

CSE 592 - Discussion 5

Supervised Learning, Linear Models & Generalization

Zhengjie (Jerry) Xu / Fall 26

Logistics

Homework 2 is due Wednesday, October 7 at 11:59 p.m.

Homework 3 will be out on October 7, focusing on machine learning.

Gibbs sampling in practice

Each step resamples one variable from $P(X \mid \text{Markov blanket})$.

Consecutive samples are correlated. Early samples depend on the start.

So we discard a burn-in period before counting.

Q: A and B agree 99.9% of the time. The chain is at $(A, B) = (T, T)$. Why is reaching (F, F) slow?

A: One variable changes per step, so the chain must pass through (T, F) or (F, T) .

Those states are very unlikely, so the chain rarely moves there: it mixes slowly.

Recap: supervised learning

Training data: examples (x, y) . Learn a predictor f , then predict y for new x .
Instead of modeling how x and y were generated, learn f directly.

Q: Which kind of task is each one?

Task	Output	Kind
Tomorrow's high temperature	a real number	
Which of 10 digits is in an image	one of 10 labels	
Is this transaction fraud?	yes / no	

Recap: three design decisions

Hypothesis class: which predictors f can we choose from?

Loss function: given the training data, how good is f ?

Optimization algorithm: how do we find the best f in the class?

The first two define the problem. The third is how we solve it.

Same split as before: search problem vs. A^* , CSP vs. backtracking,

Bayes net vs. variable elimination, and now loss vs. gradient descent.

Recap: linear regression

Features $\phi(x) = [1, x]$. Weights w . Prediction $f_w(x) = w \cdot \phi(x)$.

Residual: $f_w(x) - y$. Squared loss: $(f_w(x) - y)^2$.

$\text{TrainLoss}(w) = \text{average of } (w \cdot \phi(x) - y)^2 \text{ over the training set}$

Why squares? Least squares = maximum likelihood if $y = w \cdot \phi(x) + \text{Gaussian noise}$.

Laplace noise instead gives absolute error $|f_w(x) - y|$.

Training loss by hand

Training data: $(0, 1)$, $(1, 2)$, $(2, 6)$.

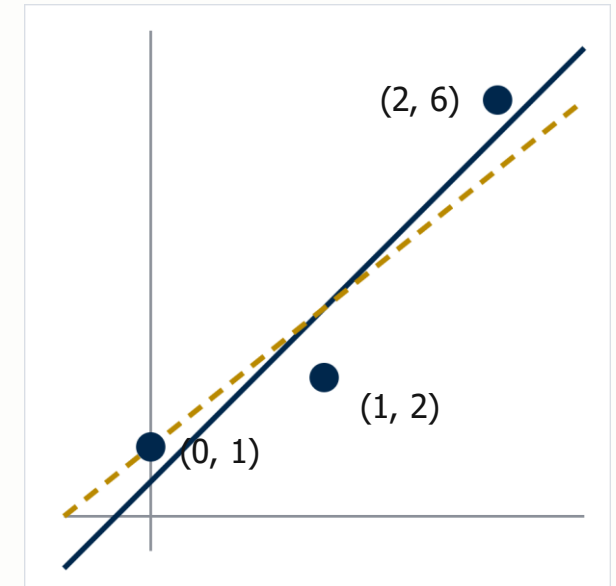
$f(x) = 1 + 2x$: residuals $0, 1, -1$. TrainLoss = $2/3 \approx 0.67$

$f(x) = 0.5 + 2.5x$: residuals $-0.5, 1, -0.5$. TrainLoss = 0.5

Q: The first line fits $(0, 1)$ exactly. Why is the second line better?

A: The loss averages squared errors. Two small misses beat one miss of size 1.

$0.5 + 2.5x$ is the least-squares line: no line has lower training loss.



solid: $0.5 + 2.5x$ · dashed: $1 + 2x$

Squared vs. absolute loss

Add one bad measurement: (3, 60), when the trend predicts about 8.

