CPSC 340: Machine Learning and Data Mining

Linear Classifiers: predictions

Original version of these slides by Mark Schmidt, with modifications by Mike Gelbart.

Admin

- Assignment 4:
  - Due Friday of next week

- Midterm:
  - Well done!
  - You have until Thursday to discuss grading concerns (in person)
    - See Piazza for post regarding Q4a
  - You can collect exam papers after class or in office hours

Part 3 Key Ideas: Linear Models, Least Squares

- Focus of Part 3 is linear models:
  - Supervised learning where prediction is linear combination of features:

$$\hat{y}_i = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id}$$
$$= w^T x_i$$

- Regression:
  - Target y_i is numerical, testing ($\hat{y}_i == y_i$) doesn't make sense.

Good fit that doesn't exactly pass through any point.

- Squared error: $\frac{1}{2}\sum_{i=1}^{n}(w^T x_i - y_i)^2$ or $\frac{1}{2}\|Xw - y\|^2$
  - Can find optimal 'w' by solving "normal equations".

Part 3 Key Ideas: Gradient Descent, Error Functions

- For large 'd' we often use gradient descent:
  - Iterations only cost O(nd).
  - Converges to a critical point of a smooth function.
  - For convex functions, it finds a global optimum.

- L_1-norm and L_∞-norm errors:

$$\|Xw - y\|_1 \qquad \|Xw - y\|_\infty$$

  - More/less robust to outliers.
  - Can apply gradient descent after smoothing with Huber or log-sum-exp.

Part 3 Key Ideas: Change of basis, Complexity Scores

- Change of basis: replaces features x_i with non-linear transforms z_i:
  - Add a bias variable (feature that is always one).
  - Polynomial basis.
  - Radial basis functions (non-parametric basis).

- We discussed scores for choosing “true” model complexity.
  - Validation score vs. AIC/BIC.

- Search and score for feature selection:
  - Define a “score” like BIC, and do a “search” like forward selection.

Part 3 Key Ideas: Regularization

- L0-regularization (AIC, BIC):
  - Adds penalty on the number of non-zeros to select features.

$$f(w) = \|Xw - y\|^2 + \lambda\|w\|_0$$

- L2-regularization (ridge regression):
  - Adding penalty on the L2-norm of ‘w’ to decrease overfitting:

$$f(w) = \|Xw - y\|^2 + \frac{\lambda}{2}\|w\|^2$$

- L1-regularization (LASSO):
  - Adding penalty on the L1-norm decreases overfitting and selects features:

$$f(w) = \|Xw - y\|^2 + \lambda\|w\|_1$$

Key Idea in Rest of Part 3

- The next few lectures will focus on:
  - Using linear models for classification

- It may seem like we're spending a lot of time on linear models.
  - Linear models are used a lot and are understandable.
    - ICBC only uses linear models for insurance estimates.
  - Linear models are also the building blocks for more-advanced methods.
    - "Latent-factor" models in Part 4 and "deep learning" in Part 5.

Motivation: Identifying Important E-mails

- How can we automatically identify 'important' e-mails?

- A binary classification problem ("important" vs. "not important").
  - Labels are approximated by whether you took an "action" based on mail.
  - High-dimensional feature set (that we'll discuss later).

- Gmail uses a linear classifier for this problem.

Binary Classification Using Regression?

- Can we apply linear models for binary classification?
  - Set $y_i = +1$ for one class ("important").
  - Set $y_i = -1$ for the other class ("not important").
- At training time, fit a linear regression model:

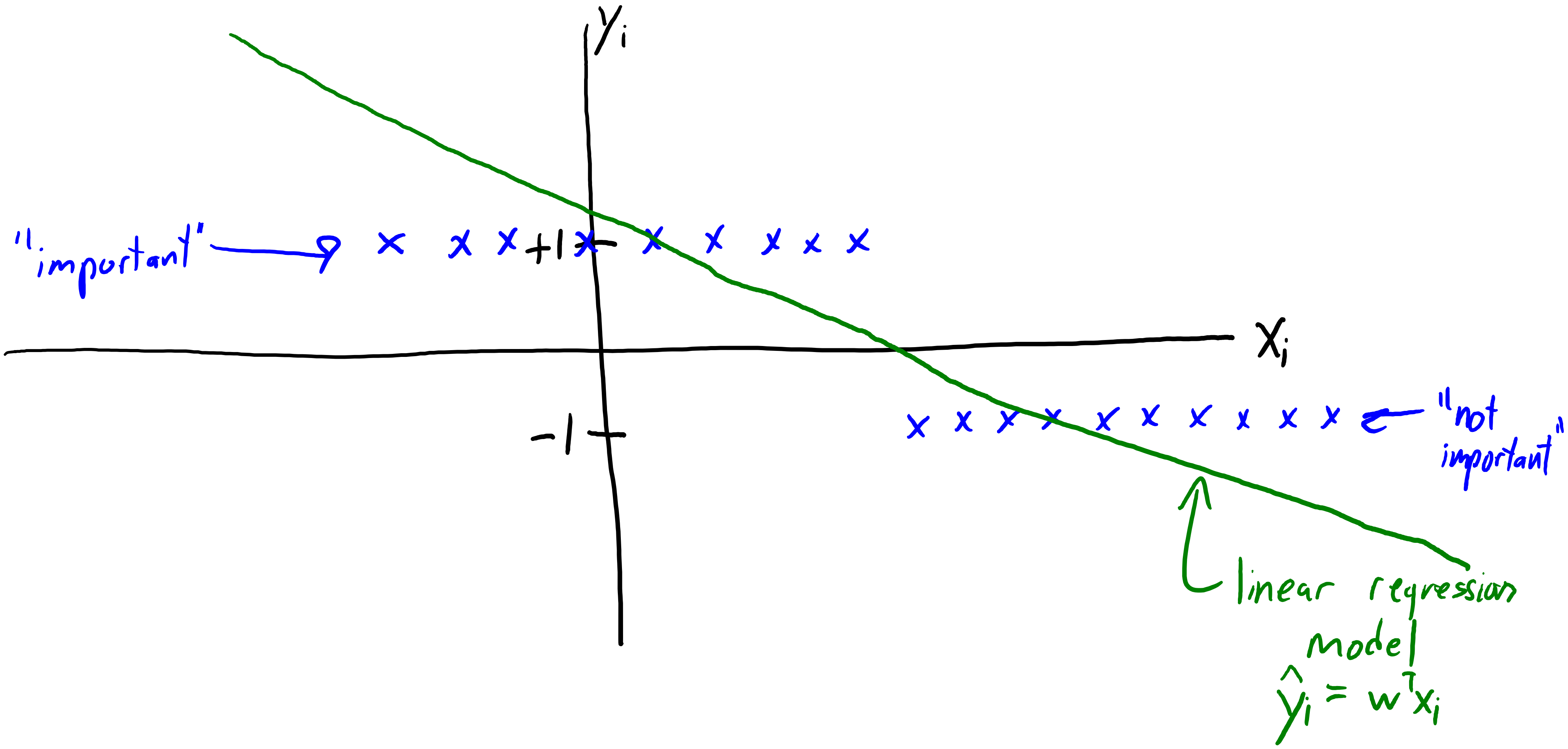
$$\hat{y}_i = w_1 x_{i1} + w_2 x_{i2} + \cdots + w_d x_{id}$$
$$= w^T x_i$$

- The model will try to make $w^T x_i = +1$ for "important" e-mails, and $w^T x_i = -1$ for "not important" e-mails.

