CPSC 340:
Machine Learning and Data Mining

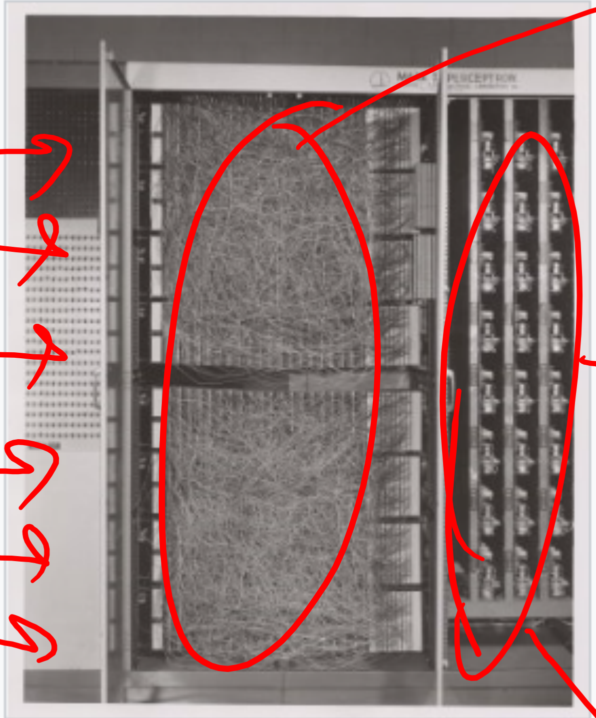
L1 regularization & linear Classifiers

Bonus slides

$$z_i = (x_i^2 \quad x_1 x_2 \quad x_3)$$

History [edit]

x_i



y_i

w

The Mark I Perceptron machine was the first implementation of the perceptron algorithm. The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image. The main visible feature is a patchboard that allowed experimentation with different combinations of input features. To the right of that are arrays of potentiometers that implemented the adaptive weights.[2]:213

https://en.wikipedia.org/wiki/Perceptron

L1-Regularization as a Feature Selection Method

- Advantages:
  - Deals with conditional independence (if linear).
  - Sort of deals with collinearity:
    - Picks at least one of “mom” and “mom2”.
  - Very fast with specialized algorithms.
- Disadvantages:
  - Tends to give false positives (selects too many variables).
- Neither good nor bad:
  - Does not take small effects.
  - Says “gender” is relevant if we know “baby”.
  - Good for prediction if we want fast training and don’t care about having some irrelevant variables included.

"Elastic Net": L2- and L1-Regularization

- To address non-uniqueness, some authors use L2- and L1-:

$$f(w) = \frac{1}{2}\|Xw - y\|^2 + \frac{\lambda_2}{2}\|w\|^2 + \lambda_1\|w\|_1$$

- Called "elastic net" regularization.
  - Solution is sparse and unique.
  - Slightly better with feature dependence:
    - Selects both "mom" and "mom2".

- Optimization is easier though still non-differentiable.

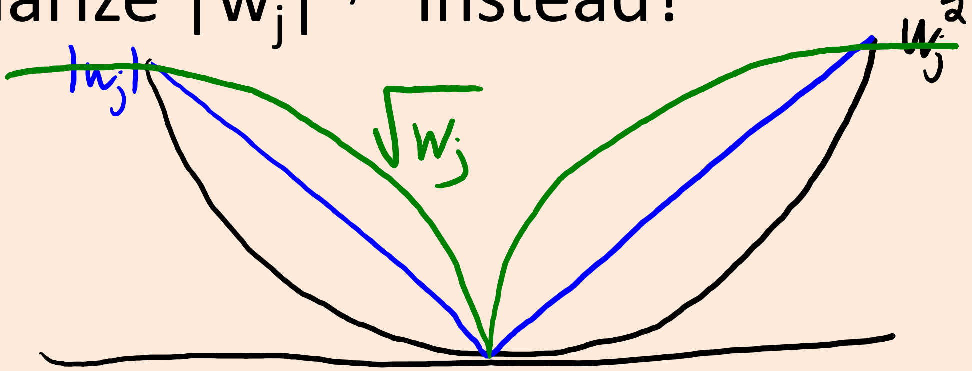
L1-Regularization Debiasing and Filtering

- To remove false positives, some authors add a debiasing step:
  - Fit ‘w’ using L1-regularization.
  - Grab the non-zero values of ‘w’ as the “relevant” variables.
  - Re-fit relevant ‘w’ using least squares or L2-regularized least squares.

- A related use of L1-regularization is as a filtering method:
  - Fit ‘w’ using L1-regularization.
  - Grab the non-zero values of ‘w’ as the “relevant” variables.
  - Run standard (slow) variable selection restricted to relevant variables.
    - Forward selection, exhaustive search, stochastic local search, etc.

