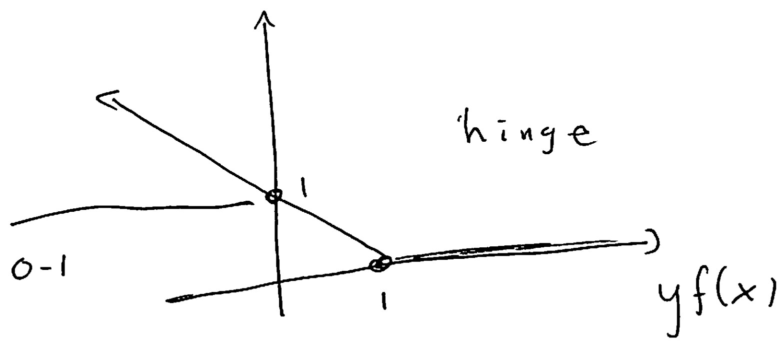
Risk minimization view of classification

$$\text{loss}_{01}(f(x), y) = \begin{cases} 0 & yf(x) > 0 \\ 1 & \text{otherwise} \end{cases}$$

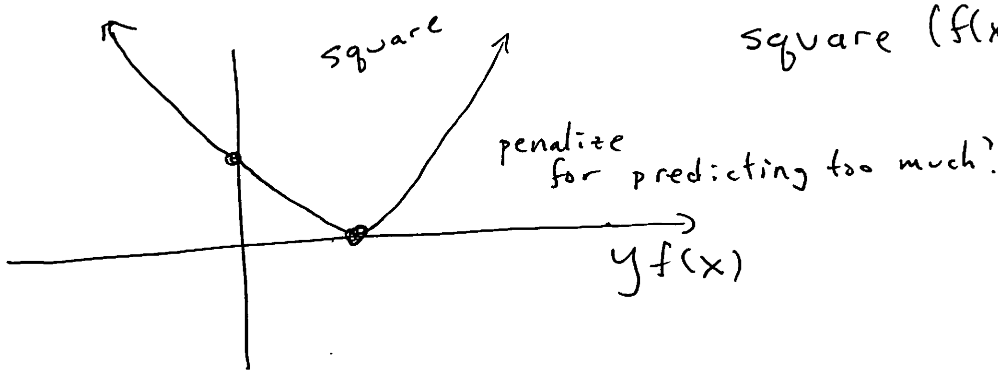
This loss is not convex.

Surrogate losses

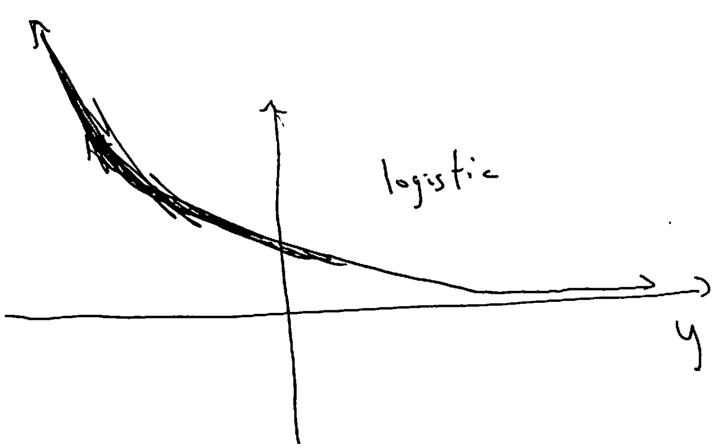
$$\text{hinge}(f(x), y) = \max(1 - yf(x), 0)$$



$$\text{square}(f(x), y) = (f(x) - y)^2 = (1 - f(x)y)^2$$



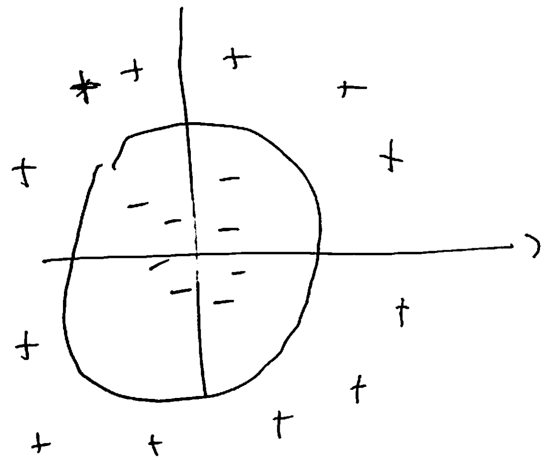
$$\text{logistic}(f(x), y) = \log\left(1 + e^{-yf(x)}\right)$$



