ANNOUNCEMENTS

- MIDTERM: 03/13, wednesday

  material:
  - Decision Trees
  - KNNs
  - Perceptron
  - Practical issues (overfitting, ...)
  - Reductions
  - Linear models (but not kernel methods)
    - in other words, today and wednesday's lecture, but not 03/11's lecture
  - Because I'm behind, <u>no A3b</u> (study for the midterm!)

- Next lecture:

    - Gradient Descent
    - Reading: <u>Why momentum really works</u>, up to Polynomial Regression

LINEAR MODELS

We have so far learned about the perceptron, decision trees, and K-NN. We studied algorithms to train those models, and how to assess their performance.

But their training algorithms are kind of heuristic: "do this and hope it works well."

Here's a different idea: write a formula for how bad a model is, then "training" means "find the model that optimizes that formula."

Linear models all work by having the training procedure pick one hyperplane and the formula always goes through the margin that training examples attain.

(We assume all input points have a column with value 1, so no bias term is needed)

$$\text{Loss-of-model}(w) = \sum_{(x,y)} \text{loss-of-sample}(\text{margin}(x,w), y)$$

WHICH LOSS TO CHOOSE?

Misclassification loss: $\ell(m, y) \begin{cases} 0, & \text{if } \operatorname{sign}(m) = y \\ 1, & \text{otherwise} \end{cases}$

This is the obvious loss, but we rarely use it, because it leads to an NP-Hard optimization problem.

To explain what causes the trouble, let's refactor our loss definition, so that the loss takes a single parameter, which is "positive on the correct side of the margin".

Our margin calculation then needs to know about the label:

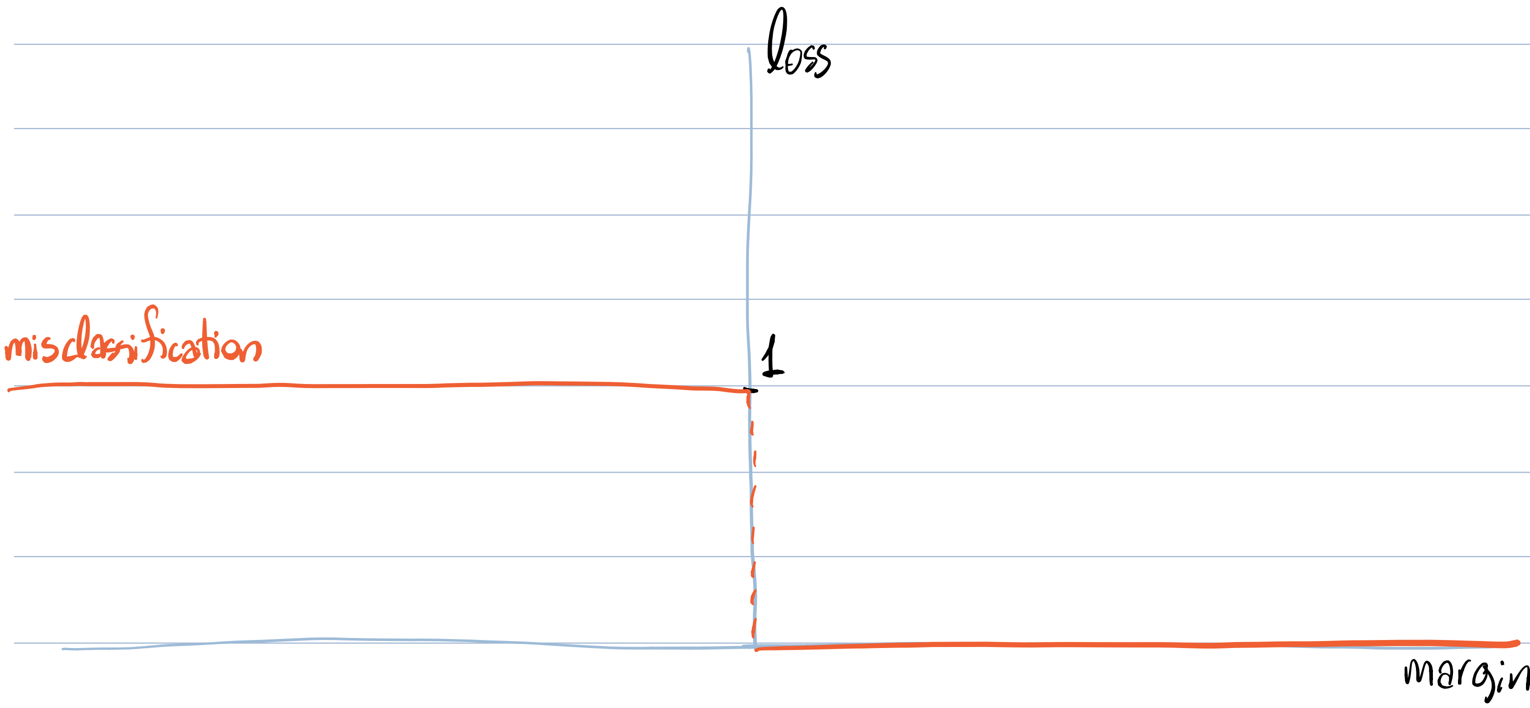
$$\text{margin}(w, x, y) = \langle w, x \rangle y$$

