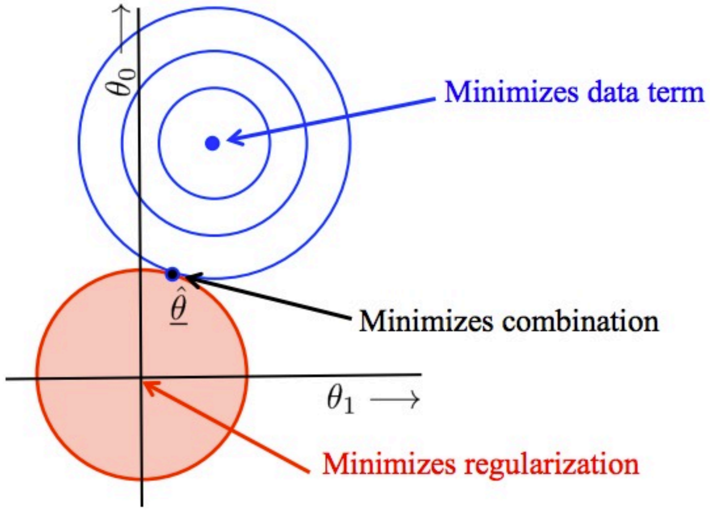




L1 and L2 norms

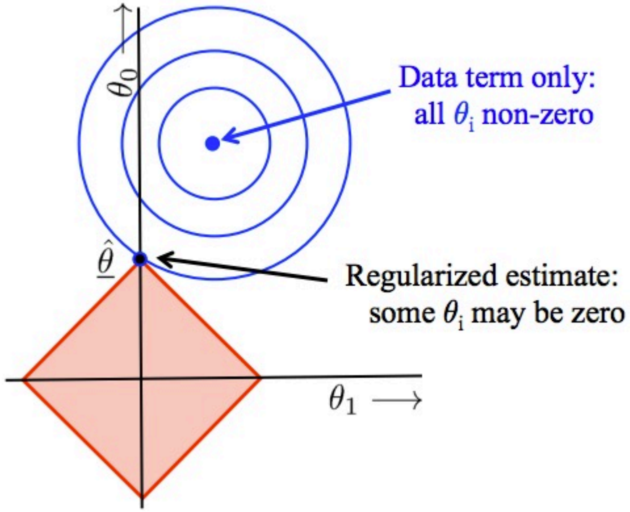
- Common choices: L1 and L2 are both convex ($L_{p<1}$ *is not convex*)
- *Regularized objective*: balance between minimizing the error and the regularization cost

L2 optimum will be sparse, ONLY if the data loss term is minimized at the axis



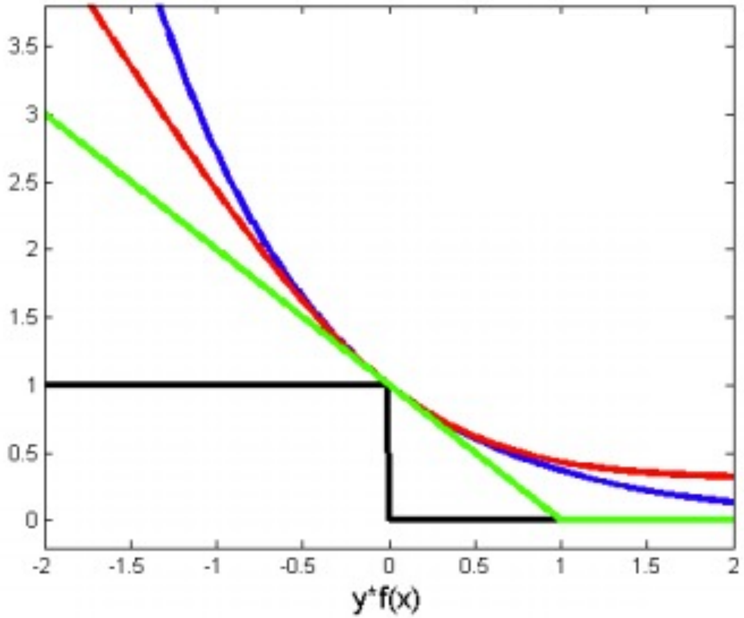
L1 norm contour are ***sharp****, will intersect with the contour of data loss term even when the data loss term min point is not at the axis*

