

Machine Learning

SGD (continue) and Linear Regression

DSC 240

Jan 28, 2025

Instructor: Prof. Yu-Xiang Wang

Last lecture: How to learn a linear classifier?

- Non-linearly separable case: NP-hard.
 - Our solution: “just relax” from 0-1 loss (i.e., mistakes), into surrogate loss minimization.
- Logistic loss: a convex and smooth upper bound of 0-1 loss.
- Continuous optimization problem and gradient Descent
- Faster than gradient descent: stochastic gradient descent

Recap: Gradient Descent and Stochastic Gradient Descent

- Gradient descent

$$\theta_{t+1} = \theta_t - \eta_t \nabla f(\theta_t)$$

- Stochastic gradient descent

$$\theta_{t+1} = \theta_t - \eta_t \hat{\nabla} f(\theta_t)$$

- One way to construct the Stochastic Gradient


1. Pick a **single** data point i uniformly at random

2. Calculate $\nabla_{\theta} \ell(\theta, (x_i, y_i))$

Intuition of the SGD algorithm with logistic loss

$$\nabla \ell(w, (x_i, y_i)) = \frac{\exp(-y_i \cdot x_i^T w)}{1 + \exp(-y_i \cdot x_i^T w)} (-y_i x_i)$$

Intuition of the SGD algorithm with logistic loss

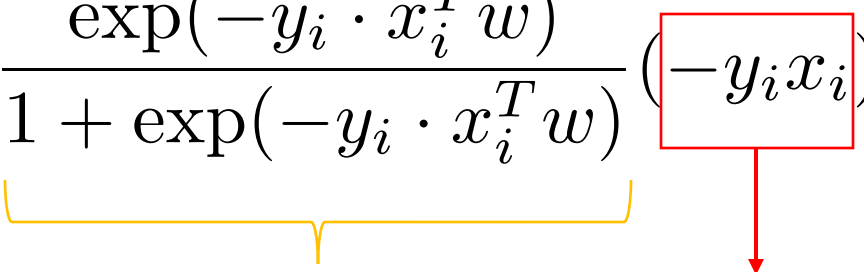
$$\nabla \ell(w, (x_i, y_i)) = \frac{\exp(-y_i \cdot x_i^T w)}{1 + \exp(-y_i \cdot x_i^T w)} (-y_i x_i)$$


Scalar > 0 :

≈ 0 if the prediction
is correct

≈ 1 otherwise

Intuition of the SGD algorithm with logistic loss

$$\nabla \ell(w, (x_i, y_i)) = \frac{\exp(-y_i \cdot x_i^T w)}{1 + \exp(-y_i \cdot x_i^T w)} \underbrace{\left(-y_i x_i \right)}$$


Scalar > 0 :
 ≈ 0 if the prediction
is correct
 ≈ 1 otherwise

Vector of dimension d :
provides the direction
of the gradient

Intuition of the SGD algorithm with logistic loss

$$\nabla \ell(w, (x_i, y_i)) = \frac{\exp(-y_i \cdot x_i^T w)}{1 + \exp(-y_i \cdot x_i^T w)} \underbrace{\left(-y_i x_i \right)}$$

Scalar > 0 :
 ≈ 0 if the prediction
is correct
 ≈ 1 otherwise

Vector of dimension d :
provides the direction
of the gradient

If we receive an example $[1, 0, 0.0375, 80]$ like the one before.
And a label $y = 1$ saying that this is a spam.

How will the SGD update change the weight vector?

Intuition of the SGD algorithm with logistic loss

$$\nabla \ell(w, (x_i, y_i)) = \frac{\exp(-y_i \cdot x_i^T w)}{1 + \exp(-y_i \cdot x_i^T w)} \underbrace{\left(-y_i x_i \right)}$$

Scalar > 0:
≈ 0 if the prediction
is correct
≈ 1 otherwise

Vector of dimension d:
provides the direction
of the gradient

If we receive an example [1, 0, 0.0375, 80] like the one before.
And a label $y = 1$ saying that this is a spam.

How will the SGD update change the weight vector?