Q: A line misses this point by about 50. How much does it add to each loss?

A: Squared: about $50^2 = 2,500$. Absolute: about 50.

Under squared loss, the line tilts a lot to reduce that one error.

Absolute loss treats it like any other miss, so the fit barely moves.

Gaussian noise rarely produces outliers; Laplace noise does. Pick the loss to match.

Recap: gradient descent

Start at $w = 0$. Repeat: $w \leftarrow w - \eta \nabla \text{TrainLoss}(w)$

η is the step size. The number of steps T is another hyperparameter.

For squared loss:

$$\nabla \text{TrainLoss}(w) = \text{average of } 2 (w \cdot \phi(x) - y) \phi(x)$$

Each example pushes w along $\phi(x)$, in proportion to its residual.

An example we already predict exactly contributes 0.

One step of gradient descent

Data (0, 1), (1, 2), (2, 6). $\phi(x) = [1, x]$. Start $w = [0, 0]$, $\eta = 0.1$.

Residuals at $w = 0$: -1, -2, -6. TrainLoss = $41/3 \approx 13.7$

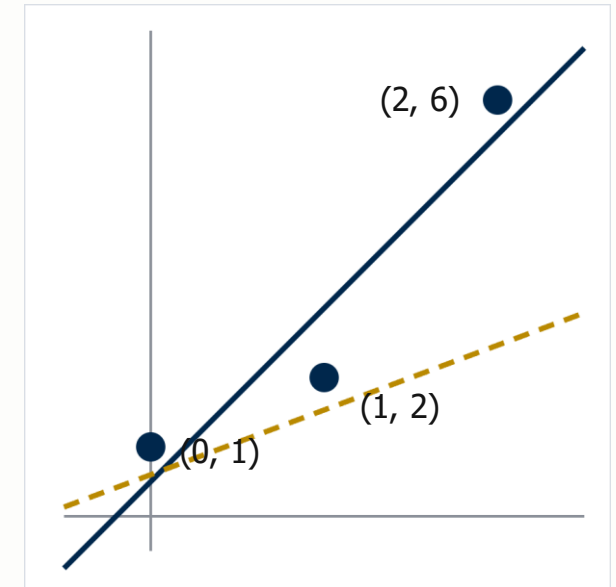
$$\nabla = (2/3)(-1 \cdot [1, 0] - 2 \cdot [1, 1] - 6 \cdot [1, 2]) = (2/3)[-9, -14] = [-6, -9.33]$$

$w \leftarrow [0, 0] - 0.1 \cdot [-6, -9.33] = [0.6, 0.93]$. TrainLoss ≈ 4.29

Q: Which example pushed w the most, and why?

A: (2, 6): it has the largest residual (-6) and the largest feature ($x = 2$).

After ~ 200 steps, $w \approx [0.5, 2.5]$, the least-squares line.



solid: optimum $0.5 + 2.5x$ · dashed: after one step, $0.6 + 0.93x$

Step size and SGD

Same data, 20 steps of gradient descent:

Step size η	w after 20 steps	TrainLoss
0.1	$\approx [0.77, 2.30]$, still moving	≈ 0.53
0.3	$\approx [0.52, 2.48]$	≈ 0.50
0.5	$\approx [-947, -1312]$	diverges

Q: Why does $\eta = 0.5$ blow up?

A: Each step overshoots the minimum by more than it started. Here $\eta > 0.42$ diverges.

SGD: update after each example, in random order. Noisier steps, many more of them.

Low training loss is not the goal

A memorizer returns y for training inputs and 0 otherwise: zero training loss.

A parabola through our 3 points also has zero training loss.

What we want: low error on new examples from the same distribution.

Q: You try 50 step sizes and keep the one with the best validation error. What's wrong?

A: The validation set helped choose the model, so its error is now optimistic.

Tune on a validation set. Touch the test set once, at the end.