Nonlinear classification

Simplest approach: add more features

- polynomials
- histograms
- kernels



$$x \to \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}$$

Just as with regression, apply nonlinear map to $\vec{x}_i$ and learn on output:

$$\varphi : \mathcal{X} \to \mathbb{R}^D$$

$$\hat{f}(\vec{x}) = \vec{w}^T \varphi(\vec{x})$$

$$\underset{\vec{w}}{\text{minimize}} \quad \sum_{i=1}^{n} \text{loss}\left(\vec{w}^T \varphi(\vec{x}_i), y_i \right)$$

minimize $\quad J(\vec{w}) \equiv \frac{1}{n} \sum_{i=1}^{n} \text{loss}(\vec{w}^T x_i, y_i) + \frac{\gamma}{2}\|\vec{w}\|^2$

GRADIENT:

$$\nabla J(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} \left(\left. \frac{\partial\, \text{loss}(z, y_i)}{\partial z} \right|_{z = \vec{w}^T \vec{x}_i} \right) \cdot \vec{x}_i + \gamma \vec{w}$$

Stochastic Gradient: pick i at random and set

$$G_i(\vec{w}) = \left. \frac{\partial\, \text{loss}}{\partial z} \right|_{z = \vec{w}^T x_i} \vec{x}_i + \gamma \vec{w}$$

(no summation, one example)

stochastic gradient descent

$$\vec{w}_{k+1} = \vec{w}_k - \alpha\, G_{i_k}(\vec{w}_k)$$

Note:
- $G_{i_k}(\vec{w}_k) = 0$ does not mean $\nabla J(\vec{w}_k) = 0$
- This algorithm might not even decrease the function value.

Why should this work?

Example: least-squares

$$J(\vec{w}) = \frac{1}{2n} \sum_{i=1}^{n} (w - y_i)^2 \qquad w \in \mathbb{R}$$

S.G: $\quad G_i(w) = w - y_i, \qquad \alpha_k = \frac{1}{k}$

Init: $\quad w_1 = 0$

$$w_2 = w_1 - w_1 + y_1 = y_1$$

$$w_3 = w_2 - \tfrac{1}{2}(w_2 - y_2) = \tfrac{1}{2} y_1 + \tfrac{1}{2} y_2$$

$$w_4 = w_3 - \tfrac{1}{3}(w_3 - y_3) = \tfrac{1}{3}(y_1 + y_2 + y_3)$$

$$w_n = \frac{1}{n} \sum_{i=1}^{n} y_i \longrightarrow \text{the minimizer!}$$

$$\nabla J(\vec{w}_n) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{1}{n} \sum_{i=1}^{n} y_i - y_i \right)$$

$$= \frac{1}{n} \sum y_i - \frac{1}{n} \sum y_i = 0$$

So SGD finds the minimum

Why "stochastic?"

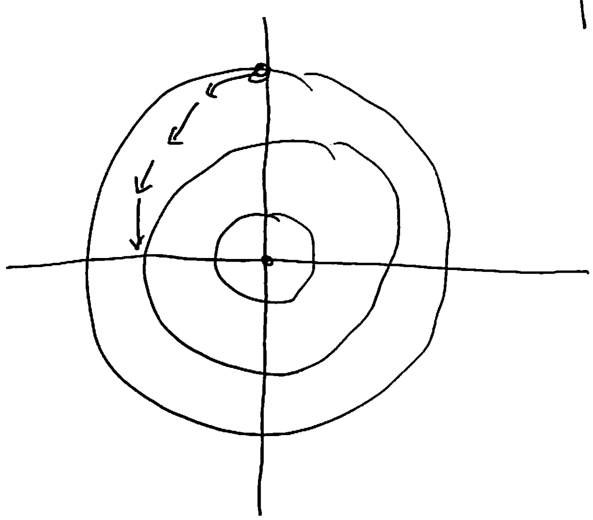
$$J(\vec{w}) = \frac{1}{20} \sum_{k=0}^{9} \left(\cos\left(\frac{\pi k}{10}\right) w_1 + \sin\left(\frac{\pi k}{10}\right) w_2 \right)^2$$