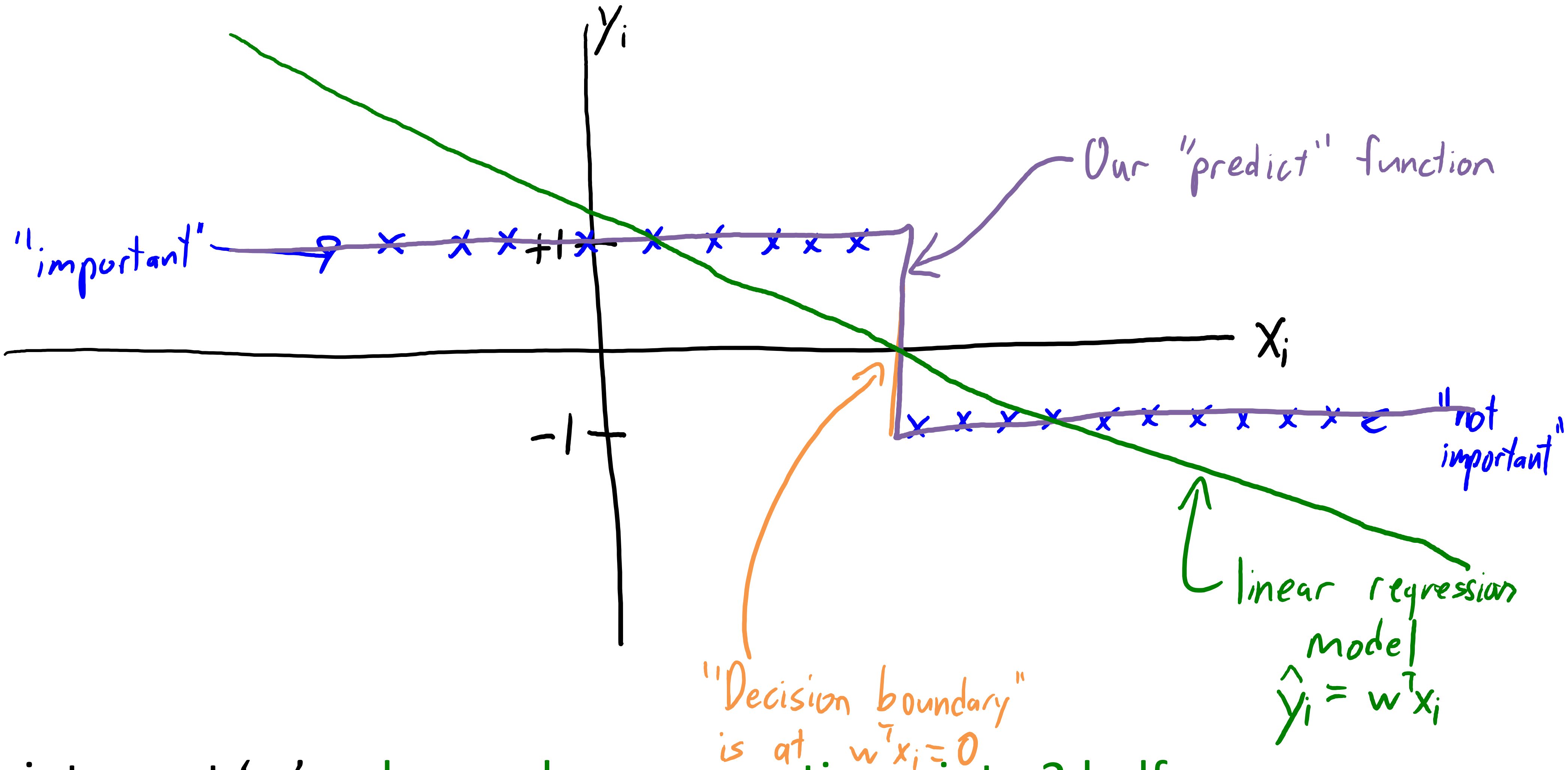
Binary Classification Using Regression?