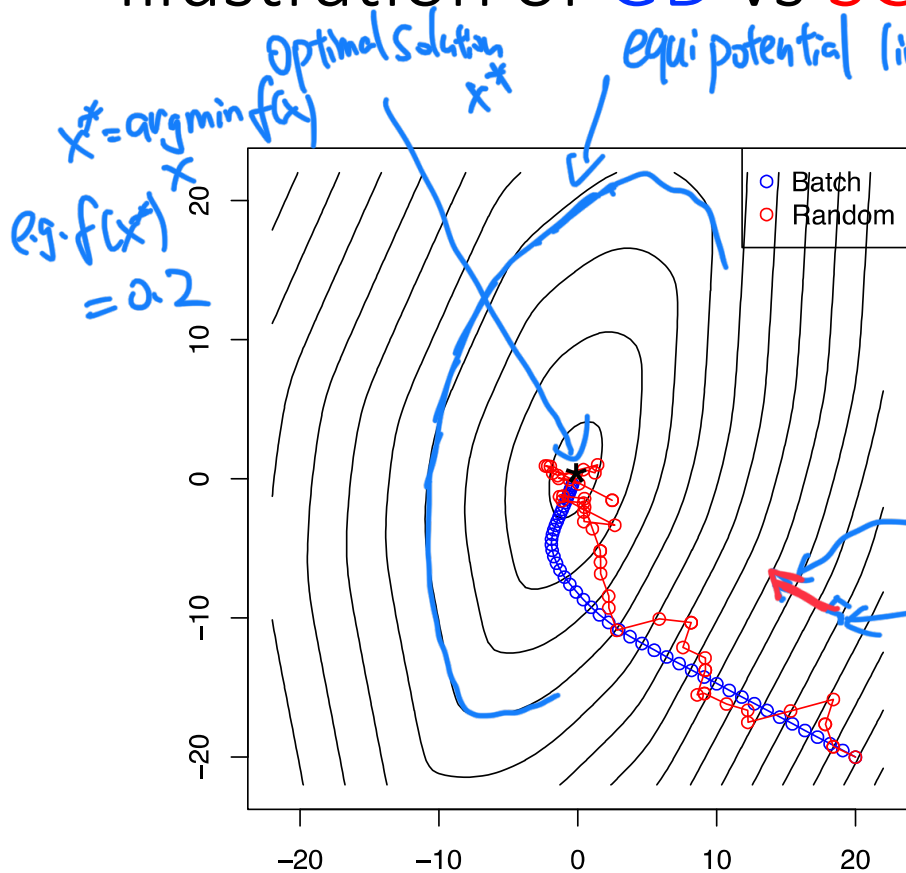
Then by moving w towards the negative gradient direction, we are changing the weight vector by increasing the weights. i.e., increasing the amount they contribute to the score function (if currently the classifier is making a mistake on this example)

Today

- Finishing SGD
 - Connection to Perceptron
 - What learning rate to use for SGD?
 - Practical tricks for getting SGD to work
- Linear regression

Illustration of GD vs SGD

e.g. $\{x \mid f(x) = 0.7\}$



Rule of thumb for stochastic methods:

- generally thrive far from optimum
- generally struggle close to optimum

Quiz: What is the running time comparison?

Observation: With the time gradient descent taking one step. SGD would have already moved many steps.

Interpretation of GD vs SGD for learning linear classifier

- GD:
$$w_{t+1} = w_t - \eta_t \frac{1}{n} \sum_{i=1}^n \nabla_w \ell(w_t, (x_i, y_i))$$

- Average the “feedback signal” from all examples then update w .
- If examples don’t agree with each other, poll them, then commit to change.

- SGD:

$$w_{t+1} = w_t - \eta_t \nabla_w \ell(w_t, (x_i, y_i))$$

- Update w after seeing every example! (just like Perceptron)
- Learning from more incremental feedback

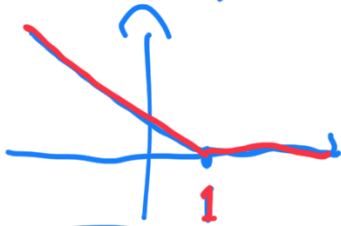
Is human learning more like GD or SGD?