Recap: linear classification

Labels $y \in \{-1, +1\}$. Predict $\text{sign}(w \cdot \phi(x))$.

Score $w \cdot \phi(x)$: how confidently we predict $+1$.

Margin $(w \cdot \phi(x)) y$: how correct we are. Positive = correct.

Zero-one loss: 1 if margin ≤ 0 . Flat almost everywhere: zero gradient.

Hinge loss: $\max(0, 1 - \text{margin})$. Zero once the margin is at least 1.

Logistic loss: $\log(1 + e^{(-\text{margin})})$. Never exactly zero.

Margins and losses by hand

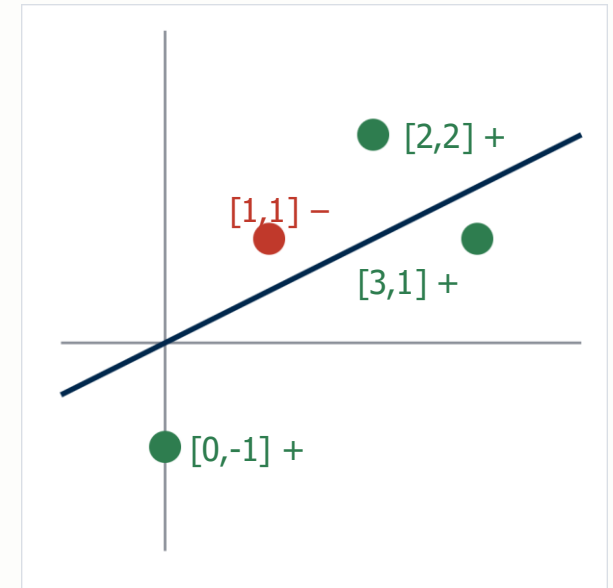
$w = [1, -2]$. $\phi(x) = x$.

x	y	Score	Margin	Hinge	Logistic
[3, 1]	+1	1	1	0	0.31
[1, 1]	-1	-1	1	0	0.31
[2, 2]	+1	-2	-2	3	2.13
[0, -1]	+1	2	2	0	0.13

Average: zero-one $1/4$, hinge 0.75 , logistic ≈ 0.72 .

Q: Which examples contribute to the hinge loss gradient?

A: Only [2, 2]. The others have margin ≥ 1 , so their hinge loss is flat at 0.



solid line: boundary $x_1 - 2x_2 = 0$ · green = +1, red = -1

One SGD step on the hinge loss

If margin < 1 : $\nabla \text{Loss} = -y \phi(x)$, so $w \leftarrow w + \eta y \phi(x)$

Update on the mistake $[2, 2]$, $y = +1$, $\eta = 0.5$:

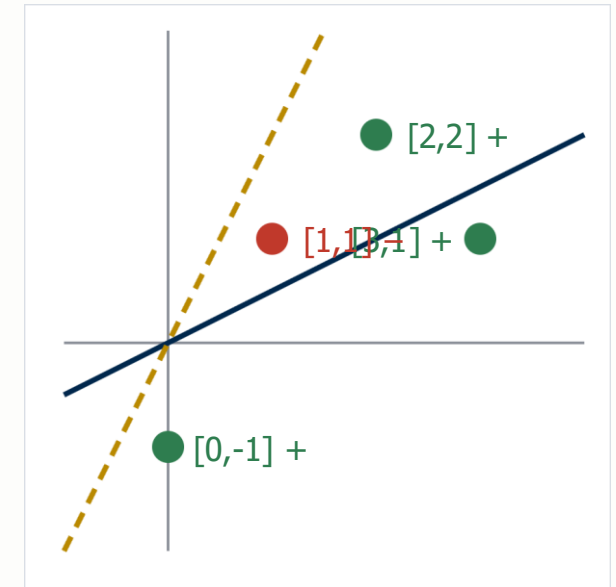
$$w \leftarrow [1, -2] + 0.5 \cdot (+1) \cdot [2, 2] = [2, -1]$$

New margins: $[3, 1] \rightarrow 5$, $[1, 1] \rightarrow -1$, $[2, 2] \rightarrow 2$, $[0, -1] \rightarrow 1$

Q: Did one update fix everything?