- Can we apply linear models for binary classification?
  - Set $y_i = +1$ for one class ("important").
  - Set $y_i = -1$ for the other class ("not important").
- Linear model gives real numbers like 0.9, -1.1, and so on.
- So to predict, we look at the sign of $w^T x_i$.
  - If $w^T x_i = 0.9$, predict $\hat{y}_i = +1$.
  - If $w^T x_i = -1.1$, predict $\hat{y}_i = -1$.
  - If $w^T x_i = 0.1$, predict $\hat{y}_i = +1$.
  - If $w^T x_i = -100$, predict $\hat{y}_i = -1$.

Decision Boundary in 1D



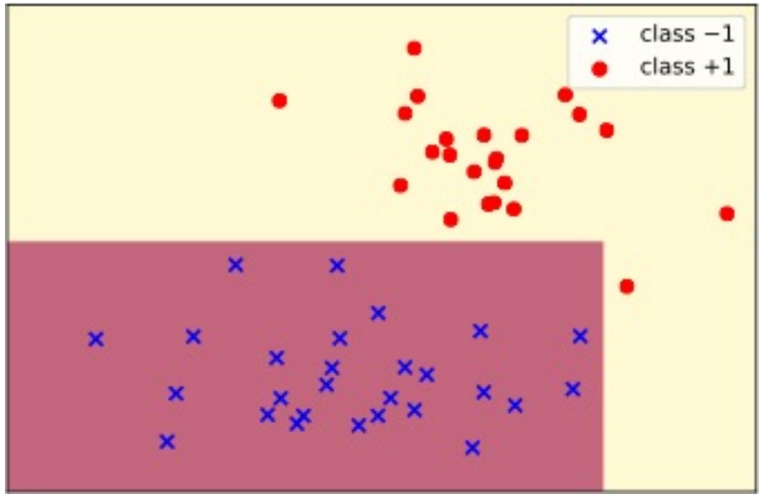
Decision Boundary in 1D



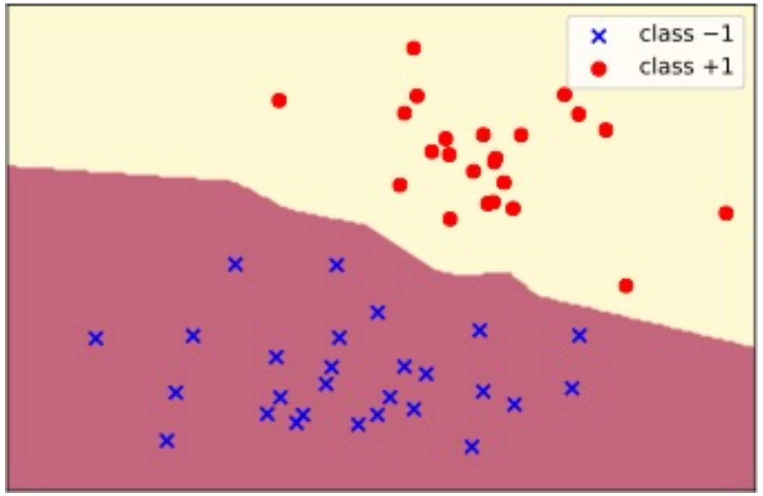
- We can interpret 'w' as hyperplane separating x into 2 half-spaces:
  - Half-space where $w^Tx_i > 0$ and half-space where $w^Tx_i < 0$.