Perceptron Algorithm can be viewed as a variant of SGD

perceptron $w_{t+1} = w_t + y_i x_t \cdot \mathbb{1}(w^T x_t < 0)$

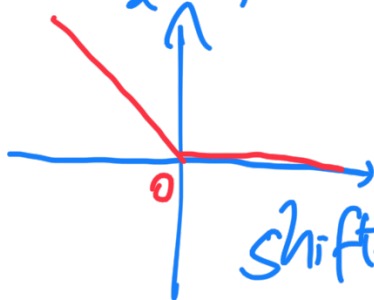
- It uses a shifted version of "Hinge loss" --- aka "Perceptron loss" $= w_t - y \cdot \nabla l(w)$ ^{mistake}
for some l ?

Hinge loss $l(w) = \max\{0, 1 - y \cdot x^T w\}$



$$\nabla l(w) = \begin{cases} (-1) \times (-y \cdot x) = y \cdot \vec{x} & \text{when } y \cdot x^T w < 1 \\ 0 \times (-y \cdot \vec{x}) = \vec{0} & \text{when } y \cdot x^T w \geq 1 \end{cases}$$

Perceptron loss $l(w) = \max\{0, -y \cdot x^T w\}$



shifted to the left by 1.

$$\nabla l_{\text{perceptron}}(w) = \begin{cases} y \cdot \vec{x} & \text{if } y \cdot x^T w < 0 \\ 0 & \text{otherwise.} \end{cases}$$

learning rate $\eta = 1$

Perceptron Algorithm can be viewed as a variant of SGD

- It uses a shifted version of “Hinge loss” --- aka “Perceptron loss”
- **Exercise (high chance to be in midterm):** Calculate the gradient of the “Perceptron loss”. Show that the update rule after a single data point is selected is the same as the Perceptron’s algorithm when learning rate = 1.

Perceptron Algorithm can be viewed as a variant of SGD

- It uses a shifted version of “Hinge loss” --- aka “Perceptron loss”
- **Exercise (high chance to be in midterm):** Calculate the gradient of the “Perceptron loss”. Show that the update rule after a single data point is selected is the same as the Perceptron’s algorithm when learning rate = 1.
- Important differences to SGD
 1. Perceptron does not require the update direction to be an “unbiased” estimate. It could go with the iid sampling version or shuffle-then-“for”-over version.
 2. It does not require a choice of learning rate.
 3. It does not aim at optimizing an objective function (thus does not necessarily succeed in doing so).

Do GD and SGD come with provable guarantees?

- Yes! When the objective function is convex

Do GD and SGD come with provable guarantees?

- Yes! When the objective function is convex

also if f is twice differentiable
 $\lambda_{\max}(\nabla^2 f) \leq L$
largest eigenvalue

Theorem (GD): Assume $f(\theta)$ is **convex** and **L -gradient Lipschitz**, i.e.,
 $\nabla f(\theta) - \nabla f(\theta') \leq L\|\theta_1 - \theta_2\|$ for any θ, θ' .

Then if $\eta \leq 1/L$, the T^{th} iteration of the GD returns θ_T that satisfies

$$f(\theta_T) - \min_{\theta} f(\theta) \leq \frac{\|\theta_1 - \theta^*\|^2}{2\eta T}.$$

Do GD and SGD come with provable guarantees?

- Yes! When the objective function is convex

Theorem (GD): Assume $f(\theta)$ is **convex** and **L -gradient Lipschitz**, i.e., $\nabla f(\theta) - \nabla f(\theta') \leq L\|\theta_1 - \theta_2\|$ for any θ, θ' .