➔ *L1 encourages sparsity*

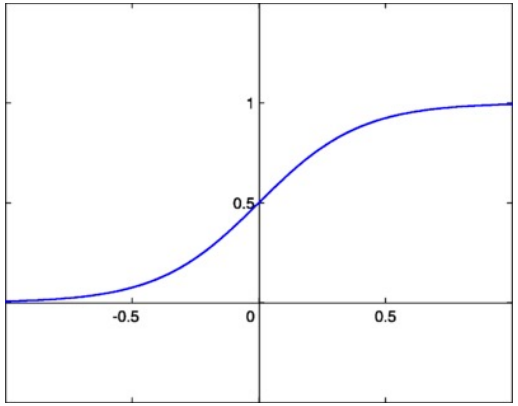


Surrogate Loss functions

- Surrogate loss function: smooth approximation to the 0-1 loss
  - Upper bound to 0-1 loss



Logistic Regression



$$h_{\mathrm{w}}(x) = g(\mathrm{w}^T x)$$

$$z = \mathrm{w}^T x$$

$$g(z) = \frac{1}{1+e^{-z}}$$

Sigmoid (logistic) function

Known as a sigmoid/logistic function

- Smooth transition between 0-1

- **Can be interpreted as the conditional probability**
- Decision Boundary
  - y=1: h(x) >0.5➔ $w^t x>=0$
  - Y=0: h(x) <0.5 ➔$w^t x<0$

- **Learning**: optimize the likelihood of the data
  - **Likelihood**: *probability of our data under current parameters*
  - For easier optimization, we look into the log likelihood (*negative*)

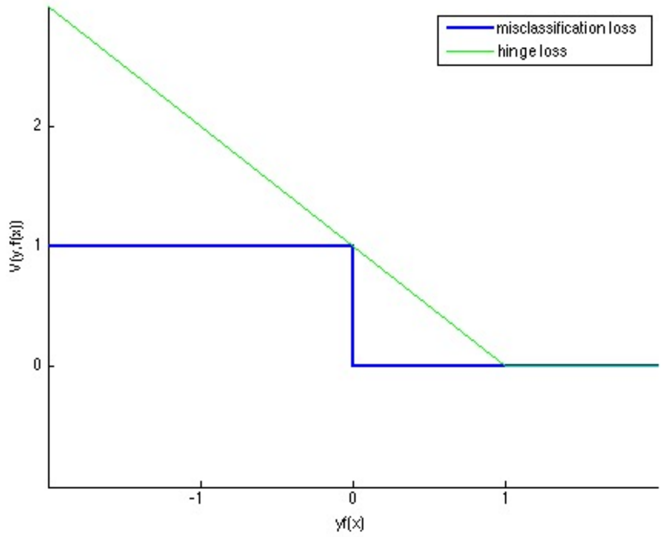
$$Err(\mathrm{w}) = -\sum_i y^i log(e^{g(w,x_i)}) + (1-y^i)log(1-e^{g(w,x_i)}) + \frac{1}{2}\lambda||w||^2$$

Hinge Loss

- Another popular choice for loss function is the hinge loss

$$L(y, f(x)) = \max(0, 1 - y f(x))$$

  - We will discuss in the context of support vector machines (SVM)



It's easy to observe that:

(1) The hinge loss is an upper bound to the 0-1 loss

(2) The hinge loss is a good approximation for the 0-1 loss

(3) BUT …

It is not differentiable at $y(w^T x)=1$

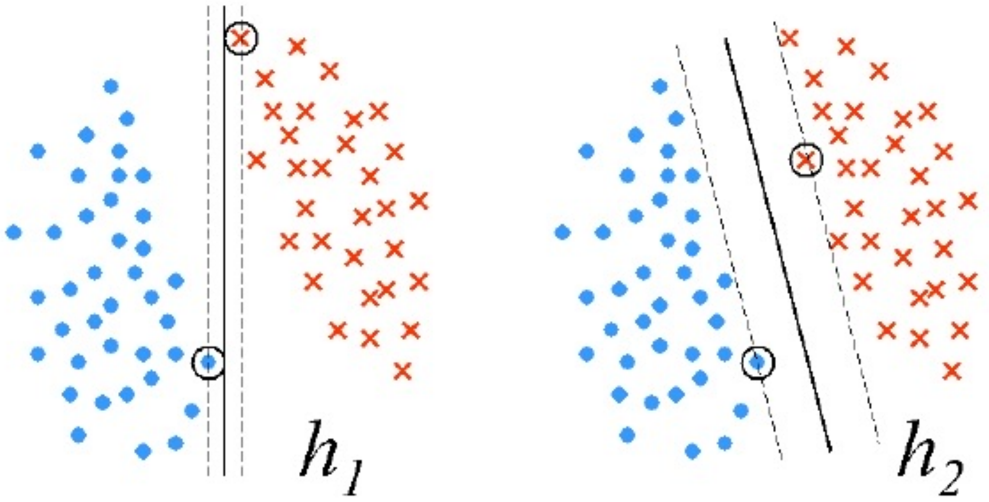
Solution: Sub-gradient descent

Reminder: Margin of a classifier

- Distance between a separator (hyperplane) and an example (point)

$$\frac{y(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})}{||\boldsymbol{w}||}$$

- Margin: the value that minimizes that distance for a given dataset.
- Larger margin can be indicative of better generalization



Hard SVM Intuition

The margin of a classifier: *the distance of the nearest point*

Recall: $\frac{y(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x})}{||\boldsymbol{w}||}$

We want to find the max margin classifier: $argmax_w\, [y(w^T x)\ /\ ||w||]$

If we fix $||w|| = 1$, we can focus on maximizing the functional margin

$$w^* = argmax_{||w||=1} \min_{(x,y) \in S} \overbrace{y\,(w^T x)}$$

Or, fix the functional margin $y(w^T x) \geq 1$, *and focus on minimizing* $||w||$

$$w^* = argmin\,||w|| \quad \text{s.t. } y(w^T x) \geq 1$$

Hard SVM Optimization

- We have shown that the sought-after weight vector **w** is the solution of the following optimization problem:

SVM Optimization:

Minimize: $\frac{1}{2} ||w||^2$

Subject to: $\forall$ (x,y) $\forall$ S: $y\, w^T x \geq 1$

- This is an optimization problem in (n+1) variables, with |S|=m inequality constraints.

Non-Separable Case

Want to relax the constraints:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1.$$

Introduce slack variables ξ_i:

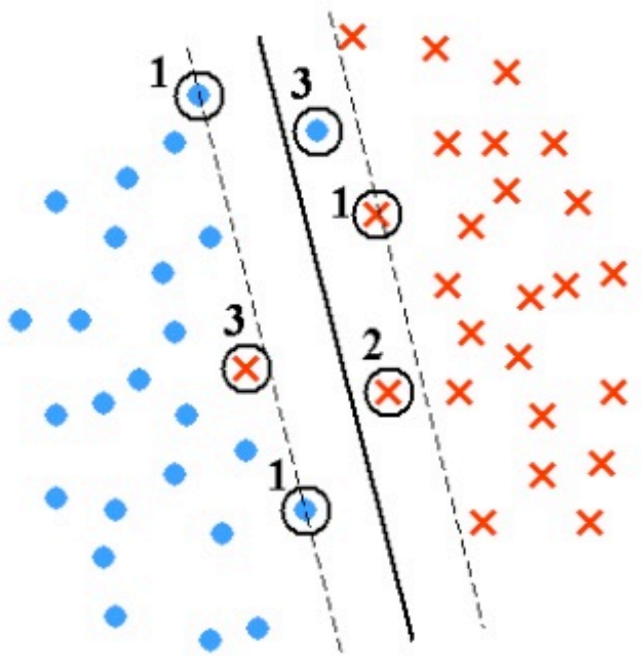
$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i,$$

