Examples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$

$x_i \in \mathbb{R}^d$, where d = # of features

$y_i \in \mathbb{R}$ is a label.

training data set

Goal: learn a model h, sometimes called a hypothesis, from some family $\mathcal{H}$ of possible models.

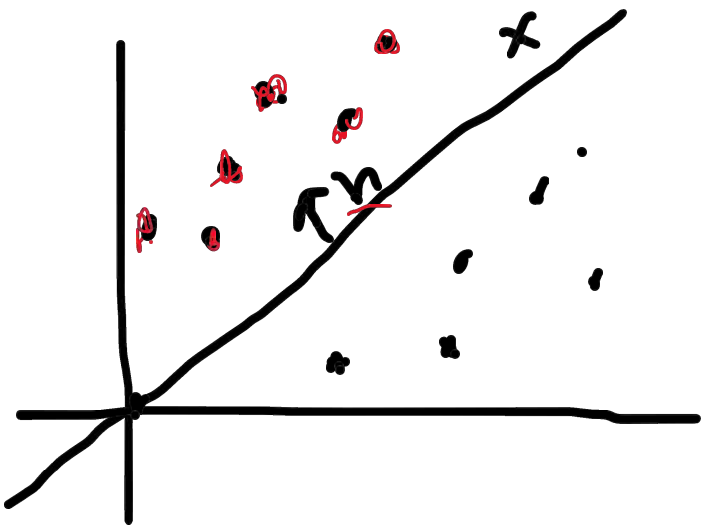
Given a new unlabeled point $x \in R^d$, use your model h to predict label y. This x is called test data.

How well h predicts y is called generalization error.

Linear Classification:

labels are 0 or 1

red black.



$\langle h, x \rangle$ is normal vector to line

Training data: $(x_1, 0), (x_2, 1), (x_3, 1), \ldots$

Want h to separate red and black points.

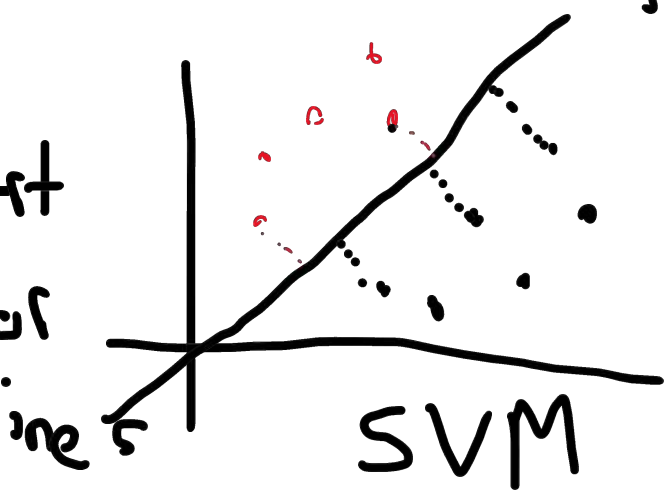
Want: $\langle x_i, h \rangle \geq 0$ for all red points x_i

$\langle x_i, h \rangle \leq 0$ for all black points x_i

Linear Classification

categorical - output is binary

with a margin γ



SVM

which line should I choose?

choose maximum margin line.

closest points called support vectors

Margin: want every point to be distance $\geq \gamma$ from the separating line. How to find h?

want $\langle x_i, h \rangle \geq \gamma$ for all red points

$\langle x_i, h \rangle \leq -\gamma$ for all black points

binary search for γ.

Numerical output - example is linear regression. least squares test X

Training data $(x_1, y_1), \ldots, (x_n, y_n)$ but now $y_i \in R$.

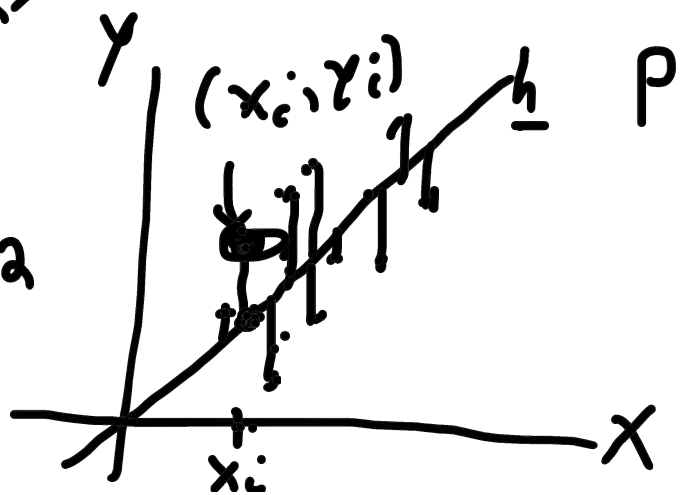
Model h is chosen to minimize loss function $L(h)$:

$$L(h) = \frac{1}{n} \sum_{i=1}^{n} (\langle x_i, h \rangle - y_i)^2$$

$\langle h, x \rangle$

x_1

$h_2 x_2$



$\langle x_i, h \rangle$ prediction, y_i label

$x_i \in R$ in example

(in general $x_i \in R^d$)

find a line to "fit" your data

How to solve regression with gradient descent

$$L(h) = \frac{1}{n} \sum_{i=1}^{n} (\langle x_i, h \rangle - y_i)^2$$ minimize a convex function

Initialize h^0 (do it randomly)

For $t = 1, 2, \ldots, T$

$$h^t \leftarrow h^{t-1} - \eta_t \nabla L|_{h^{t-1}}$$ ← what is the gradient?

End

Output $\frac{1}{T} \sum_{t=1}^{T} h^t$ (in practice just output h^T).

Need gradient of $L(h) = \frac{1}{n}\sum_{i=1}^{n} (\langle h, x_i \rangle - y_i)^2$.

Gradient is linear!! f_i

$$\nabla L = \frac{1}{n}\sum_{i=1}^{n} \nabla \underbrace{(\langle h, x_i \rangle - y_i)^2}$$

$$= \frac{1}{n}\sum_i \nabla f_i$$

$x_i \in \mathbb{R}^d$ $\quad \nabla(f+g) = \nabla f + \nabla g$

$h \in \mathbb{R}^d$

$$\nabla L = \left(\frac{\partial L}{\partial h_1}, \frac{\partial L}{\partial h_2}, \ldots, \frac{\partial L}{\partial h_d}\right)$$

$$\nabla f_i = \left(2(\langle h, x_i \rangle - y_i) x_{i1}, 2(\langle h, x_i \rangle - y_i) x_{i2}, \ldots\right)$$

$\nabla f_i|_{h^{t-1}}$ chain rule!

$$\nabla L|_{h^{t-1}} = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i|_{h^{t-1}}.$$

Chain Rule: $f(x) = u(v(x))$

$$\frac{\partial f}{\partial x} = \frac{\partial u}{\partial v} \cdot \frac{\partial v}{\partial x}$$

$v(h_1) =$

$\langle h_1, x_i \rangle - y_i$

$= \sum_{j=1}^{d} h_j \cdot x_{ij} - y_i$

$h_1 \cdot x_{i1}$

$u(v(h_1)) = v^2(h_1)$

$(2 \cdot v) \cdot x_{i1}$

$2(\langle h_1 x_i \rangle - y_i) x_{i1}$

$f_i = (\langle h_1, x_i \rangle - y_i)^2$

$\frac{\partial f_i}{\partial h_1}, \frac{\partial f_i}{\partial h_2}, \ldots$

slope / evolution

$y = mx$

Equation of li

line goes thrugh

x is 1-dimensi

Overfitting: you don't want to choose a "very weird" looking x to fit your data.

Occam's Razor: "simplest" solutions generalize better.

In regression, don't want h to have large norm.

Ridge regression: $L(h) = \frac{1}{n}\sum_{i=1}^{n}(\langle h, x_i\rangle - y_i)^2 + \lambda\|h\|^2$ (λ: adj, $\lambda > 0$)

Solve with gradient descent!

What if n is really large? Computing gradient is slow.

Stochastic Gradient Descent (SGD)

$$\nabla L = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i$$ slow because n is large

Randomly sample $I \in \{1, 2, \ldots, n\}$

$\widetilde{\nabla L} = \nabla f_I$ much faster - just one ∇f_I calculation!

$E[\widetilde{\nabla L}] = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i = \nabla L$ so unbiased estimator.