Then if $\eta \leq 1/L$, the T^{th} iteration of the GD returns θ_T that satisfies

$$f(\theta_T) - \min_{\theta} f(\theta) \leq \frac{\|\theta_1 - \theta^*\|^2}{2\eta T}.$$

Theorem (SGD): Convergence of Gradient Descent. Assume $f(\theta)$ is **convex** and $\|\nabla f(\theta)\| \leq G \forall \theta$. And in addition, the stochastic gradient satisfies that (1) $\mathbb{E}[\widehat{\nabla}f(\theta)] = \nabla f(\theta)$ and (2) $\mathbb{E}[\|\widehat{\nabla}f(\theta) - \nabla f(\theta)\|^2] \leq \sigma^2$ when given any θ .

Then for a sequence of learning rate $\eta_1, \eta_2, \dots, \eta_t, \dots$

$$\min_{t \in [T]} \mathbb{E}[f(\theta_t) - f^*] \leq \frac{\|\theta_1 - \theta^*\|^2 + (G^2 + \sigma^2) \sum_t \eta_t^2}{2 \sum_{t=1}^T \eta_t}$$

Do GD and SGD come with provable guarantees?

- Yes! When the objective function is convex

Theorem (GD): Assume $f(\theta)$ is **convex** and **L -gradient Lipschitz**, i.e., $\nabla f(\theta) - \nabla f(\theta') \leq L\|\theta_1 - \theta_2\|$ for any θ, θ' .

Then if $\eta \leq 1/L$, the T^{th} iteration of the GD returns θ_T that satisfies

$$f(\theta_T) - \min_{\theta} f(\theta) \leq \frac{\|\theta_1 - \theta^*\|^2}{2\eta T}.$$

Theorem (SGD): Convergence of Gradient Descent. Assume $f(\theta)$ is **convex** and $\|\nabla f(\theta)\| \leq G \forall \theta$. And in addition, the stochastic gradient satisfies that (1) $\mathbb{E}[\widehat{\nabla}f(\theta)] = \nabla f(\theta)$ and (2) $\mathbb{E}[\|\widehat{\nabla}f(\theta) - \nabla f(\theta)\|^2] \leq \sigma^2$ when given any θ .

Then for a sequence of learning rate $\eta_1, \eta_2, \dots, \eta_t, \dots$

$$\min_{t \in [T]} \mathbb{E}[f(\theta_t) - f^*] \leq \frac{\|\theta_1 - \theta^*\|^2 + (G^2 + \sigma^2) \sum_t \eta_t^2}{2 \sum_{t=1}^T \eta_t}$$

- Take a more advanced course in machine learning or convex optimization to learn how to prove the above theorems!
- Tons of new modern research on this that comes out at ICML, NeurIPS, ICLR, AISTATS every year...

(proofs in “convex optimization” course: DSC 243 and CSE203B)

How to choose the step sizes / learning rates?

- In theory:

- Gradient decent: $1/L$ where L is the **Gradient Lipschitz constant** of the function we minimize.

- SGD:
$$\sum_t \eta_t = \infty, \sum_t \eta_t^2 < \infty$$

- e.g.
$$\eta_t \in [1/t, 1/\sqrt{t})$$

- In practice:

- Use cross-validation on a subsample of the data.
- Fixed learning rate for SGD is usually fine.
- If it diverges, decrease the learning rate.
- If for extremely small learning rate, it still diverges, check if your gradient implementation is correct.

Diagnosing learning algorithm by plotting the “Learning Curve”

Demo of GD / SGD and their learning curves.

The power of SGD

- Extremely general:
 - Specify an end-to-end differentiable score function, e.g., a complex neural network.
 - Beyond the context of machine learning
- Extremely simple: a few lines of code.
- Extremely scalable
 - Just a few pass of the data, no need to store the data
- People are continuing to discover that many methods are special cases of SGD.