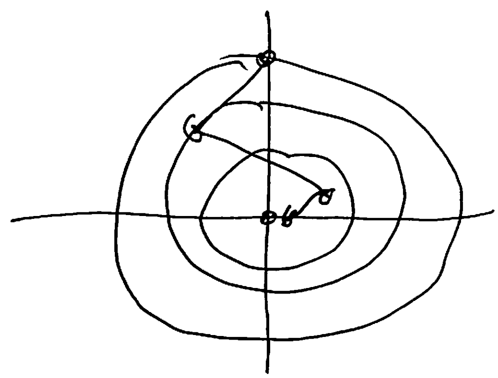
optimum: $\vec{w} = 0$

stepsize $\alpha = 1$

$$\vec{w} - G_i(\vec{w}) = \frac{1}{2} \begin{bmatrix} 1 - \cos\left(\frac{\pi k}{5}\right) & \sin\left(\frac{\pi k}{5}\right) \\ \sin\left(\frac{\pi k}{5}\right) & 1 - \cos\left(\frac{\pi k}{5}\right) \end{bmatrix} \vec{w}$$

run in order $\boxed{\vec{w}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}}$ random order





Noisy Gradient descent:

Let ν_i be an iid noise process

And run $\vec{w}_{k+1} = \vec{w}_k - \alpha \nabla J(\vec{w}_k) + \alpha \nu_k$

If $\mathbb{E}[\nu_i] = 0$ and ν_k is independent of w_k

$$\mathbb{E}[\alpha \nabla J(\vec{w}_k) + \alpha \nu_k] = \alpha \, \mathbb{E}[\nabla J(w_k)]$$

so direction is correct in expectation.

Example: $\text{loss}(\vec{w}^T \vec{x}_i, y_i) = \max(1 - y_i \vec{w}^T \vec{x}_i, 0)$

$= \text{hinge}(\vec{w}^T \vec{x}_i, y_i)$

$$G_i(\vec{w}) = y_i e_i(\vec{w}) \, x_i + \gamma \vec{w}$$

$$e_i(\vec{w}) = \begin{cases} 1 & \text{if } y_i \vec{w}^T x_i \leq 1 \\ 0 & \text{o.w.} \end{cases}$$

Algorithm: Pick i at random

If $y_i \vec{w}_k^T \vec{x}_i \leq 1$:

$$\vec{w}_{k+1} = (1 - \beta_1) \vec{w}_k + \beta_2 y_i \vec{x}_i$$

else:

$$\vec{w}_{k+1} = (1 - \beta_1) \vec{w}_k$$

PERCEPTRON

more generally

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

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

solve $\nabla f(w) = 0 \longrightarrow w_\star = \frac{1}{n} \sum_{i=1}^{n} y_i$

here stochastic gradients push $\rightarrow$

here stochastic gradients push $\leftarrow$

region of confusion.

This explains behavior of SGD:

fast convergence to neighborhood of $w_\star$

Use step-size / learning rate decay to get to $w_\star$

Tricks of the trade:

Epochs: rather than choose i at random, shuffle the data and run in random order (can buy big speedups)

- reducing learning rate:
  rule of thumb. pick step size as big as possible so that you don't diverge. anneal using epoch doubling, or exponential decay.

- momentum

$$w_{k+1} = w_k - \alpha_k g(w_k) + \beta\,(\underbrace{w_k - w_{k-1}})$$

if previous direction was good, then continuing along that direction is good. Helps accelerate in "narrow" valleys.

$\beta = 0.9$, but tune it!