And our misclassification loss is now

$$\ell_0(m) = \begin{cases} 1, & \text{if } m < 0 \\ 0, & \text{otherwise} \end{cases}$$

CONVEX LOSS SURROGATES

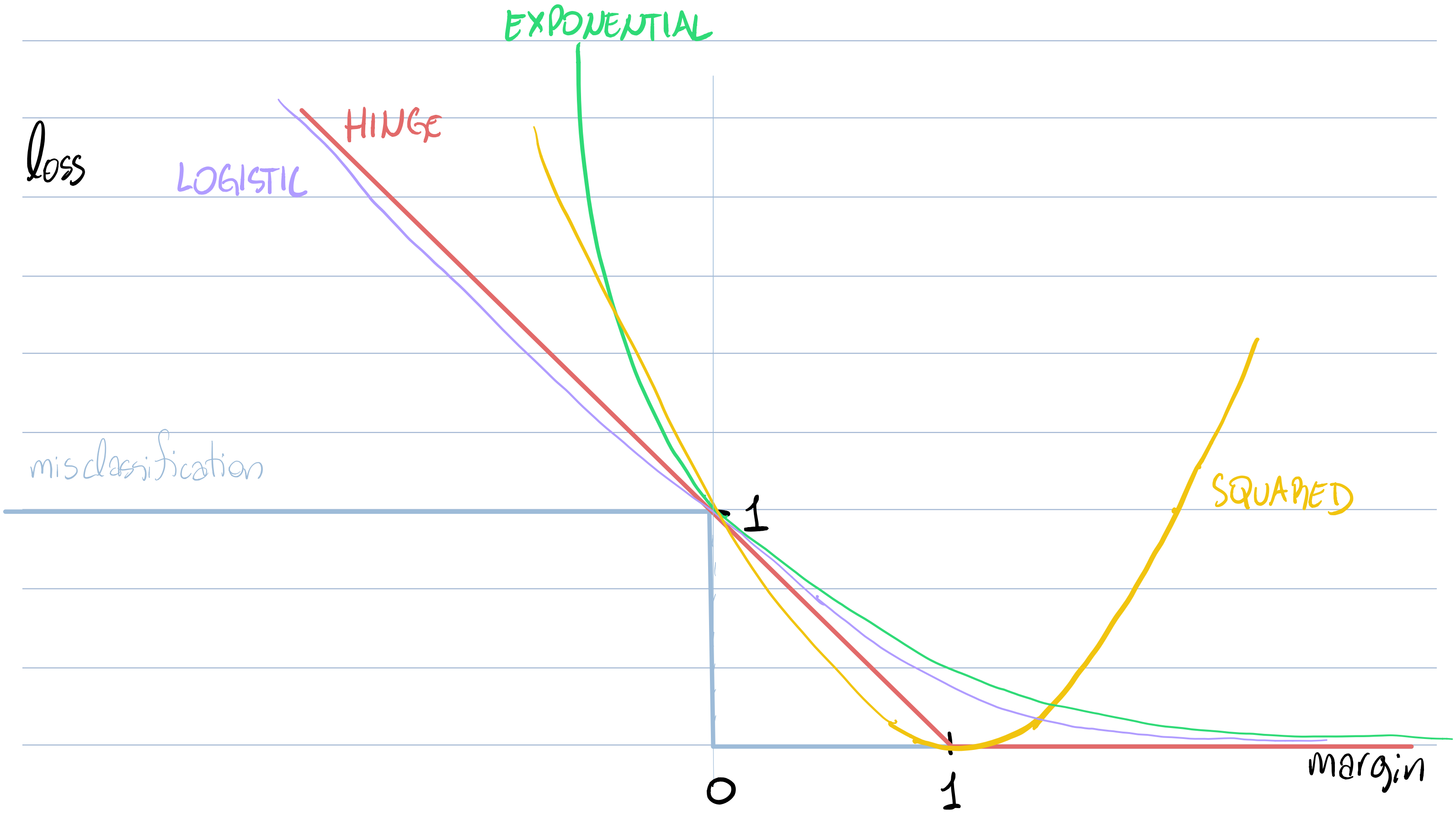
The trouble with misclassification loss is now graphically clearer. This loss is not convex:



Non-convex optimization is hard to do well. (We will see it later in the course: neural networks are the canonical modern non-convex, non-linear models)

We need losses that are similar to misclassification so that they are useful, but that are also convex (and differentiable) so that they are practical.

THE CONVEX LOSS ZOO



$$d = 1 - y$$

$$\ell_Q(y) = (1-y)^2 \qquad \ell_Q(d) = d^2$$

$$\ell_E(y) = \exp(-y) \qquad \ell_E(d) = \exp(1+d)$$

$$\ell_H(y) = \max(0, 1-y) \qquad \ell_H(d) = \max(0, d)$$

$$\ell_L(y) = \frac{1}{\log 2} \log(1 + \exp(-y)) \qquad \ell_L(d) \propto \log(1 + \ell_E(d))$$

QUADRATIC: Keep it above zero and easy to get closed form

EXPONENTIAL: Keep it monotonic

HINGE: Keep it monotonic, compact support on "good" side

LOGISTIC: Keep it monotonic, smooth, "not too angry"

EMPIRICAL RISK VS. STRUCTURAL RISK

What happens with the best model we found on training data? On future data?

Our loss needs to prevent overfitting. One way to think about the potential for overfitting is by considering how "jiggly" our model is w.r.t. the training data.

Let's control how flexible our models can be:

$$\text{Loss-of-model}(w) = \sum_{(x,y)} \text{loss-of-sample}(y, \text{margin}(x,w,y))$$

$$+ \lambda \cdot \text{model-complexity}(w)$$

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

If $w = (0, \dots, 0)$, how flexible is that model?
What if $w = (10^{100}, \dots, 10^{100})$? Use norm of vector as proxy for complexity!

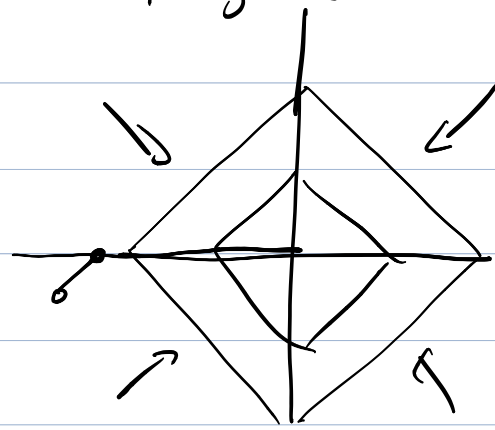
THE REGULARIZATION ZOO

$$\nabla_w \|w\|^2 = 2w$$

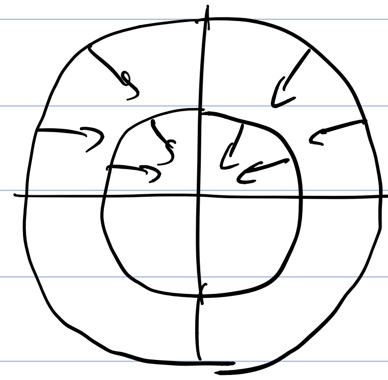
l_0 regularization

$$l_0(w) = \sum_i [w_i \neq 0]$$

l_1 regularization

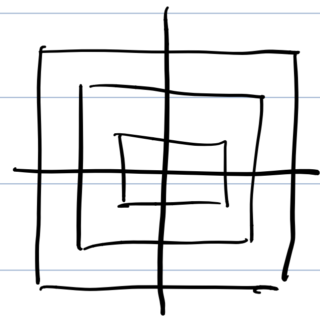


l_2 regularization



l_3 regularization

l_∞ regularization



$$\|w\|_p = \sqrt[p]{\sum |w_i|^p}$$

EXTRAS