A: No. $[2, 2]$ is fixed, but $[1, 1]$ now has margin -1 : a new mistake.

That's normal for SGD. Later updates on $[1, 1]$ will push back.



solid: old boundary, $w = [1, -2]$ · dashed:
new, $w = [2, -1]$

Predicting probabilities: logistic regression

Treat the score as log-odds: $P(y = +1 \mid x) = \sigma(w \cdot \phi(x))$, $\sigma(z) = 1 / (1 + e^{-z})$

$\sigma(0) = 0.5$. $\sigma(2) \approx 0.88$. $\sigma(-1) \approx 0.27$.

Maximum likelihood $\rightarrow$ minimize $\log(1 + e^{-\text{margin}})$: the logistic loss.

Q: With $w = [1, -2]$, what is $P(y = +1)$ for $x = [3, 1]$? For $x = [2, 2]$?

A: Scores 1 and -2: $\sigma(1) \approx 0.73$ and $\sigma(-2) \approx 0.12$.

HW2: you derive the gradient of this loss and train the classifier.

Recap: where does test error come from?

f^* : best possible predictor. g : best in our class F . $\hat{f}$: what we learn.

Approximation error: how good is the best predictor in F ?

Estimation error: how well did we pick within F from limited data?

Q: Which error dominates each case?

Situation	Main problem
A line fit to U-shaped data, 1 million points	Approximation
Degree-20 polynomial fit to 10 points	Estimation

Bigger F : approximation $\downarrow$, estimation $\uparrow$. The sweet spot depends on data size.

Lucky predictors

F has 500 predictors that guess at random: 50% error on new data.

We have $n = 8$ training examples.

One predictor gets all 8 right with probability $(1/2)^8 = 1/256$.

Q: What is the chance that at least one predictor gets all 8 right?

A: $1 - (255/256)^{500} \approx 0.86$.

ERM would pick it: zero training loss, 50% test error.

More predictors $\rightarrow$ more chances for a bad one to look good by luck.

The PAC bound

Finite F , zero-one loss, some perfect predictor in F , i.i.d. data.

If $n \geq (1/\epsilon)(\ln |F| + \ln(1/\delta))$, then with probability $\geq 1 - \delta$, ERM's error $\leq \epsilon$.

Example: $|F| = 10^6$, $\epsilon = 0.05$, $\delta = 0.01$:

$n \geq 20 \cdot (13.8 + 4.6) \approx 369$ examples

Q: How many more examples if $|F|$ doubles? If δ drops to 0.0001?

A: Doubling $|F|$: + $20 \cdot \ln 2 \approx 14$. Smaller δ : + $20 \cdot \ln 100 \approx 92$.

Cost grows with $\log |F|$ and $\log(1/\delta)$, but with $1/\epsilon$.

What about infinite classes?

Lines are infinite, but on a computer each weight is a 32-bit float.

d weights $\rightarrow |F| \leq 2^{(32d)}$, so $\ln |F| \leq 32 d \ln 2 \approx 22d$.

Q: $d = 10$ features, $\epsilon = 0.1$, $\delta = 0.05$. Roughly how many examples?

A: $n \geq 10 \cdot (222 + 3) \approx 2,250$. Linear in the number of features.

A memorizer can represent every function of x : $\ln |F|$ grows with the number of inputs.

Neural networks have more parameters than examples, yet generalize. Open question!

Features: linear in w, not in x

Energy use vs. outside temperature: heating below 18 °C, cooling above.

$$\phi(x) = [1, \max(0, 18 - x), \max(0, x - 18)]$$

Suppose training gives $w = [50, 4, 6]$.

Q: Predicted energy at 8 °C, 18 °C, and 28 °C?