Reduce variance of SGD by using minibatches

Instead of sampling 1 example, we sample b examples. b = batch size. $I_1, I_2, \ldots, I_b$ are sampled examples

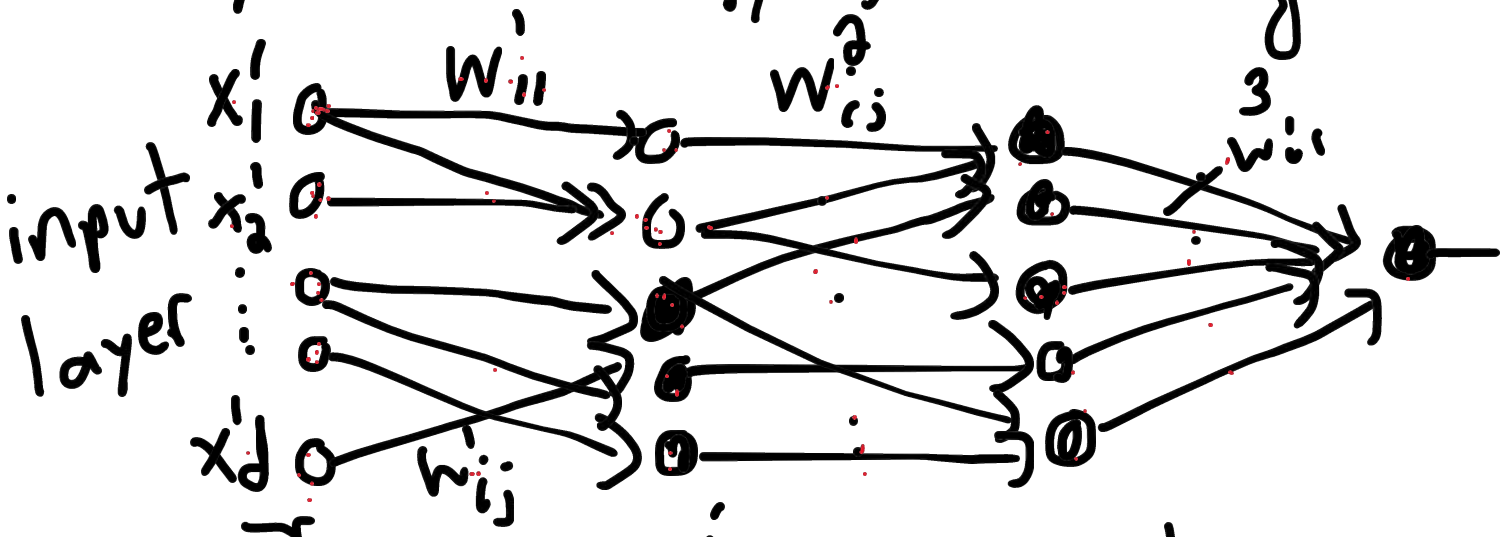
$$\widetilde{\nabla L} = \frac{1}{b} \sum_{j=1}^{b} \nabla f_{I_j}.$$

If $b = n$, get GD. If $b = 1$, get SGD.

b provides a time versus variance tradeoff.

Feedforward Neural Networks: $x^i \in R^d$

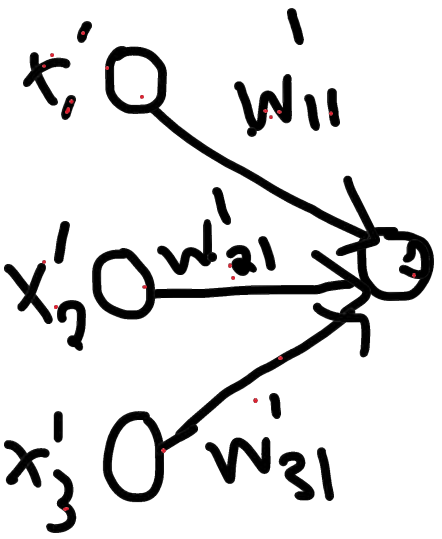
$(x^1, y^1), \ldots, (x^n, y^n)$ is training data. $y^i \in R$.



$w_{j,k}^i$ is weight connecting j-th node in i-th layer to k-th node in i+1 layer

Zoom in

Zoom In:



$$f(x_1' \cdot w_{11}' + x_2' w_{21}' + x_3' w_{31}')$$

f is activation function (non-linear)

How to learn weights?

back propagation

ReLU (rectified linear unit)

Sigmoid