Non-Convex Regularizers

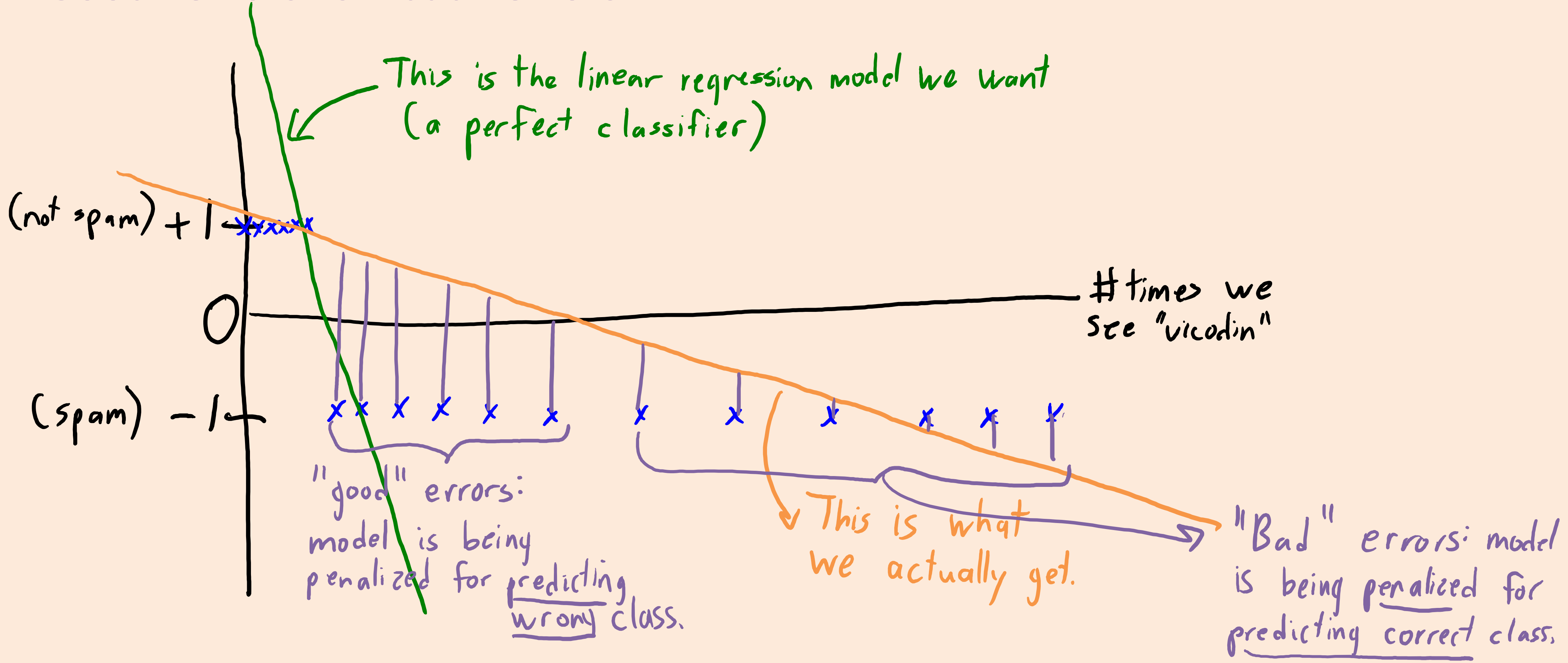
- Regularizing $|w_j|^2$ selects all features.
- Regularizing $|w_j|$ selects fewer, but still has many false positives.
- What if we regularize $|w_j|^{1/2}$ instead?



- Minimizing this objective would lead to fewer false positives.
  – Less need for debiasing, but it's not convex and hard to minimize.
- There are many non-convex regularizers with similar properties.
  – L1-regularization is (basically) the "most sparse" convex regularizer.

Can we just use least squares??

- What went wrong?
  - "Good" errors vs. "bad" errors.