Checkpoint: Algorithms for learning linear classifiers

	Perceptron	Gradient Descent (with Logistic loss)	Stochastic GD (with Logistic loss)
Assumption	Margin > 0	n.a.	n.a.
Computational complexity	$O(n/\text{Margin}^2)$	$O(nd * \text{Num of Iterations})$	$O(d * \text{Number of iterations})$
Optimization view	Minimize Avg Perceptron Loss using Online GD with learning rate = 1	Minimize Avg Logistic loss	Minimize Avg Logistic loss
Hyperparameters	None	Learning rate	Learning rate
Guarantees	Reach 0 error eventually	Reach approx minimum loss Subopt = $O(1/\text{Num Iterations})$	Reach approx minimum loss $E[\text{Subopt}] = O(1/\sqrt{\text{num iter}})$
Pros / cons	<ul style="list-style-type: none"> • Works even in online / adversarial setting • Unstable when assumption is false 	<ul style="list-style-type: none"> • Consistent descent • Slow • Does not minimize classification error 	<ul style="list-style-type: none"> • Fast to get good approx • Slow to get highly accurate minimizer • Does not minimize classification error

Today

- ~~Finishing SGD~~
 - ~~Connection to Perceptron~~
 - ~~What learning rate to use for SGD?~~
 - ~~Practical tricks for getting SGD to work~~
- Linear regression

Recap: The supervised learning problem

- Feature space: \mathcal{X}
- Label space: \mathcal{Y}
- A classifier (hypothesis): $h : \mathcal{X} \rightarrow \mathcal{Y}$
- A hypothesis class: \mathcal{H}
- Data: $(x_1, y_1), \dots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$
- Learning task: Find $h \in \mathcal{H}$ that “works well”.

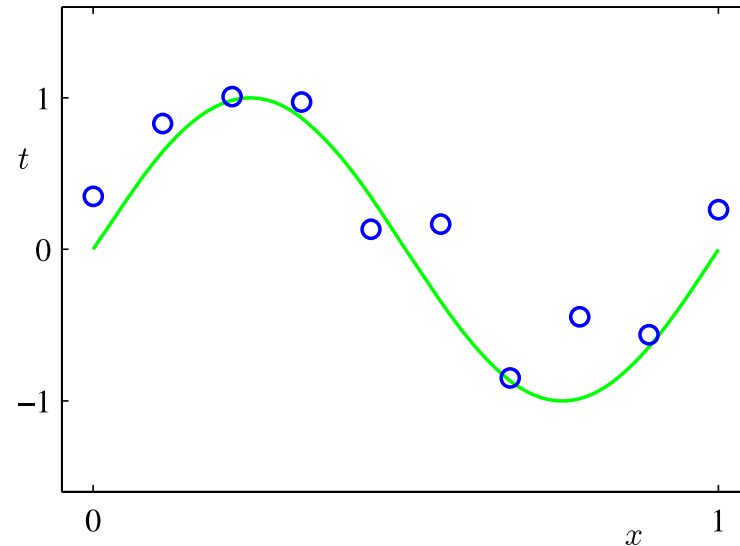
Examples of supervised learning problems

	Binary classification	Multi-class classification	Regression
Feature space	\mathbb{R}^d	\mathbb{R}^d	\mathbb{R}^d
Label space	$\{-1, 1\}$	$\{1, 2, 3, \dots, K\}$	\mathbb{R}
Popular Performance metric	Classification error (0-1 loss) for test data	Classification error (0-1 loss) for test data	?
Popular surrogate loss (for training)	Logistic loss	Multiclass logistic loss aka. Cross-Entropy loss	?

So far we've only talked about binary classification.
Multi-class classification will be derived after the midterm.
We will focus on "regression" today.

“Regression” Example 1: Curve Fitting

Figure 1.2 Plot of a training data set of $N = 10$ points, shown as blue circles, each comprising an observation of the input variable x along with the corresponding target variable t . The green curve shows the function $\sin(2\pi x)$ used to generate the data. Our goal is to predict the value of t for some new value of x , without knowledge of the green curve.



- What are the feature space, label space?
- What is a reasonable hypothesis class to use?

Examples of hypothesis classes for this problem

- Polynomials

$$h(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$

- Sine function

$$h(x, t) = \sin(2\pi t)$$

- Anything else?