Decision Boundary in 2D

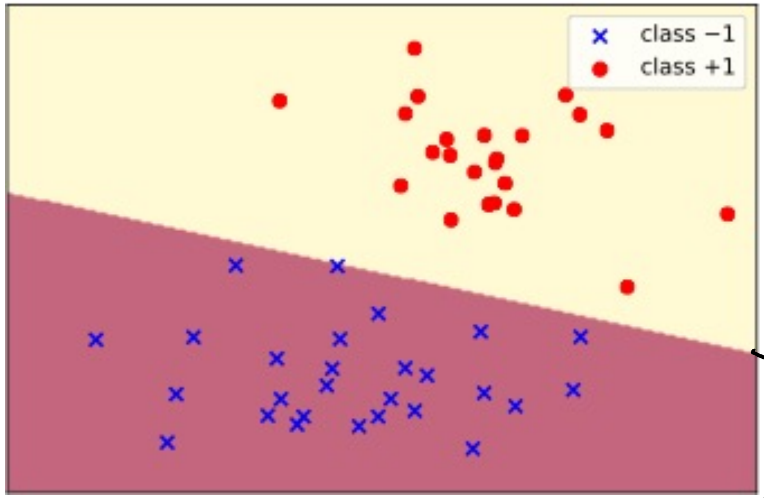
decision tree



KNN



linear classifier



- A linear classifier would be linear function $\hat{y}_i = \beta + w_1 x_{i1} + w_2 x_{i2}$ coming out of the page (the boundary is at $\hat{y}_i = 0$).
- Or recall from multivariable calculus that a plane in d-dimensions is defined by its normal vector in d-dimensions, plus an intercept/offset.

$\hat{y}_i = w^T x_i = 0$

Perceptron Algorithm

- One of the first "learning" algorithms was the "perceptron" (1957).
  - Searches for a 'w' such that $\text{sign}(w^T x_i) = y_i$ for all i.

- Perceptron algorithm:
  - Start with $w^0 = 0$.
  - Go through examples in any order until you make a mistake predicting y_i.
    - Set $w^{t+1} = w^t + y_i x_i$.
  - Keep going through examples until you make no errors on training data.

- Intuition for step: if $y_i = +1$, "add more of x_i to w" so that $w^T x_i$ is larger.

$$(w^{t+1})^T x_i = (w^t + x_i)^T x_i = (w^t)^T x_i + x_i^T x_i = (\text{old prediction}) + \|x_i\|^2$$

- If a perfect classifier exists, this algorithm finds one in finite number of steps.
  - In this case we say the training data is "linearly separable"

Lecture continues in Jupyter notebook…

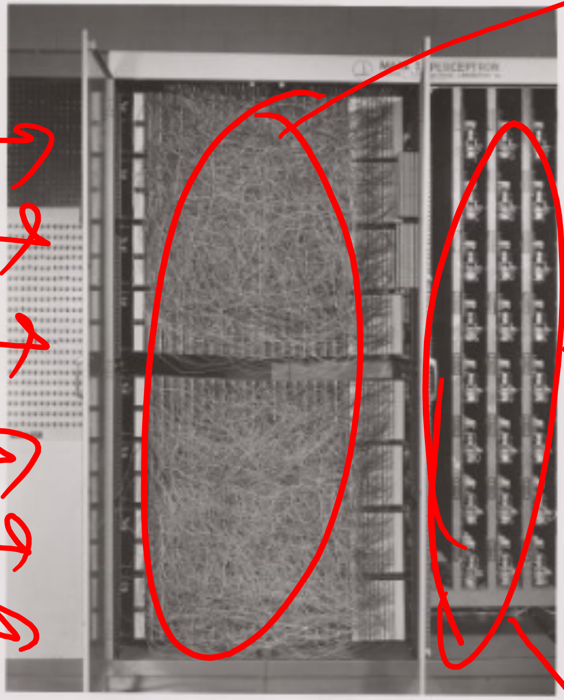
Summary

- Binary classification using regression:
  - Encode using y_i in {-1,1}.
  - Use $\text{sign}(w^T x_i)$ as prediction.
  - "Linear classifier" (a hyperplane splitting the space in half).
- Perceptron algorithm: finds a perfect classifier (if one exists).
- Least squares is a weird error for classification.

History [edit]

$Z_i = (x_1^2 \;\; x_1 x_2 \;\; x_3)$

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

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 $\text{sign}(y_t) = \text{sign}(x_t^T w^*)$.
- Whenever we make a mistake, we have:

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

- 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.