Where $\xi_i \geq 0$, *an error occurs when $\xi_i > 0$*

Thus we can assign an extra cost for errors, as follows:

$$\begin{array}{ll} \text{Minimize} & f(\mathbf{w}, b, \boldsymbol{\xi}) \equiv \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{m} \xi_i \\ \text{subject to} & y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i; \quad \xi_i \geq 0, \quad i = 1, \ldots, m \end{array}$$

Visualizing Solution in the non-Separable Case



1. Margin support vectors $\xi_i = 0$ Correct
2. Non-margin support vectors $\xi_i < 1$ Correct (in margin)
3. Non-margin support vectors $\xi_i > 1$ Error

Soft SVM

- Notice that the relaxation of the constraint: $y_i\, w_i^t x_i \geq 1$ can be done by introducing a slack variable ξ (per example) and requiring:

$$y_i\, w_i^t x_i \geq 1 - \xi_i \quad ; \quad \xi_i \geq 0$$

- Now, we want to solve:

$$\text{Min } \tfrac{1}{2}\, ||w||^2 + c \sum_i \xi_i \quad \text{subject to } \xi_i \geq 0$$

- ***Which can be written as:***

$$\text{Min } \tfrac{1}{2}\, w^T w + c \sum_i \max(0, 1 - y_i\, w_i^t x_i).$$

Soft SVM (2)

- The hard SVM formulation assumes linearly separablity.
  - *A natural relaxation*: maximize the margin while minimizing the # of examples that violate the margin (separability) constraints.
  - *This leads to non-convex problem that is hard to solve.*
  - *Instead*, move to a surrogate loss function that is convex.

- SVM relies on the **hinge loss** function:

$$\mathrm{Min}_w \ \tfrac{1}{2} ||w||^2 + c \sum_{i\ (x,y) \in S} \max(0, 1 - y\, w^t x)$$

- where the parameter c controls the tradeoff between large margin (small $||w||$) and small hinge-loss.

SVM Objective Function

General Form of a learning algorithm:

- **Minimize** empirical loss, and **Regularize** (to avoid over fitting)

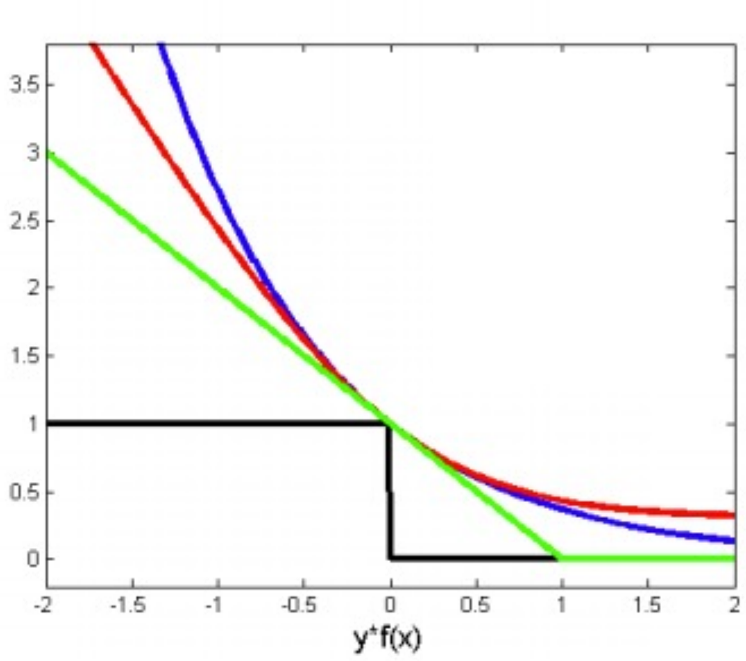
$$\text{Min}\ \underbrace{\tfrac{1}{2}\,||w||^2}_{\text{Regularization term}} + c\ \underbrace{\Sigma \max(0, 1 - y_i\, wx_i)}_{\text{Empirical loss}}$$

Can be replaced by other **regularization functions**

Can be replaced by other **loss functions**

Surrogate Loss functions

- Surrogate loss function: smooth approximation to the 0-1 loss
  - Upper bound to 0-1 loss