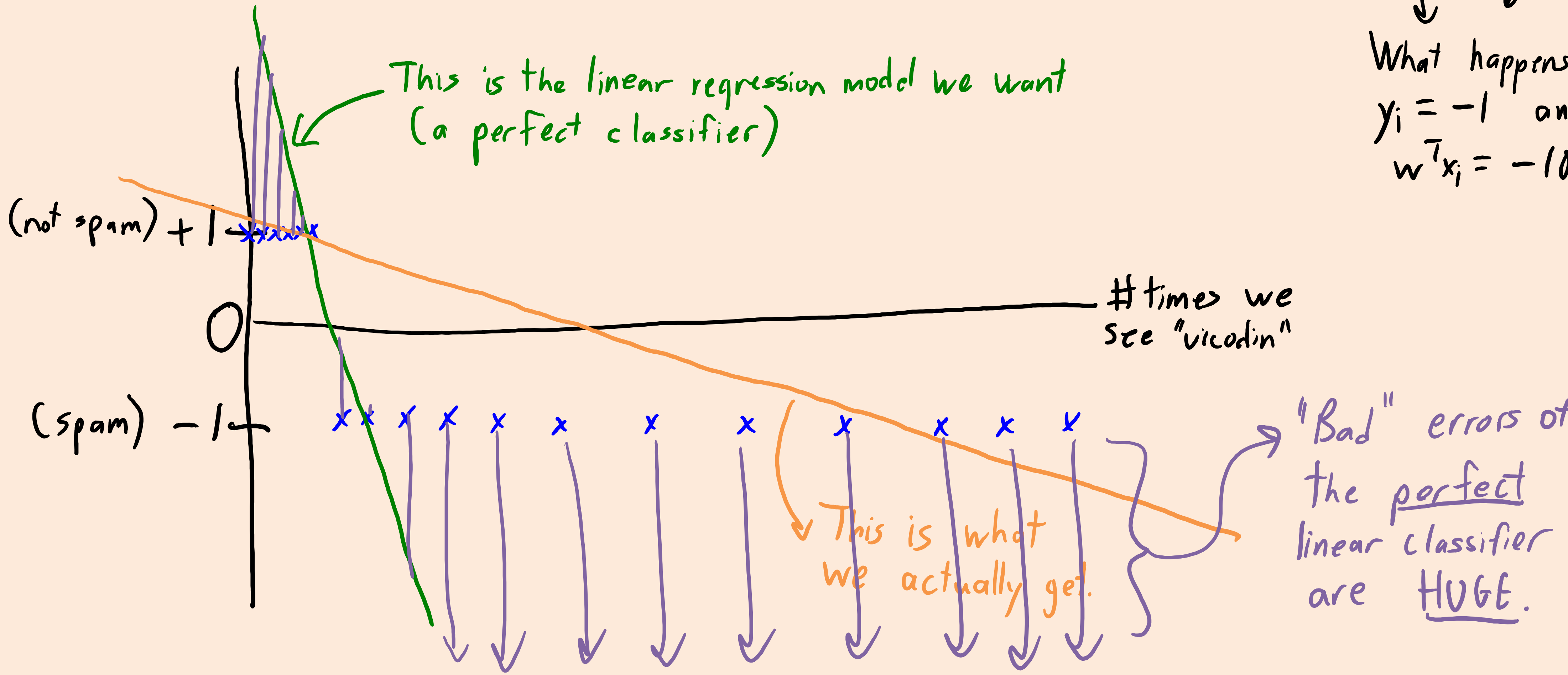


Can we just use least squares??

- What went wrong?
  - "Good" errors vs. "bad" errors.

$$f(w) = \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

What happens if $y_i = -1$ and $w^T x_i = -1000$?



Online Classification with Perceptron

- Perceptron for online linear binary classification [Rosenblatt, 1957]
  - Start with $w_0 = 0$.
  - At time 't' we receive features x_t.
  - We predict $\hat{y}_t = \text{sign}(w_t^T x_t)$.
  - If $\hat{y}_t \neq y_t$, then set $w_{t+1} = w_t + y_t x_t$.
    - Otherwise, set $w_{t+1} = w_t$.

(Slides are old so above I'm using subscripts of 't' instead of superscripts.)

- Perceptron mistake bound [Novikoff, 1962]:
  - Assume data is linearly-separable with a "margin":
    - There exists w^* with $||w^*||=1$ such that $\text{sign}(x_t^T w^*) = \text{sign}(y_t)$ for all 't' and $|x^T w^*| \geq \gamma$. >0
  - Then the number of total mistakes is bounded.
    - No requirement that data is IID.

Perceptron Mistake Bound

- Let's normalize each x_t so that $||x_t|| = 1$.
  - Length doesn't change label.
- Whenever we make a mistake, we have $\text{sign}(y_t) \neq \text{sign}(w_t^T x_t)$ and

$$
\begin{aligned}
\|w_{t+1}\|^2 &= \|w_t + y x_t\|^2 \\
&= \|w_t\|^2 + 2 \underbrace{y_t w_t^T x_t}_{<0} + 1 \\
&\leq \|w_t\|^2 + 1 \\
&\leq \|w_{t-1}\|^2 + 2 \\
&\leq \|w_{t-2}\|^2 + 3.
\end{aligned}
$$

- So after 'k' errors we have $||w_t||^2 \leq k$.

Perceptron Mistake Bound

- Let's consider a solution w*, so $\text{sign}(y_t) = \text{sign}(x_t^T w^*)$.
  - And let's choose a w* with $||w^*|| = 1$,
- Whenever we make a mistake, we have:

$$
\begin{aligned}
\|w_{t+1}\| &= \|w_{t+1}\|\|w_*\| \\
&\geq w_{t+1}^T w_* \\
&= (w_t + y_t x_t)^T w_* \\
&= w_t^T w_* + y_t x_t^T w_* \\
&= w_t^T w_* + |x_t^T w_*| \\
&\geq w_t^T w_* + \gamma.
\end{aligned}
$$

  - Note: $w_t^T w_* \geq 0$ by induction (starts at 0, then at least as big as old value plus γ).
- So after 'k' mistakes we have $||w_t|| \geq \gamma k$.

Perceptron Mistake Bound

- So our two bounds are $||w_t|| \leq \text{sqrt}(k)$ and $||w_t|| \geq \gamma k$.

- This gives $\gamma k \leq \text{sqrt}(k)$, or a maximum of $1/\gamma^2$ mistakes.
  - Note that $\gamma > 0$ by assumption and is upper-bounded by one by $||x|| \leq 1$.
  - After this ‘k’, under our assumptions
    we’re guaranteed to have a perfect classifier.