Polynomial regression under square loss

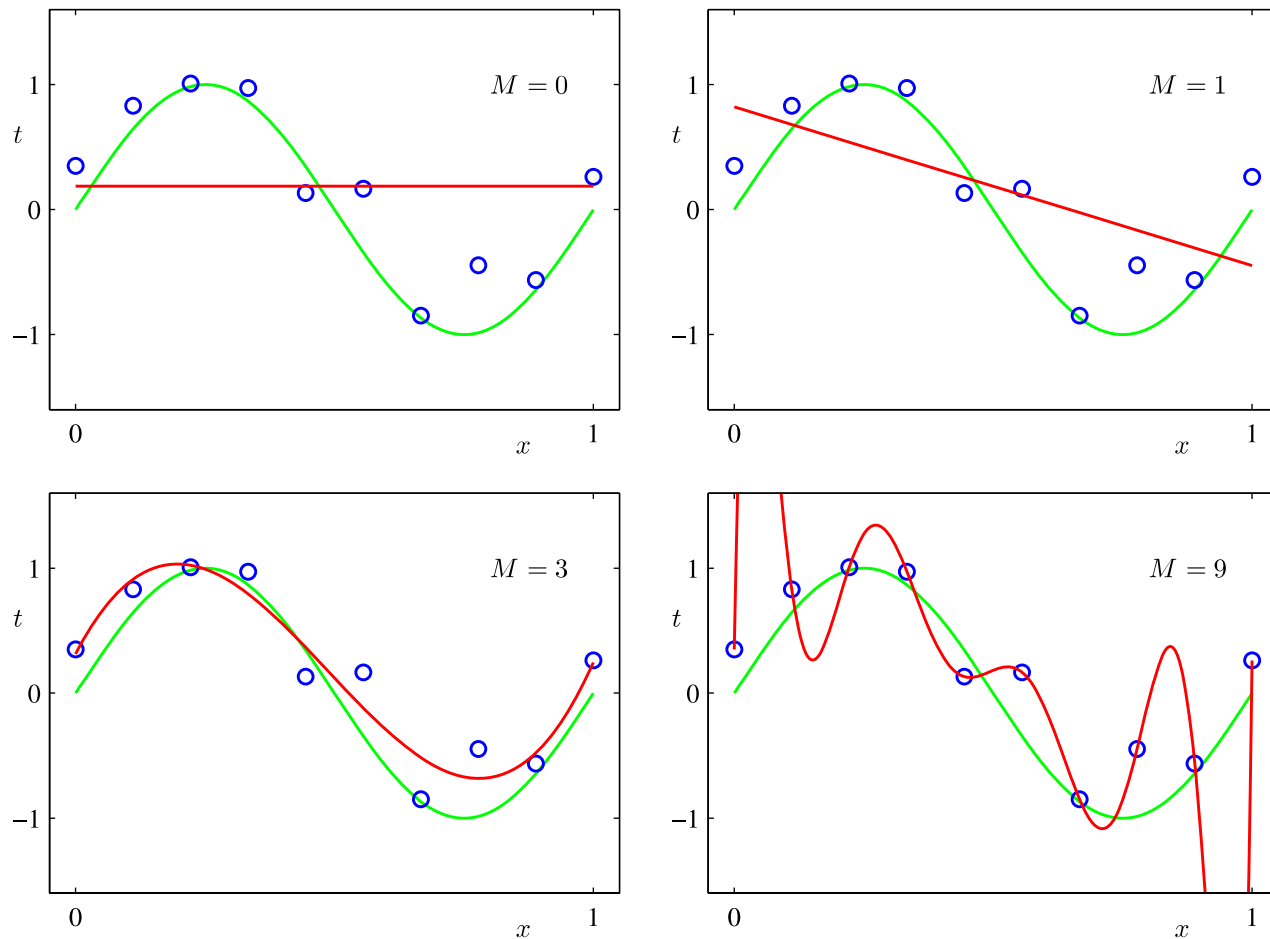
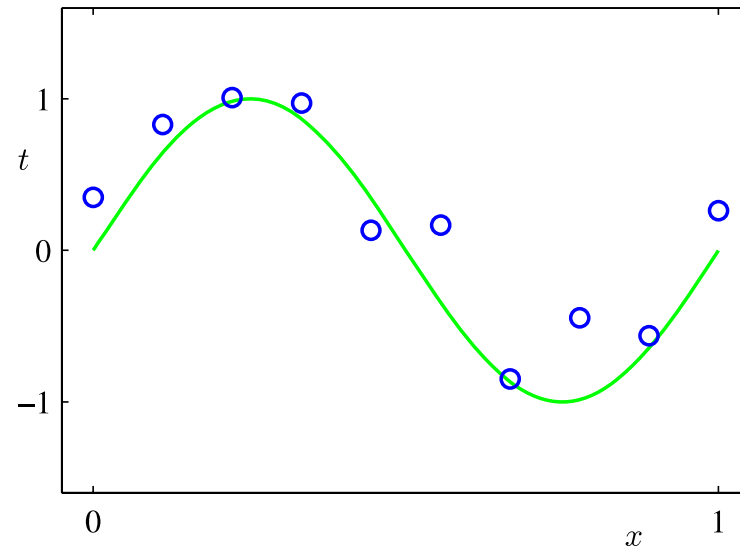