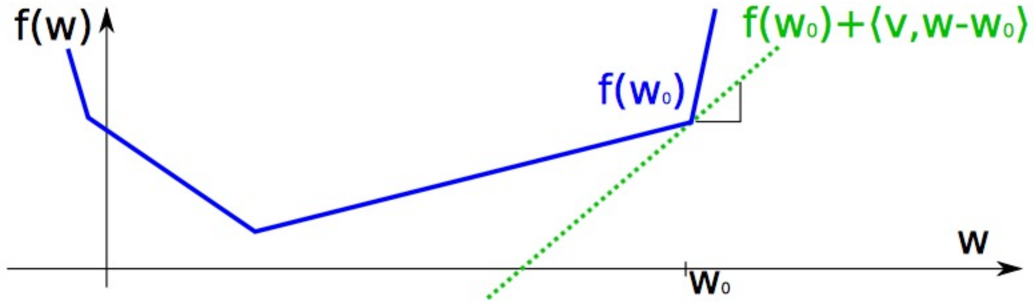


Subgradient descent

- asda

Let $f : \mathbb{R}^D \to \mathbb{R}$ be a convex, not necessarily differentiable, function.
A vector $v \in \mathbb{R}^D$ is called a **subgradient** of f at w_0, if

$$f(w) \geq f(w_0) + \langle v, w - w_0 \rangle \quad \text{for all } w.$$

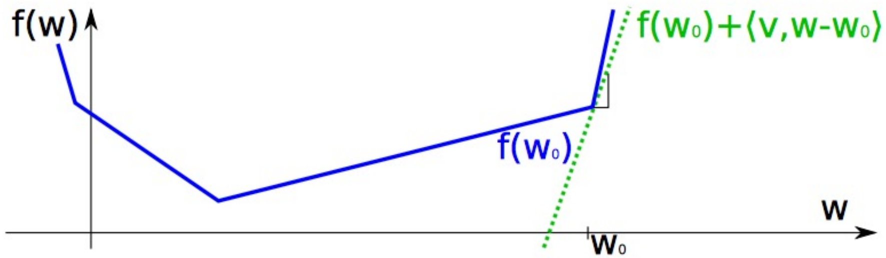


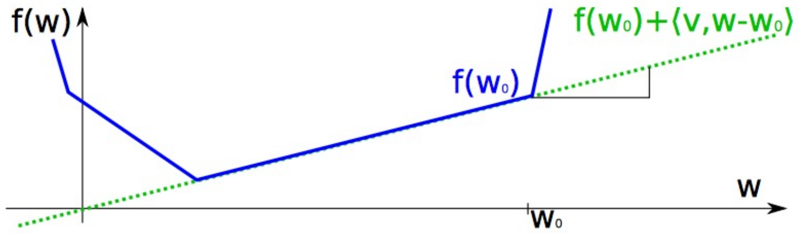
Slides by Sebastian Nowozin and Christoph H. Lampert “structured models in computer vision” tutorial CVPR 2011

Subgradient descent

Let $f : \mathbb{R}^D \to \mathbb{R}$ be a convex, not necessarily differentiable, function.
A vector $v \in \mathbb{R}^D$ is called a **subgradient** of f at w_0, if

$$f(w) \geq f(w_0) + \langle v, w - w_0 \rangle \quad \text{for all } w.$$





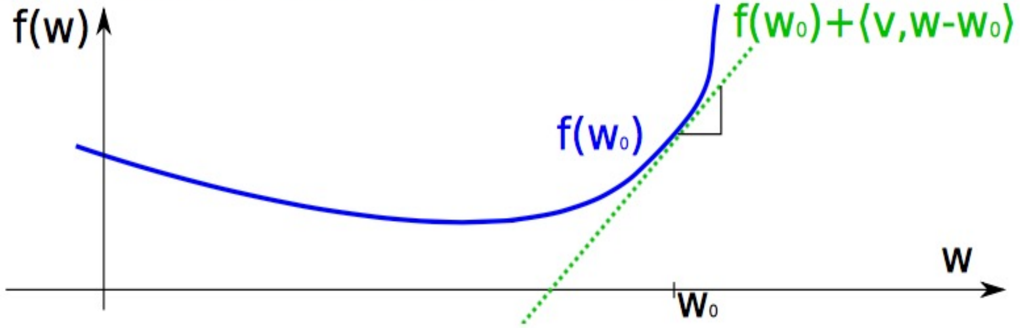
Slides by Sebastian Nowozin and Christoph H. Lampert “structured models in computer vision” tutorial CVPR 2011