A: 8 °C: $50 + 4 \cdot 10 = 90$. 18 °C: 50. 28 °C: $50 + 6 \cdot 10 = 110$.

The score is a V shape in x, but still linear in w: same loss, same gradient descent.

Good features encode what we know about the problem.

When a line can't separate the data

XOR: (0,1) and (1,0) are +1. (0,0) and (1,1) are -1.

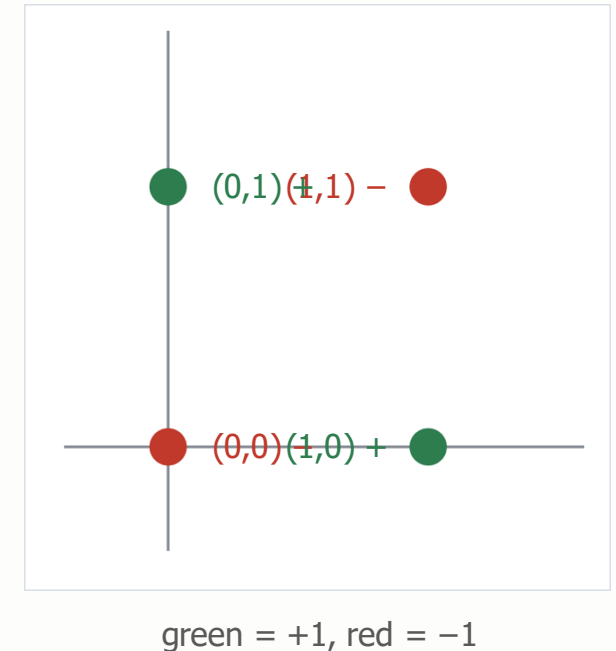
Q: Can $\text{sign}(w \cdot [1, x_1, x_2])$ classify all four?

A: No. Any line that puts both + points on one side puts a - point there too.

Add a feature: $\phi(x) = [1, x_1, x_2, x_1x_2]$, $w = [-0.5, 1, 1, -2]$

(0,0): -0.5 (1,0): 0.5 (0,1): 0.5 (1,1): -0.5 All four correct.

Feature engineering works, but someone has to think of x_1x_2 .



Recap: two-layer neural networks

Hidden layer: $h = \sigma(V \phi(x))$, one hidden unit per row of V .

Output: score = $w \cdot h$. Regression: the score. Classification: its sign.

The last layer is a linear predictor on learned features h .

$\sigma = \text{ReLU}(z) = \max(0, z)$ or sigmoid. A hard threshold has zero gradient.

Q: Why not use $\sigma(z) = z$?

A: Then score = $w \cdot (V \phi(x)) = (V^T w) \cdot \phi(x)$: just another linear predictor.

A tiny network that solves XOR

$$h_1 = \text{ReLU}(x_1 + x_2), \quad h_2 = \text{ReLU}(x_1 + x_2 - 1), \quad \text{score} = h_1 - 2h_2 - 0.5$$

x	h₁	h₂	Score	Prediction
(0, 0)	0	0	-0.5	-1
(1, 0)	1	0	0.5	+1
(0, 1)	1	0	0.5	+1
(1, 1)	2	1	-0.5	-1

Q: What do the hidden units compute?

A: h_1 counts the inputs that are on. h_2 is 1 only when both are on.

The output layer is linear in $[h_1, h_2]$. The network built its own x_1x_2 -like feature.

Summary

Supervised learning = hypothesis class + loss + optimization.

Regression: squared loss (Gaussian MLE), solved by (stochastic) gradient descent.

Classification: sign of the score. Hinge and logistic losses replace zero-one.

Test error = best possible error + approximation error + estimation error.

PAC: $n \geq (1/\epsilon)(\ln |F| + \ln(1/\delta))$ examples suffice for finite F .

Features make linear models nonlinear in x . Neural networks learn the features.

HW2 is due Wednesday, October 7. Next week: deep learning.