Figure 1.4 Plots of polynomials having various orders M , shown as red curves, fitted to the data set shown in Figure 1.2.

Discussion: What should be the performance metric for “curve fitting”?

Figure 1.2 Plot of a training data set of $N = 10$ points, shown as blue circles, each comprising an observation of the input variable x along with the corresponding target variable t . The green curve shows the function $\sin(2\pi x)$ used to generate the data. Our goal is to predict the value of t for some new value of x , without knowledge of the green curve.



Performance metrics for Regression problems

- Prediction problem
 - How well can one predict label y ?
 - Performance metric of interest:

- Estimation / inference problem
 - How well can one estimate the true function?
 - Need to make assumptions about how the observations are connected to the true function, typically $y = f_0(x) + noise$
 - Performance metric of interest:

How to solve “curve fitting” given a hypothesis class?

- Challenge:
 - We don't have access to future data for prediction!
 - We also don't have access to ground truth f_0
- By solving an optimization problem that **minimizes the loss function on the training data**, and hope that it generalizes.
 - Notice that we can verify if it generalizes or not using hold-out / cross-validation...

How to solve “curve fitting” given a hypothesis class?

- Challenge:
 - We don't have access to future data for prediction!
 - We also don't have access to ground truth f_0
- By solving an optimization problem that **minimizes the loss function on the training data**, and hope that it generalizes.
 - Notice that we can verify if it generalizes or not using hold-out / cross-validation...
- The “least square” objective function:

“Regression” example 2: Linear regression

- Feature space
- Label space
- Hypothesis space
- Loss function

Polynomial “Curve fitting” can be casted as a linear regression problem.

- What are the features?

- What is the linear score function?

The objective function for learning linear regression under square loss

$$\hat{\theta} = \arg \min_{\theta} \frac{1}{n} \sum_{i \in [n]} (x_i^T \theta - y_i)^2$$

- aka: Ordinary Least square (OLS)
- There is a convenient form using linear algebra

Exercise 2: What's the GD update rule?

- GD update rule?

- Apply linear algebra to simplify the expression in matrix form.

Direct solution to the linear regression problem by solving a linear system of equations

- Recall the matrix form of the gradient?

- How to solve an optimization problem by hand?
 - In the univariate case (from high school). Set derivative to 0
 - In the multi-variate case: we can set gradient to 0.
 - This returns the optimal solution if the problem is convex.

Comparing SGD and the direct solver

- Time complexity of direct solver
 - The smaller of $O(d^2 n)$ or $O(d n^2)$.
 - This is $O(n^3)$ when d and n are on the same order.
- Time complexi
 - GD: $O(\min(dn, d^2) * \text{number of iterations})$
 - SGD $O(d * \text{number of iterations})$

Summary of the lecture

- SGD
 - Perceptron as a special case of SGD
 - Choice of learning rate and convergence

- Linear regression
 - The “curve fitting” problem
 - Hypothesis class, performance metric
 - The optimization problem to solve
 - SGD / GD and the closed-form-solution

- Next lecture:
 - Regularization
 - Start “Probability / Statistics review”