Subgradient descent

Let $f : \mathbb{R}^D \to \mathbb{R}$ be a convex, not necessarily differentiable, function.
A vector $v \in \mathbb{R}^D$ is called a **subgradient** of f at w_0, if

$$f(w) \geq f(w_0) + \langle v, w - w_0 \rangle \quad \text{for all } w.$$

For differentiable f, the gradient $v = \nabla f(w_0)$ is the only subgradient.



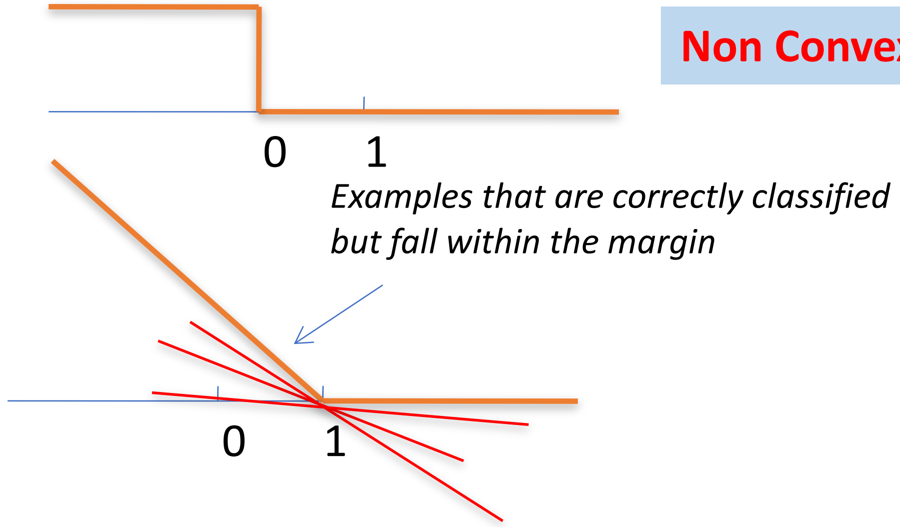
Slides by Sebastian Nowozin and Christoph H. Lampert "structured models in computer vision" tutorial CVPR 2011

Sub-Gradient

Standard 0/1 loss

Penalizes all incorrectly classified examples with the same amount

Non Convex



Hinge loss

Penalizes incorrectly classified examples and correctly classified examples that lie within the margin

Convex,
but not differentiable at x=1

Solution: *subgradient*

The ***sub-gradient*** *of a function c at* x_0 *is any vector v such that:* $\forall x : c(x) - c(x_0) \geq v \cdot (x - x_0).$

At ***differentiable*** *points this set only contains the gradient at* x_0

Intuition: *the set of all tangent lines (lines under c, touching c at* x_0*)*

$$
\begin{aligned}
&\partial_{\boldsymbol{w}} \max\{0, 1 - y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b)\} \\
&= \partial_{\boldsymbol{w}} \begin{cases} 0 & \text{if } y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 1 \\ y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) & \text{otherwise} \end{cases} \\
&= \begin{cases} \partial_{\boldsymbol{w}} 0 & \text{if } y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 1 \\ \partial_{\boldsymbol{w}} y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) & \text{otherwise} \end{cases} \\
&= \begin{cases} \boldsymbol{0} & \text{if } y_n(\boldsymbol{w} \cdot \boldsymbol{x}_n + b) > 1 \\ y_n \boldsymbol{x}_n & \text{otherwise} \end{cases}
\end{aligned}
$$

Image from CIML